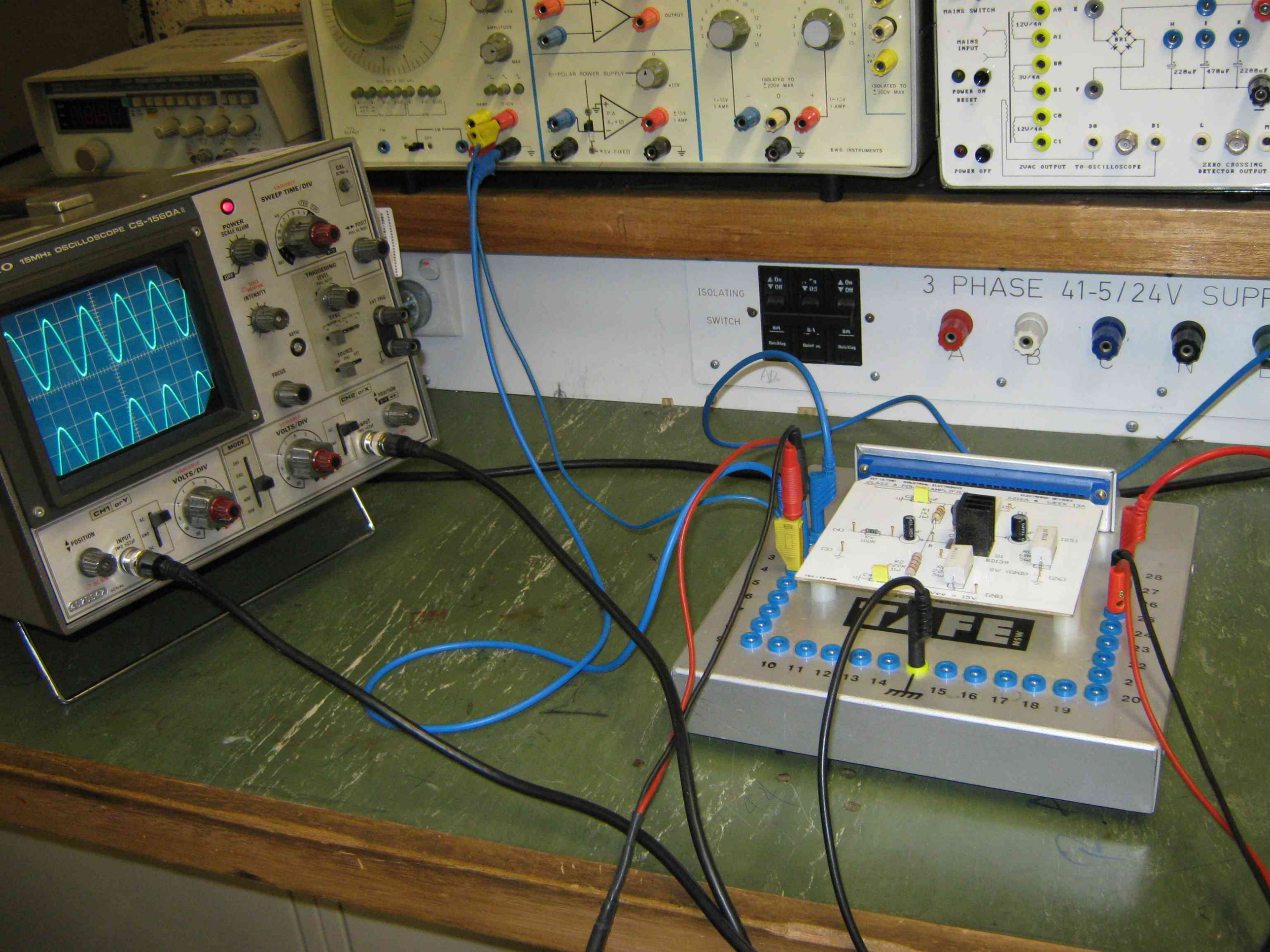
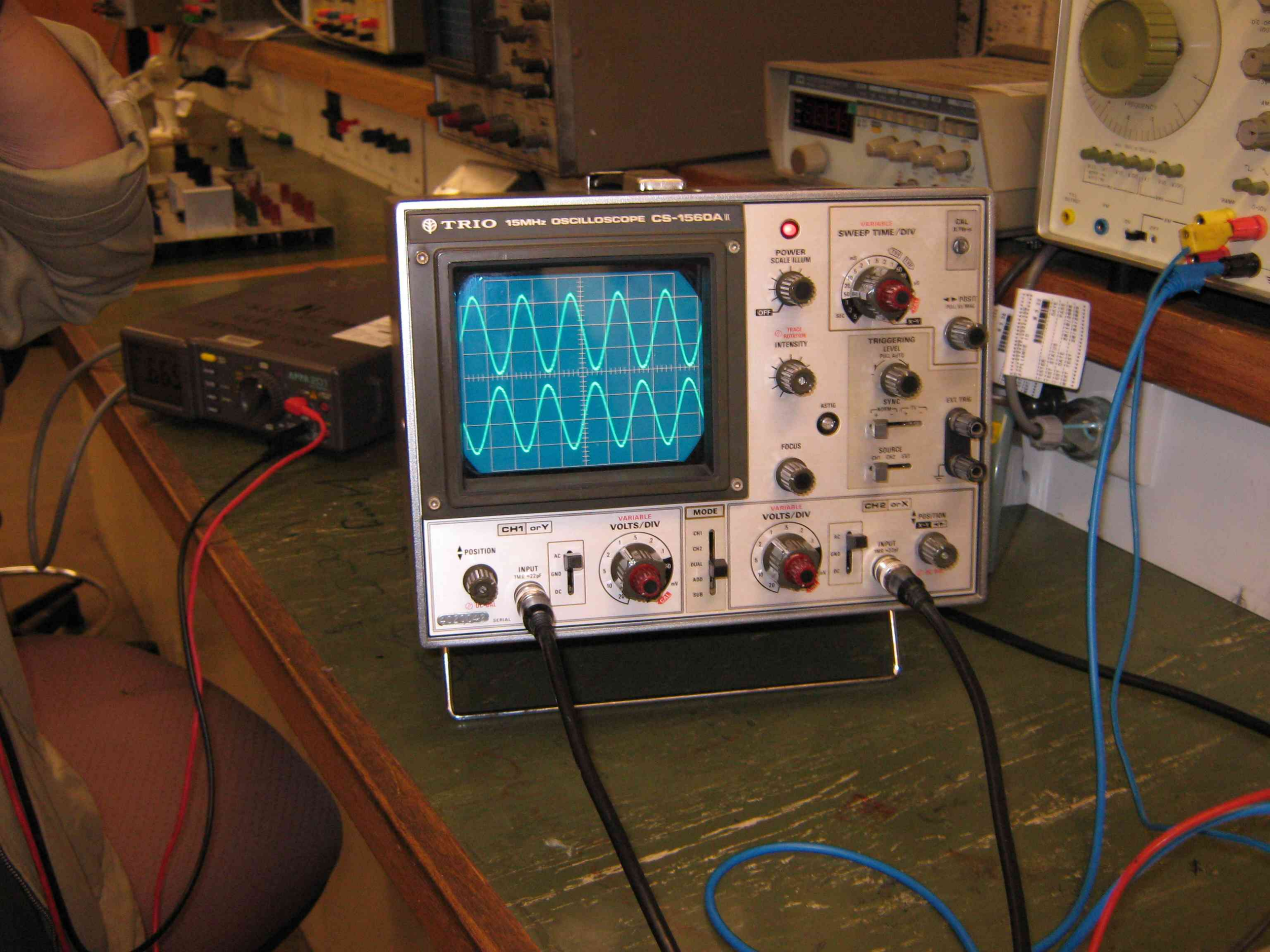

EMTEK
20MHz OSCILLOSCOPE
520
POWER
ON
OFF
INTENSITY
TRACE ROTATION
FOCUS
CAL
.5V
VERTICAL
POSITION
A
ADD
B
DUAL
POSITION
INVERT
VOLTS/DIV
VARIABLE
AC
GND
DC
INPUT A or Y
INPUT B or X
COMP. TEST
IN
OUT
1MΩ
7VA Max
15V
00.3
FUNCTION
RANGE
POWER ON-OFF
INPUT
SUNRA DIGITAL MULTIMETER
CONTROLLED BY MAIN SWITCH Nº1
240V G.P.O.
240V G.P.O.
240V MAIN SWITCH Nº1
NEUTRAL
EARTH
240V MAIN SWITCH Nº2
240V G.P.O.


SWEEP TIME/DIV
TRIGGERING LEVEL
INTENSITY
FOCUS
VOLTS/DIV
MODE
POSITION
INPUT
CH1 orY
CH2 orX
BI-POLAR POWER SUPPLY
ISOLATED TO ±200V MAX
MAINS SWITCH
MAINS INPUT
POWER ON RESET
POWER OFF
ZVAC OUTPUT
TO OSCILLOSCOPE
ZERO CROSSING DETECTOR OUTPUT
ISOLATING SWITCH
3 PHASE 41-5/24V SUPP
A
B
C
N


TRIO 15MHz OSCILLOSCOPE CS-1560A II
POWER
SCALE ILLUM
OFF
INTENSITY
FOCUS
VARIABLE
SWEEP TIME/DIV
TRIGGERING
LEVEL
SYNC
SOURCE
EXT TRIG
CH1 or Y
POSITION
INPUT
AC
GND
DC
VARIABLE
VOLTS/DIV
MODE
CH1
CH2
DUAL
ADD
SUB
CH2 or X

- Don't draw on bench
- Put equipment away
- Hang leads neat and tidy

MINI-LAB
MODEL 603B
FUNCTION GENERATOR
REGULATED POWER SUPPLY
LOW VOLTAGE AC/DC PWR SUPPLY
DUAL TRACKING DC POWER SUPPLY
3 PHASE 41-5/24V SUPPLY
ISOLATING SWITCH

- Put equipment away
- Hang leads neat and tidy

TRIO 15MHz OSCILLOSCOPE CS-1560A II

FUNCTION GENERATOR

MINI-LAB MODEL 603B

REGULATED POWER SUPPLY

LOW VOLTAGE AC/DC PWR SUPPLY

SYDNEY INSTITUTE OF TECHNOLOGY

Class (A) power Amplifier

Class (D) switch Mode circuit

ELECTRONIC DEVICES

6016A # WEEK 13B

C2 0.1uF

[1]

3 10R 1W +Vcc = +15V

Q1
BD139

E

R1
1R
1W

C1

[4]

B

[25]

R2
1R
1W

RL
10R
5W

Q2
BD140

[3] 0V

0V

C3 0.1uF C R4 10R 1W

e = -15V

2 3 4 6 7 8 9

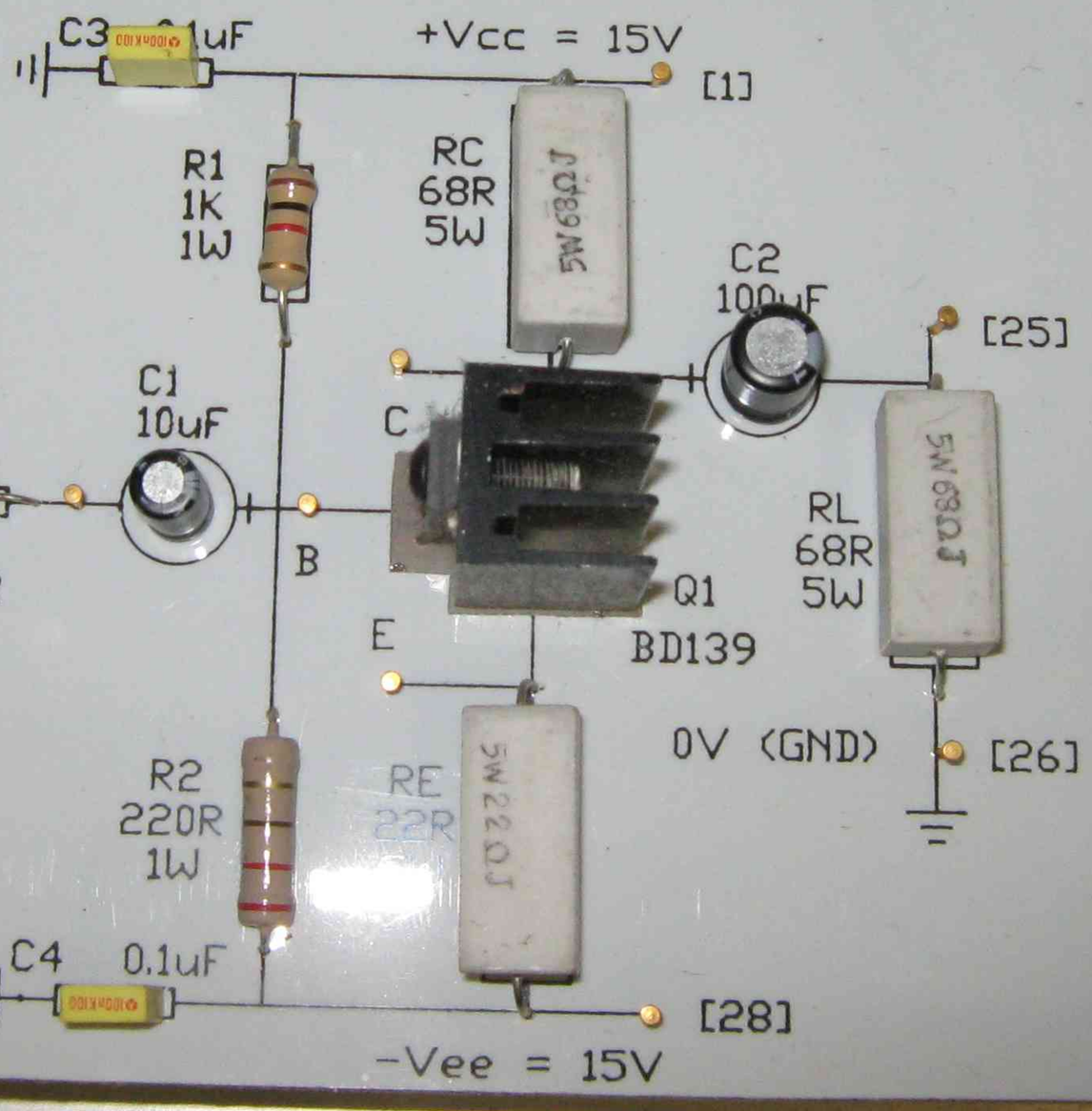
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

SIT ULTIMO INDUSTRIAL ELECTRONICS

ELECTRONIC DEVICES

CLASS A POWER AMPLIFIER

6016A # WEEK 13A

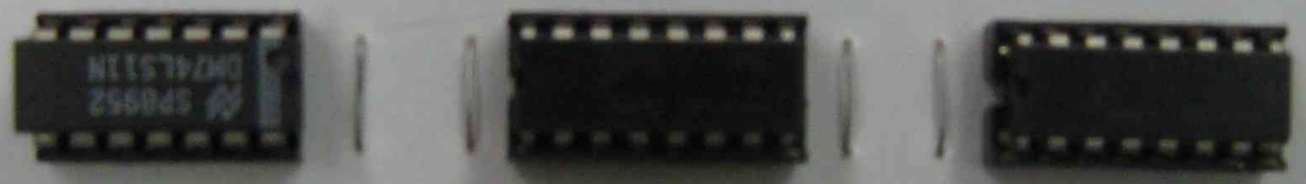


SYDNEY INSTITUTE OF TECHNOLOGY, ULTIMO
ENGINEERING SERVICES TRAINING DIVISION
ELECTROTECHNOLOGY, INDUSTRIAL ELECTRONICS
DIGITAL PRINCIPLES 6016B WEEK 14

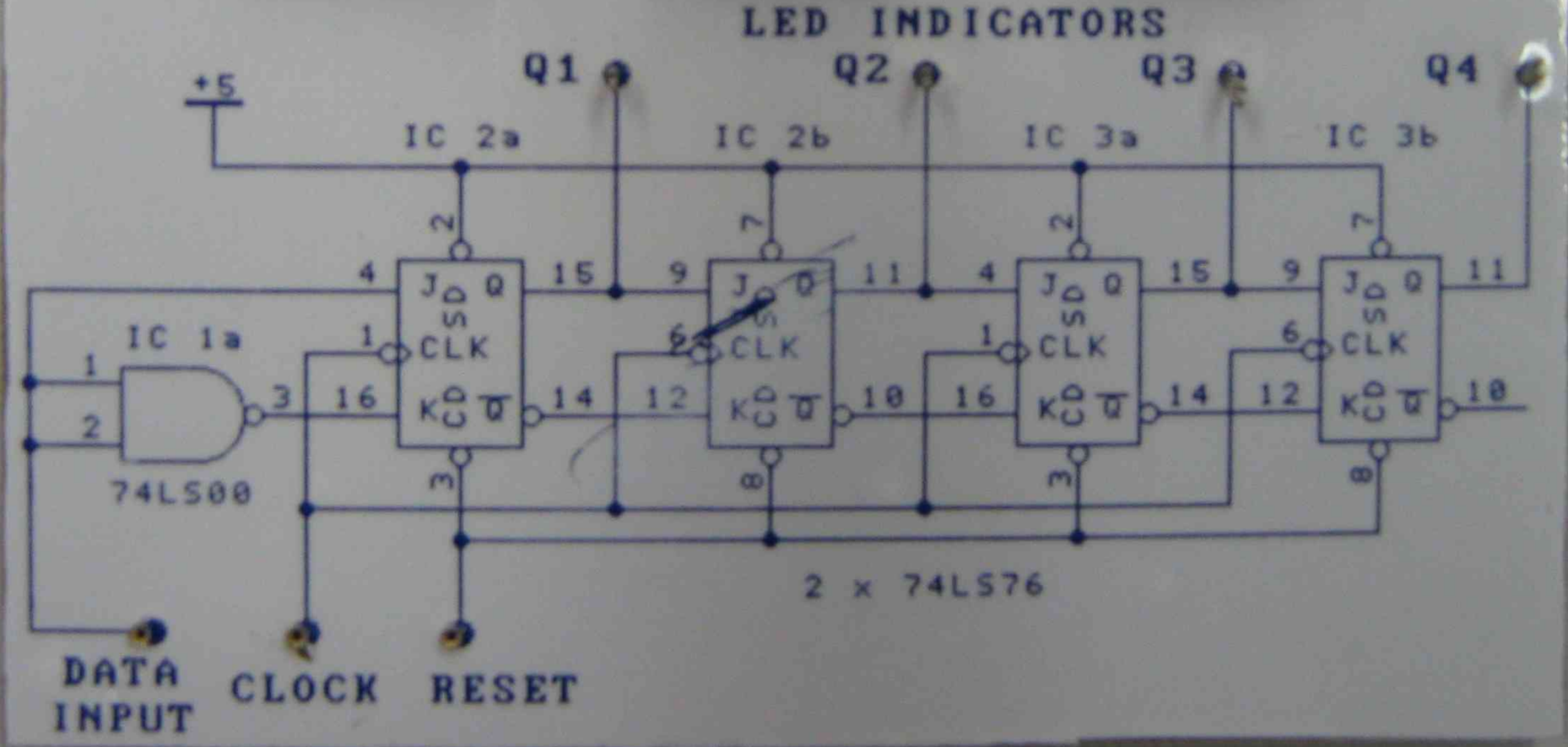
SERIAL TO PARALLEL SHIFT REGISTER

+5V.DC -> PIN 1

COMMON -> PIN



LED INDICATORS

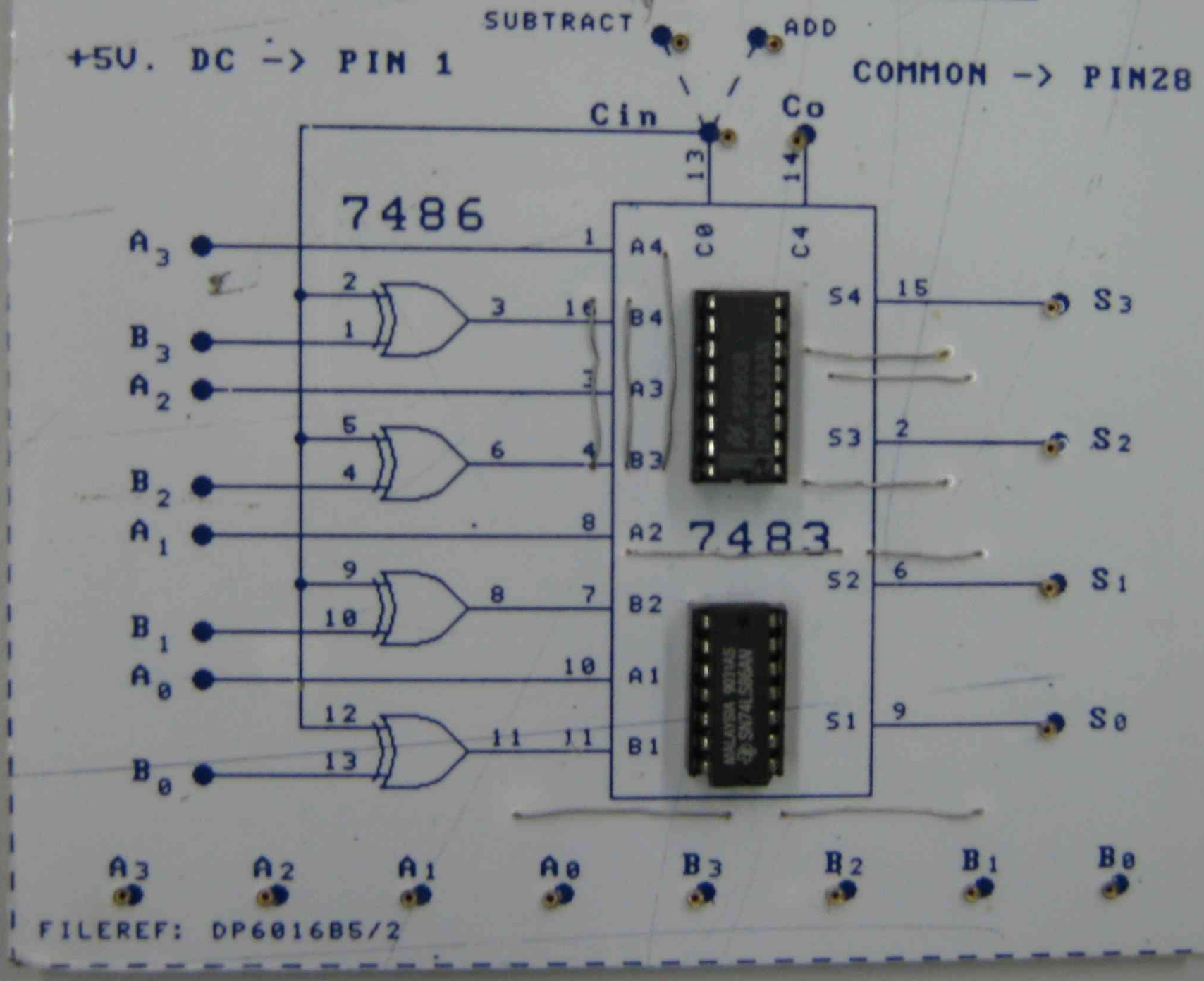


6016B DIGITAL PRINCIPLES
WEEK 14

OK
7/4/94
R.B.M.
G.B.

SYDNEY INSTITUTE OF TECHNOLOGY, ULTIMO
ENGINEERING SERVICES TRAINING DIVISION
ELECTROTECHNOLOGY, INDUSTRIAL ELECTRONICS
DIGITAL PRINCIPLES 6016B WEEK 5 PART 2

ADDER/2's COMPLEMENT SUBTRACTOR



FILEREF: 6016B5-2

OK
8-4-94

A_0 A_1 A_2 A_3

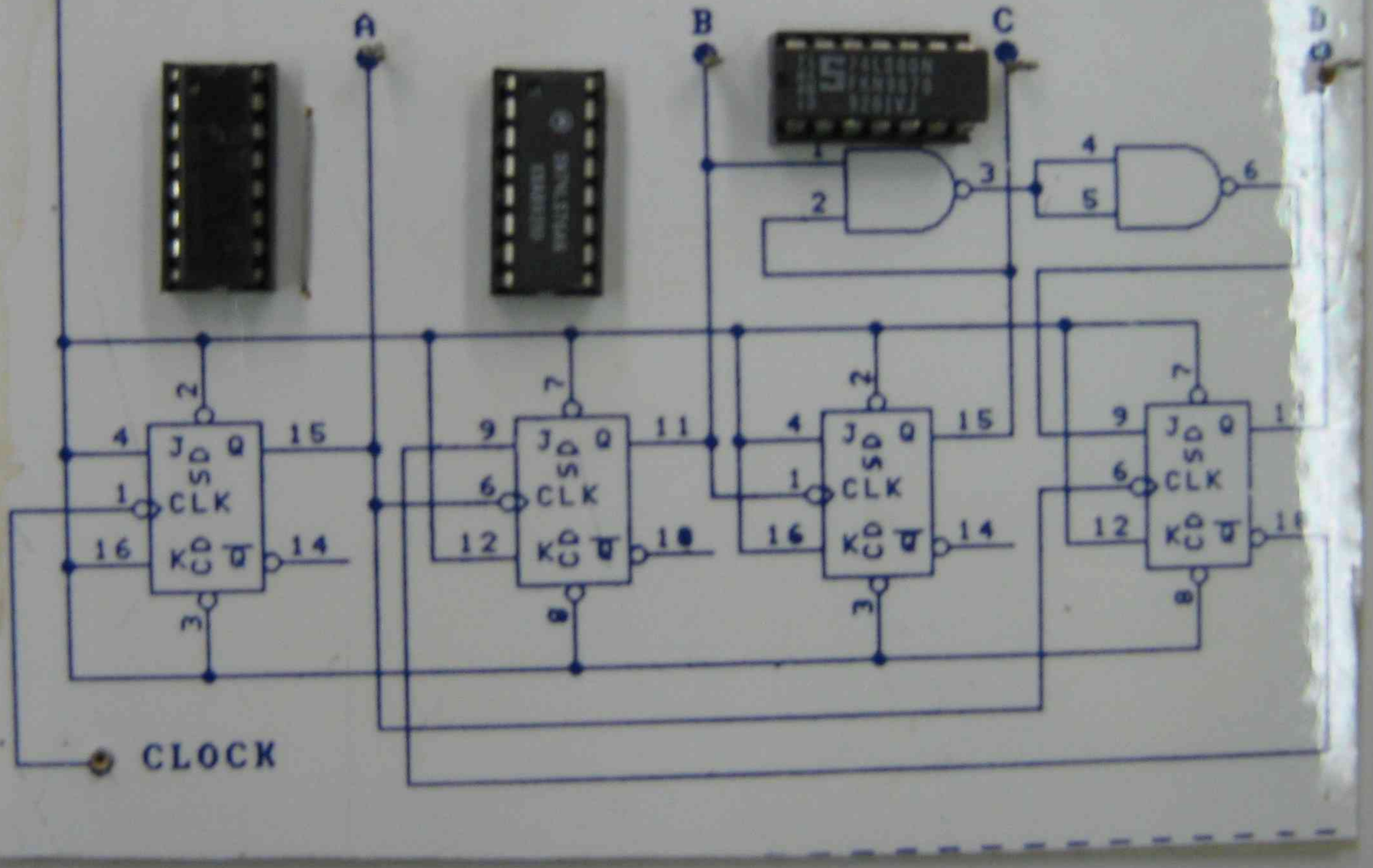
B_0 B_1 B_2 B_3

SYDNEY INSTITUTE OF TECHNOLOGY, ULTIMO
ENGINEERING SERVICES TRAINING DIVISION
ELECTROTECHNOLOGY, INDUSTRIAL ELECTRONICS
DIGITAL PRINCIPLES 6016B WEEK 15 PART 2

BCD (Modulo 10) RIPPLE COUNTER

+5V.DC -> PIN 1

COMMON -> PIN 28

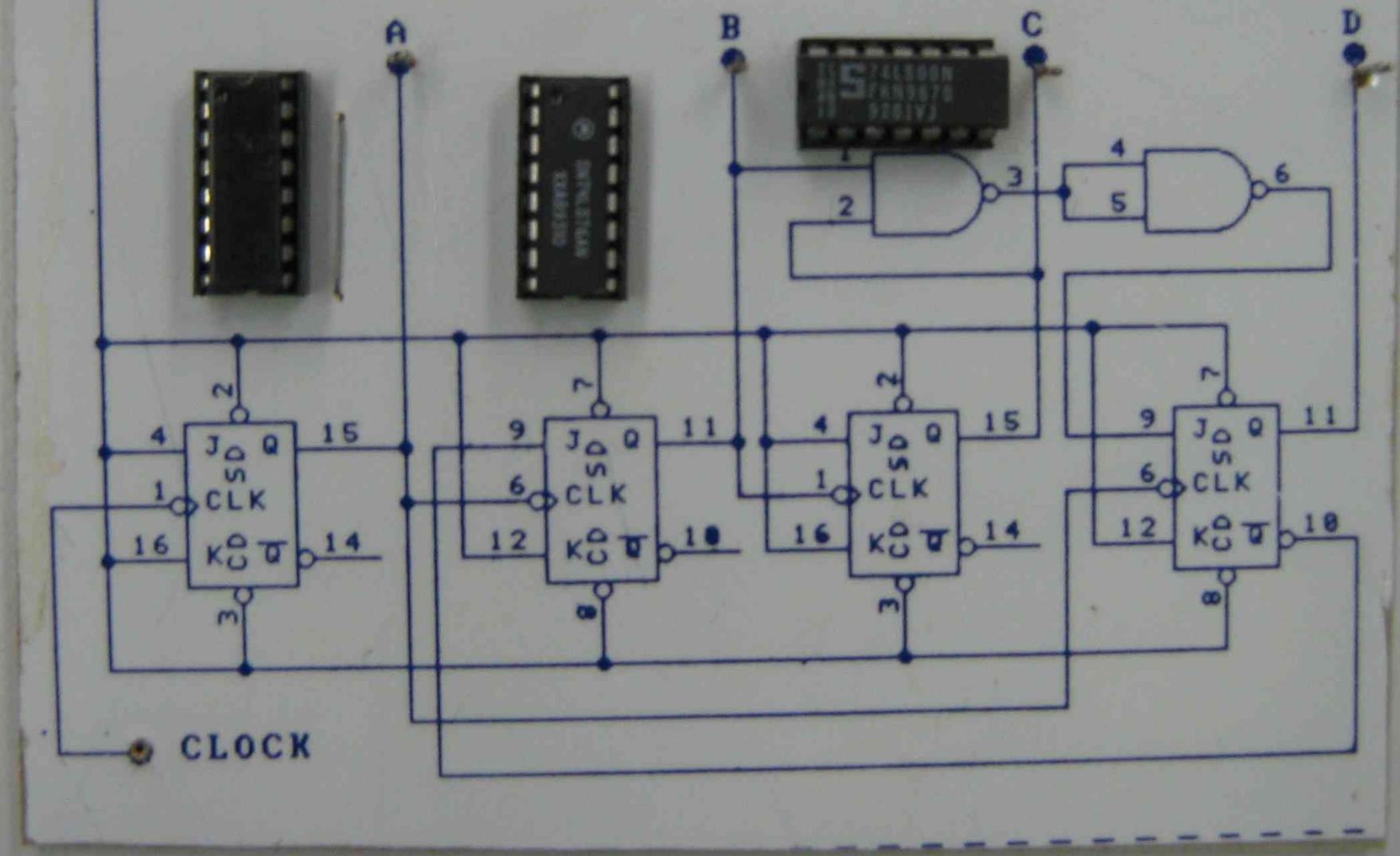


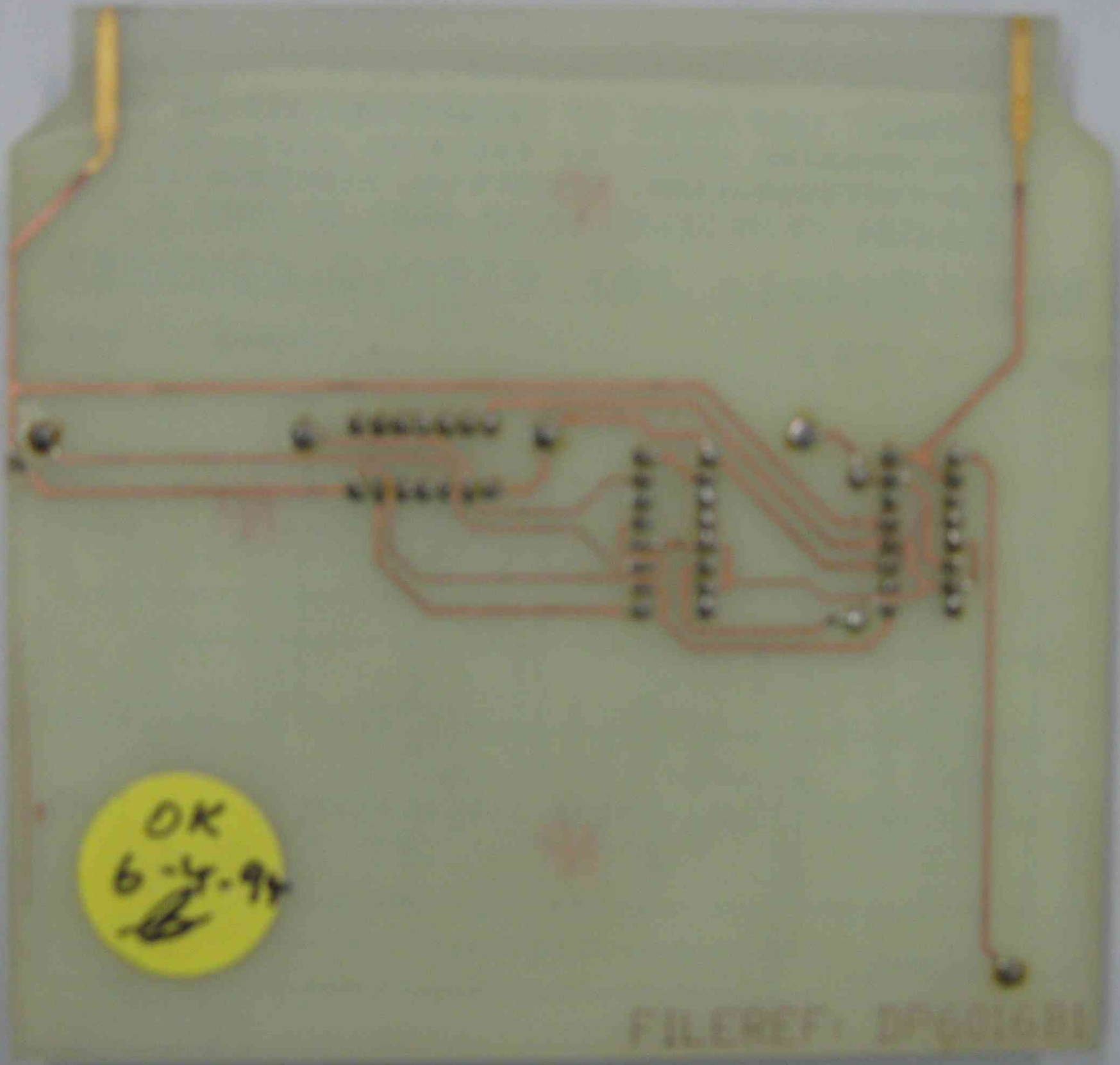
SYDNEY INSTITUTE OF TECHNOLOGY, ULTIMO
ENGINEERING SERVICES TRAINING DIVISION
ELECTROTECHNOLOGY, INDUSTRIAL ELECTRONICS
DIGITAL PRINCIPLES 6016B WEEK 15 PART 2

BCD (Modulo 10) RIPPLE COUNTER

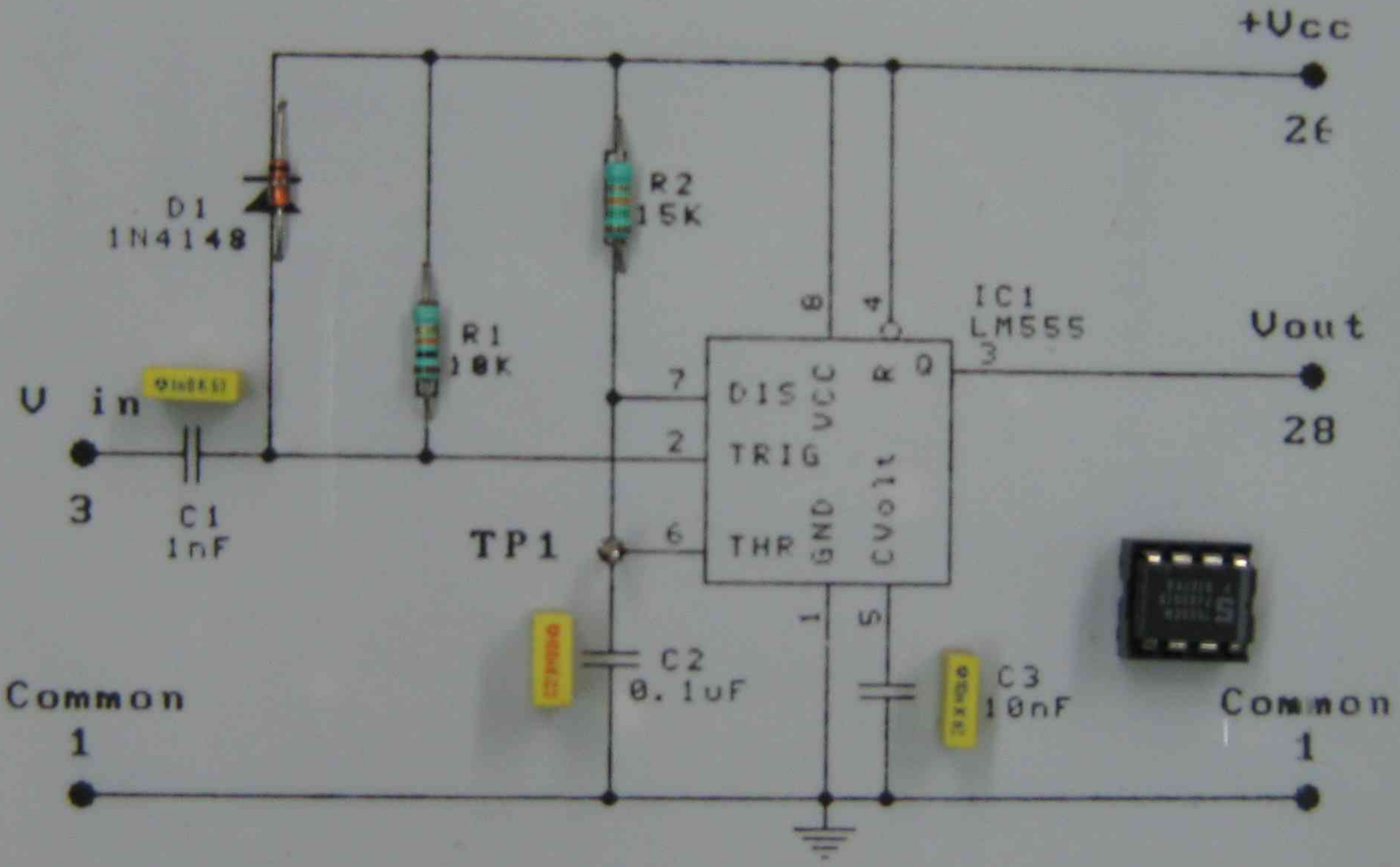
+5V.DC -> PIN 1

COMMON -> PIN28

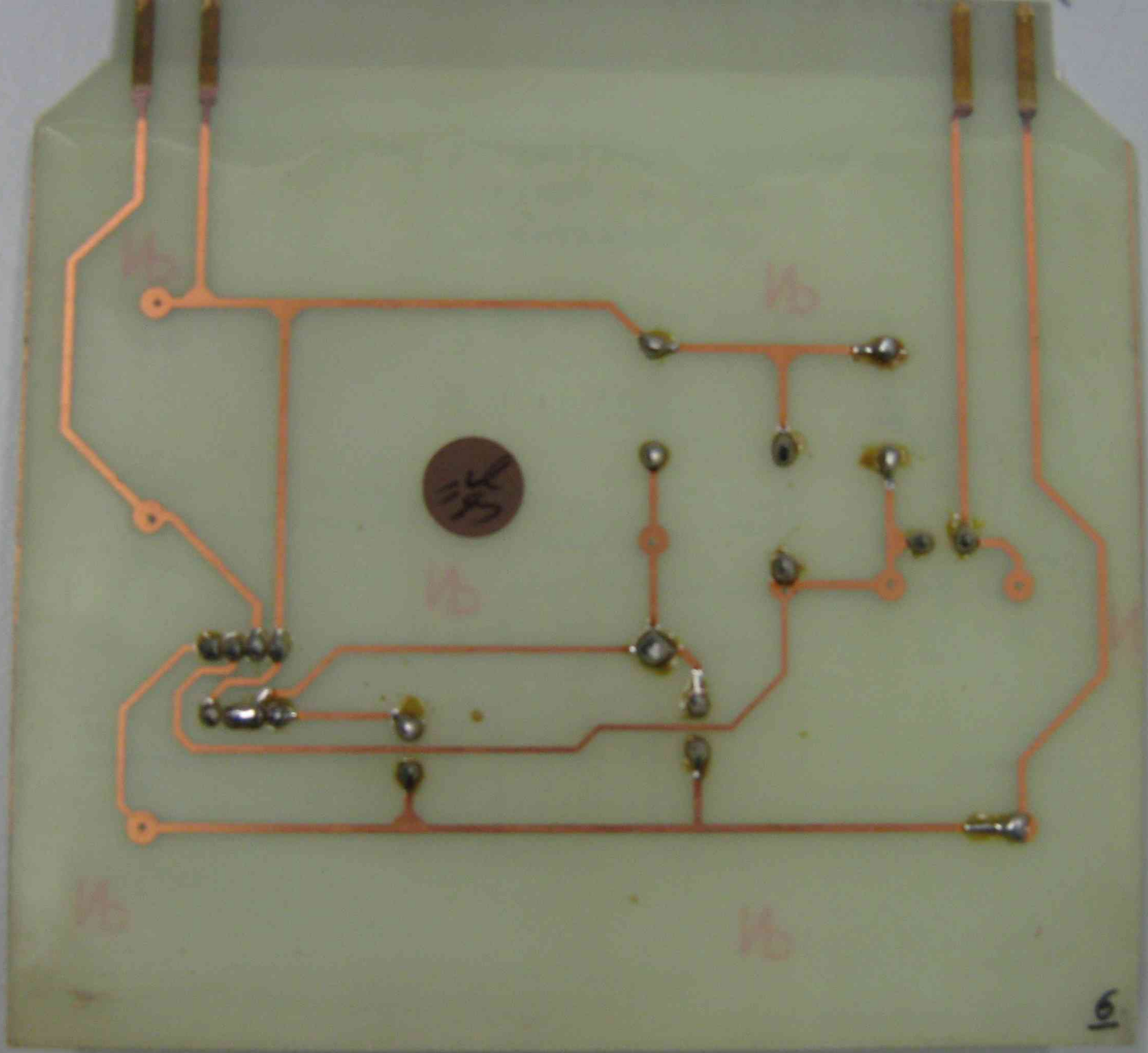



OK
6-4-91

INDUSTRIAL ELECTRONICS 6816C
WEEK 4: TIMERS
555 MONOSTABLE



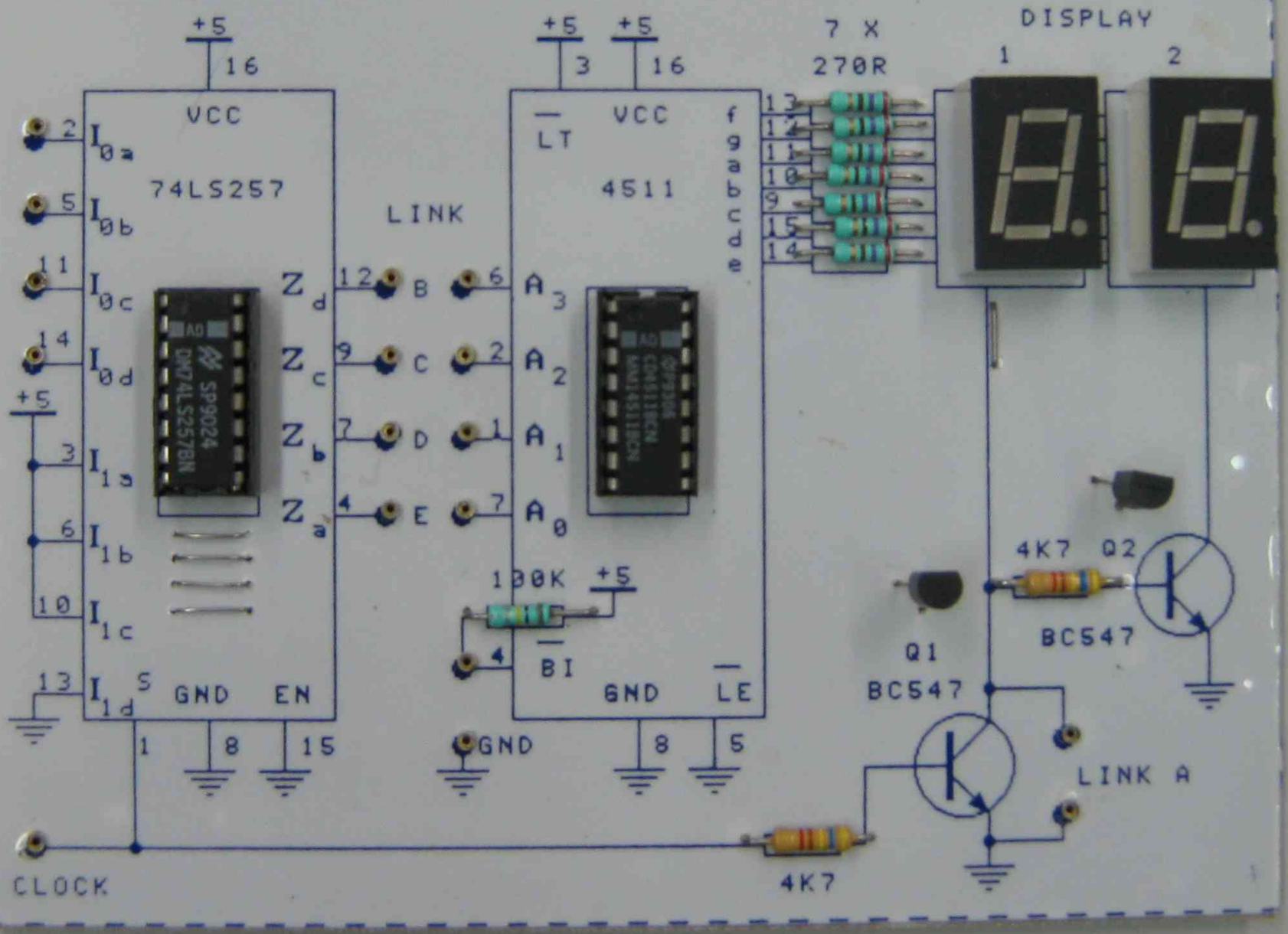
FILEREF: OPAMP555

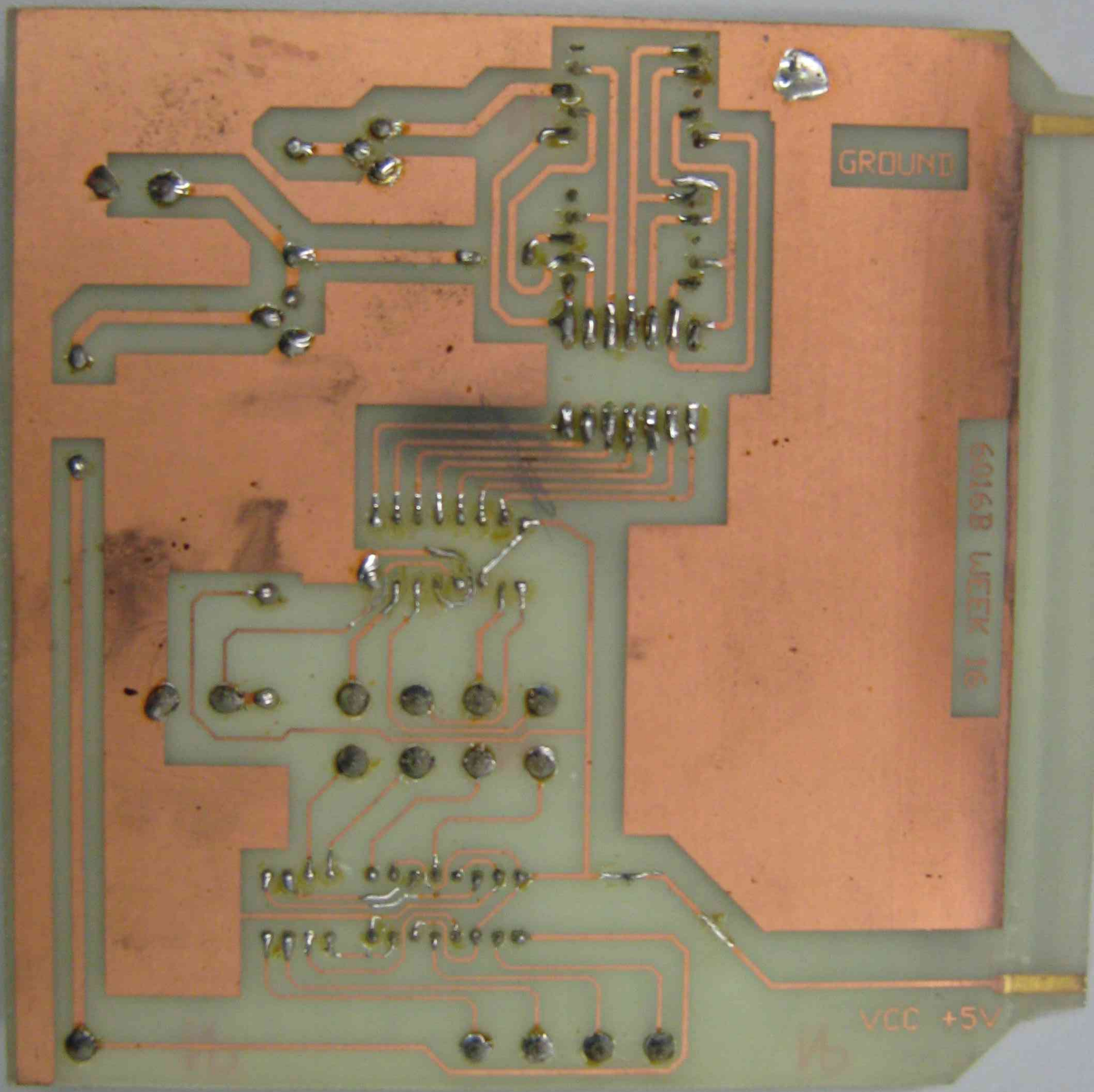

6

SYDNEY INSTITUTE OF TECHNOLOGY, ULTIMO
ENGINEERING SERVICES TRAINING DIVISION
ELECTROTECHNOLOGY, INDUSTRIAL ELECTRONICS
DIGITAL PRINCIPLES 6016B WEEK 16

DISPLAYS AND DRIVERS

1 - VCC +5V
28 - GROUND




GROUND
VCC +5V

OFF ON

POWER

male

female

UC-02 D Sub CONNECTOR

7 6 5 4 3 2 1 0

HI : RED

LO : GREEN

OPEN: NO DISPLAY

4 0
5 1
6 2
7 3

8 BITS LED DISPLAYS

+5V
GND
−5V

Min. Max. Min. Max.

+V −V

+3~+15V
−3~−15V

DC POWER

GND

CMOS
TTL

MODE SELECTOR

TTL/CMOS MODE

OUTPUT

Min. Max.

FREQUENCY

1KHz 10KHz
100Hz 100KHz
10Hz 1MHz

RANGE

Min. Max.

AMPLITUDE

PULSE GENERATOR

A
BCD 2
C
D

A
BCD 1
C
D

D2 D1

DIGITAL DISPLAYS

A
A
B
B
GND

PULSER SWITCHES

SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0

1 0 1 0

8 BITS DATA SWITCHES

+5V
GND

MEMORY
ON
OFF

INPUT

DIGITAL PROBE

240V G.P.O.

240V G.P.

V MAIN
WITCH
2

H.1.16

DIGITAL COUNTER TYPE TSA 6634/2

00000

FREQUENCY
KHz
x100
x10
Hz
x1
x0·1

PERIOD
CYCLES
1
10
100

COUNT
TIME

SENSITIVITY
75mV – 250V
MAX.
POWER
OFF

RESET
ZERO

DISPLAY TIMER
MAX.

INF.
AUTO

COUNT/TIME
GATE INPUT
+V
–V

PERIOD/TIME UNITS
10μS
1mS

TEST

SIGNAL
INPUT

MADE IN ENGLAND

VENNER ELECTRONICS LIMITED

240V G.P.O.

240V G.P.O.

DIGITAL COUNTER TYPE TSA 6634/2

H.1.16

0064

FREQUENCY

KHz

x100

x10

x1

x0·1

Hz

PERIOD CYCLES

1

10

100

COUNT

TIME

SENSITIVITY
75mV – 250V

MAX

POWER OFF

SIGNAL INPUT

RESET ZERO

DISPLAY TIMER

MAX.

INF. — AUTO

COUNT/TIME GATE INPUT

+V

−V

PERIOD/TIME UNITS

10μS

1mS

TEST

MADE IN ENGLAND

VENNER ELECTRONICS LIMITED

240V G.P.O.

240V G.

H.1.16

DIGITAL COUNTER TYPE TSA 6634/2

1378

RESET ZERO

DISPLAY TIMER

INF. AUTO

FREQUENCY

KHz

x100

PERIOD CYCLES

1

10

100

COUNT

TIME

SENSITIVITY 75mV – 250V

MAX.

POWER OFF

SIGNAL INPUT

TEST

MADE IN ENGLAND

VENNER ELECTRONICS LIMITED

DIGITAL COUNTER TYPE TSA 6634/2

H.1.16

316014

FREQUENCY

KHz

×100

×10

Hz

×1

×0·1

PERIOD CYCLES

1

10

100

COUNT

TIME

SENSITIVITY
75mV – 250V

POWER OFF

MAX.

SIGNAL INPUT

RESET ZERO

DISPLAY TIMER

MAX.

INF. — AUTO

COUNT/TIME GATE INPUT

+V

−V

PERIOD/TIME UNITS

10μS

1mS

TEST

VENNER ELECTRONICS LIMITED

MADE IN ENGLAND

DIGITAL COUNTER TYPE TSA 6634/2

H.1.16

0 0 0 0 0

FREQUENCY
KHz
x100
x10
Hz
x1
x0·1

PERIOD
CYCLES
1
10
100
COUNT
TIME

SENSITIVITY
75mV – 250V
MAX
POWER
OFF

RESET
ZERO

DISPLAY TIMER
MAX.
INF.
AUTO

COUNT/TIME
GATE INPUT
+V
–V

PERIOD/TIME UNITS
10μS
1mS

TEST

SIGNAL
INPUT

VENNER ELECTRONICS LIMITED

MADE IN ENGLAND

DIGITAL COUNTER TYPE TSA 6634/2

H.1.16

SENSITIVITY
75mV – 250V

MAX

POWER
OFF

FREQUENCY

KHz

×100

×10

Hz

×1

×0·1

PERIOD
CYCLES

1

10

100

COUNT

TIME

RESET
ZERO

DISPLAY TIMER

MAX.

INF.

AUTO

COUNT/TIME
GATE INPUT

+V

−V

PERIOD/TIME UNITS

10μS

1mS

TEST

SIGNAL
INPUT

VENNER ELECTRONICS LIMITED

MADE IN ENGLAND

DIGITAL COUNTER TYPE TSA 6634/2

H.1.16

49939

FREQUENCY
KHz
×100
×10
Hz
×1
×0·1

PERIOD
CYCLES
1
10
100
COUNT
TIME

SENSITIVITY
75mV – 250V
MAX.
POWER
OFF

RESET
ZERO

DISPLAY TIMER
MAX.
INF.
AUTO

COUNT/TIME
GATE INPUT
+V
–V

PERIOD/TIME UNITS
10μS
1mS

TEST

SIGNAL
INPUT

MADE IN ENGLAND

VENNER ELECTRONICS LIMITED

0 V MAIN
SWITCH
№2

240V G.P.O.

240 V G

H.1.16

DIGITAL COUNTER TYPE TSA 6634/2

SENSITIVITY
75mV – 250V

FREQUENCY

PERIOD
CYCLES

1
10
100

COUNT

POWER
OFF

MAX

RESET
ZERO

DISPLAY TIMER

MAX

INF

AUTO

SIGNAL
INPUT

VENNER ELECTRONICS LIMITED

MADE IN ENGLAND

240V G.P.O.

240 V

H.1.16

DIGITAL COUNTER TYPE TSA 6634/2

9320

FREQUENCY

KHz

×10

PERIOD
CYCLES

1

10

100

COUNT

SENSITIVITY
75mV – 250V

MAX

POWER
OFF

SIGNAL
INPUT

TEST

RESET
ZERO

DISPLAY TIMER

INF.

AUTO

VENNER ELECTRONICS LIMITED

MADE IN ENGLAND

- Put equipment away
- Hang leads neat and tidy

TRIO 15MHz OSCILLOSCOPE CS-1560A II

MINI-LAB MODEL 6038

FUNCTION GENERATOR

LOW VOLTAGE AC/DC PWR SUPPLY

FUNCTION GENERATOR

TRIO 15MHz OSCILLOSCOPE CS-1560A

CH1 or Y

VARIABLE
VOLTS/DIV

MODE

TRIO 15MHz OSCILLOSCOPE CS-1560A II

POWER
SCALE ILLUM
OFF

TRACE ROTATION
INTENSITY

ASTIG

FOCUS

VARIABLE
SWEEP TIME/DIV
mS 2 1 .5 .2 .1 TVV TVH
10 20 50 .1 .2 .5 SEC
μS CAL X-Y

CAL
.1Vp-p

◄► POSIT
PULL X5 MAG

TRIGGERING
LEVEL
PULL AUTO

SYNC
NORM + − TV + −

EXT. TRIG

SOURCE
CH1 CH2 EXT

CH1 orY

POSITION

INPUT
1MΩ ≈22pF

AC GND DC

VARIABLE
VOLTS/DIV
1 .5 .2 .1 2 5 10 20 50 mV CAL

MODE
CH1 CH2 DUAL ADD SUB

VARIABLE
VOLTS/DIV
1 .5 .2 .1 2 5 10 20 50 mV CAL

CH2 orX

AC GND DC

INPUT
1MΩ ≈22pF

POSITION
X-Y ◄►

FUNC

DC OFFSET (OUT) 0V (IN) RAMP FREQ.

VOLTAGE/OPERATIONAL
AMPLIFIER

GAIN
OP AMP

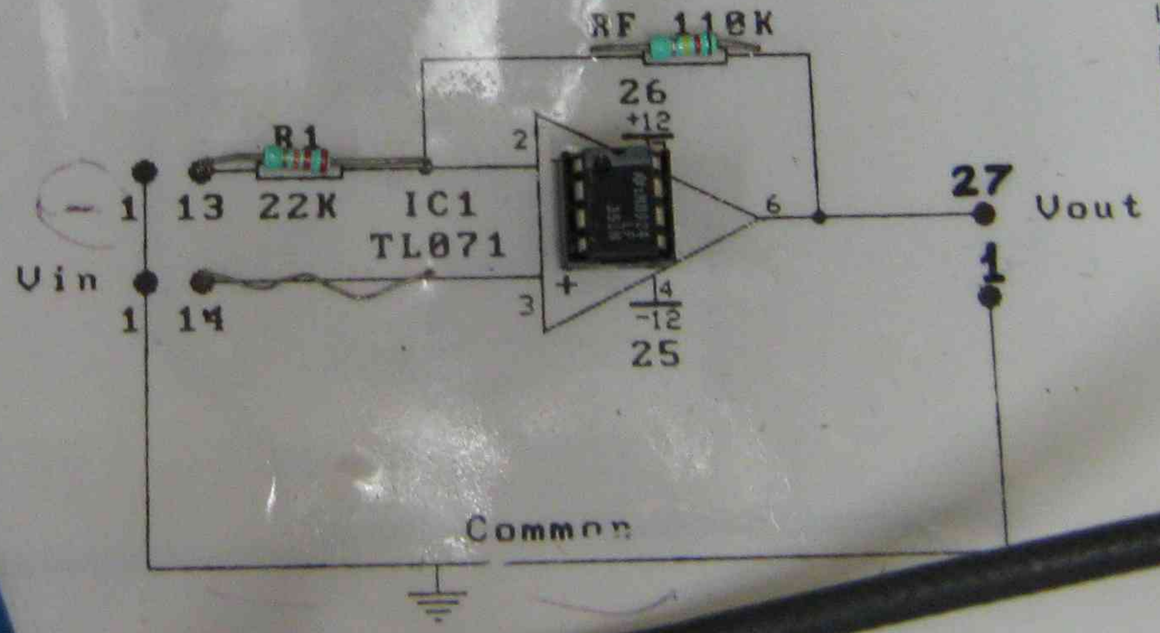
REGULATED POWER SUPPLY

SYDNEY INSTITUTE OF TECHNOLOGY

SYDNEY TECHNICAL COLLEGE

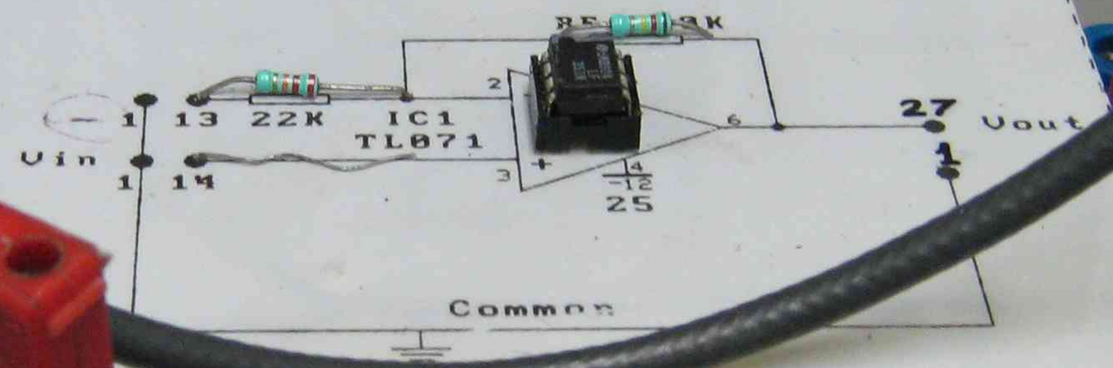
SCHOOL OF APPLIED ELECTRICITY / CAMS / 1992

NON—INVERTING AMPLIFIER



SYDNEY INSTITUTE OF TECHNOLOGY
SYDNEY TECHNICAL COLLEGE
SCHOOL OF APPLIED ELECTRICITY / CAMS / 1992

NON—INVERTING AMPLIFIER



VARIABLE
VOLTS/DIV
CH1 orY
POSITION
INPUT
1MΩ ≈22pF
AC
GND
DC
MODE
CH1
CH2
DUAL
ADD
SUB
VARIABLE
VOLTS/DIV
CH2 orX
POSITION
X-Y
INPUT
1MΩ ≈22pF
AC
GND
DC
mV
CAL

FUNCTION GENERATOR
FREQUENCY
ALL OUT X 200 kHz
X0·1 BOTH IN X0·01 Hz
X1
X10
X100
Hz
X1
X10
X100
kHz
TTL OUTPUT
FM
ON
OFF
AM
RAMP AMPLITUDE
SYMMETRY
SYMM OFF
MIN
MAX
DC OFFSET/RAMP FREQ.
(OUT)
0V
(IN)
−5V
+5V
AMPLITUDE
MIN
MAX
RAMP
0-10V

MINI-LAB
MODEL 603B
bwd INSTRUMENTS

VOLTAGE/OPERATIONAL AMPLIFIER
GAIN
OP AMP
1
100
INPUT
OUTPUT

BI-POLAR POWER SUPPLY
0
−15V
+15V
P.A.
$A_V = 10$
INPUT
± 15V
1 AMP
+5V FIXED

POWER
ON
75
100
125
50
150
25
175
0
200
VOLTS

REGULATED POWER SUPPLY
1 AMP MAX
6·3 VAC
0
6·3 VAC
ISOLATED TO ±200V MAX
1.5-15V 1 AMP
−
0
+
1.5-15V 1 AMP
ISOLATED TO ±200V MAX

TRIO 15MHz OSCILLOSCOPE CS-1560A II

MINI-LAB

MODEL 6038

FUNCTION GENERATOR

VOLTAGE/OPERATIONAL AMPLIFIER

BI-POLAR POWER SUPPLY

REGULATED POWER SUPPLY

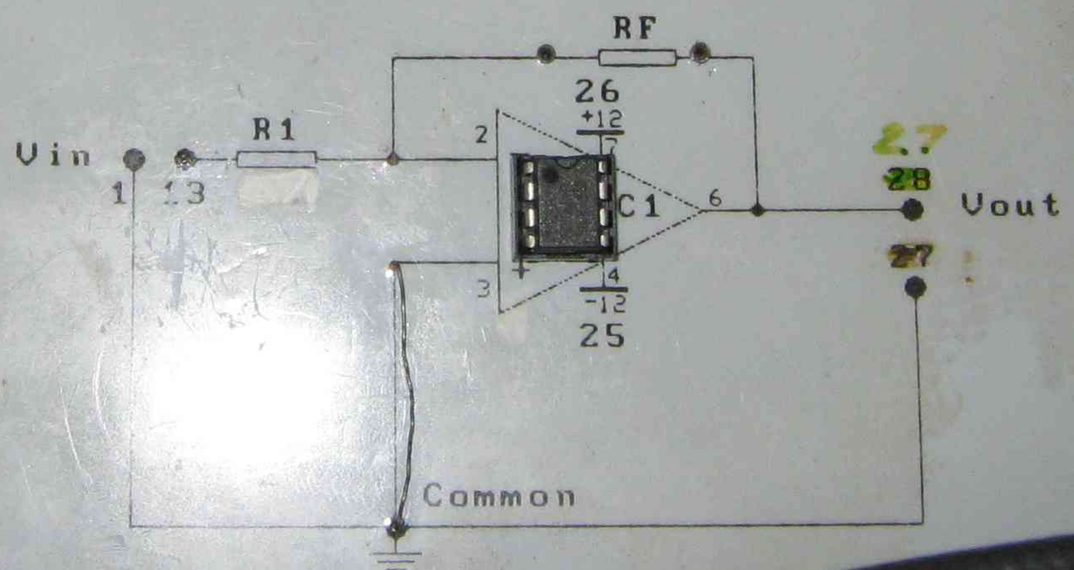
INVERTING AMPLIFIER

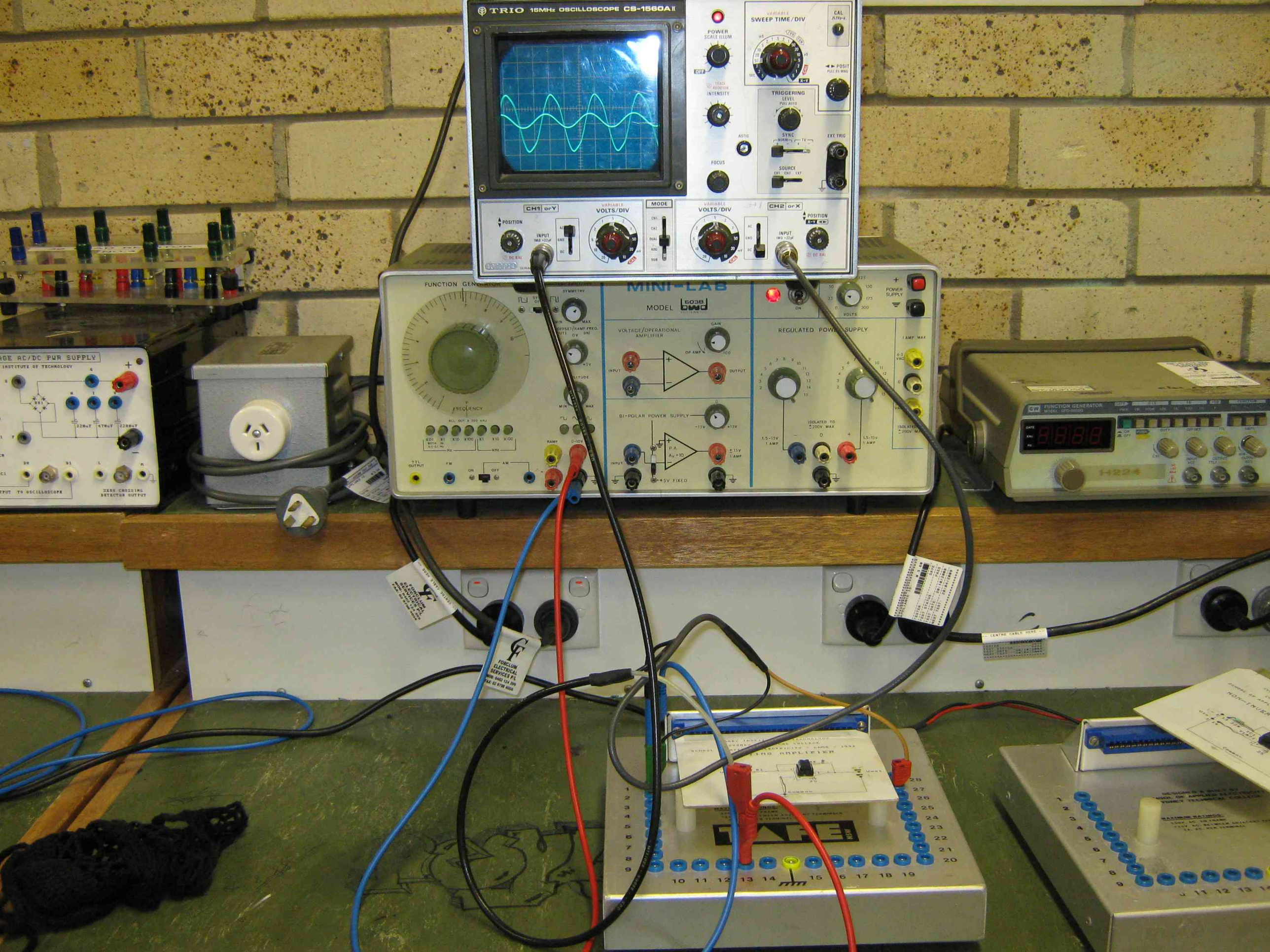
SYDNEY INSTITUTE OF TECHNOLOGY

SYDNEY TECHNICAL COLLEGE

SCHOOL OF APPLIED ELECTRICITY / CAMS / 1992

INVERTING AMPLIFIER




TRIO 15MHz OSCILLOSCOPE CS-1560A II
MINI-LAB
MODEL 6038
FUNCTION GENERATOR
REGULATED POWER SUPPLY
VOLTAGE/OPERATIONAL AMPLIFIER
BI-POLAR POWER SUPPLY
FORCLUN ELECTRICAL SERVICES P/L

CH1 orY

POSITION

INPUT
1MΩ ≈22pF

DC BAL

AC
GND
DC

VARIABLE
VOLTS/DIV

2 1 5 .2 10 .1 20 .05 50 .02 100 .01 200 .005 500 mV
CAL

MODE
CH1
CH2
DUAL
ADD
SUB

VARIABLE
VOLTS/DIV

1 .5 2 .2 5 .1 10 .05 20 .02 50 .01 100 .005 200 mV
CAL

CH2 orX

AC
GND
DC

INPUT
1MΩ ≈22pF

POSITION
X-Y

FUNCTION GENERATOR

FREQUENCY

ALL OUT X 200 kHz

X0·1 BOTH X0·01 Hz
X1 IN X0·01 Hz
X10
X100
Hz
X1
X10
X100
kHz

TTL OUTPUT

FM

AM
ON
OFF

RAMP AMPLITUDE

SYMMETRY

SYMM OFF

MIN MAX

DC OFFSET/RAMP FREQ.
(OUT) 0V (IN)

−5V +5V

AMPLITUDE

MIN MAX

RAMP 0-10V

MINI-LAB

MODEL 603B
INSTRUMENTS

VOLTAGE/OPERATIONAL AMPLIFIER

GAIN

OP AMP 1 100

INPUT

OUTPUT

BI-POLAR POWER SUPPLY

0

−15V +15V

P.A.
$A_V = 10$

INPUT

± 15V
1 AMP

+5V FIXED

ON

VOLTS
0 25 150 175 200

POWER SUPPLY

REGULATED POWER SUPPLY

1 AMP MAX

6·3 VAC

0

6·3 VAC

ISOLATED TO ± 200V MAX

1.5-15V
1 AMP

− 0 +

1.5-15V
1 AMP

ISOLATED TO ± 200V MAX

TRIO 15MHz OSCILLOSCOPE CS-1560A II

POWER
SCALE ILLUM
OFF

TRACE ROTATION
INTENSITY

FOCUS

ASTIG

VARIABLE
SWEEP TIME/DIV
mS 2 1 .5 .2 .1
5 10 20 50
SEC .1 .2 .5
μS
TVV TVH
CAL
X-Y

CAL
0.1Vp-p

◄► POSIT
PULL X5 MAG

TRIGGERING
LEVEL
PULL AUTO

SYNC
NORM + − TV + −

SOURCE
CH1 CH2 EXT

EXT. TRIG

CH1 orY
POSITION
INPUT
1MΩ ≈22pF
AC GND DC
VARIABLE
VOLTS/DIV
1 .5 .2 .1 2 5 10 20 50 mV
CAL

MODE
CH1 CH2 DUAL ADD SUB

VARIABLE
VOLTS/DIV
1 .5 .2 .1 2 5 10 20 50 mV
CAL
AC GND DC

CH2 orX
POSITION
X-Y ◄►
INPUT
1MΩ ≈22pF

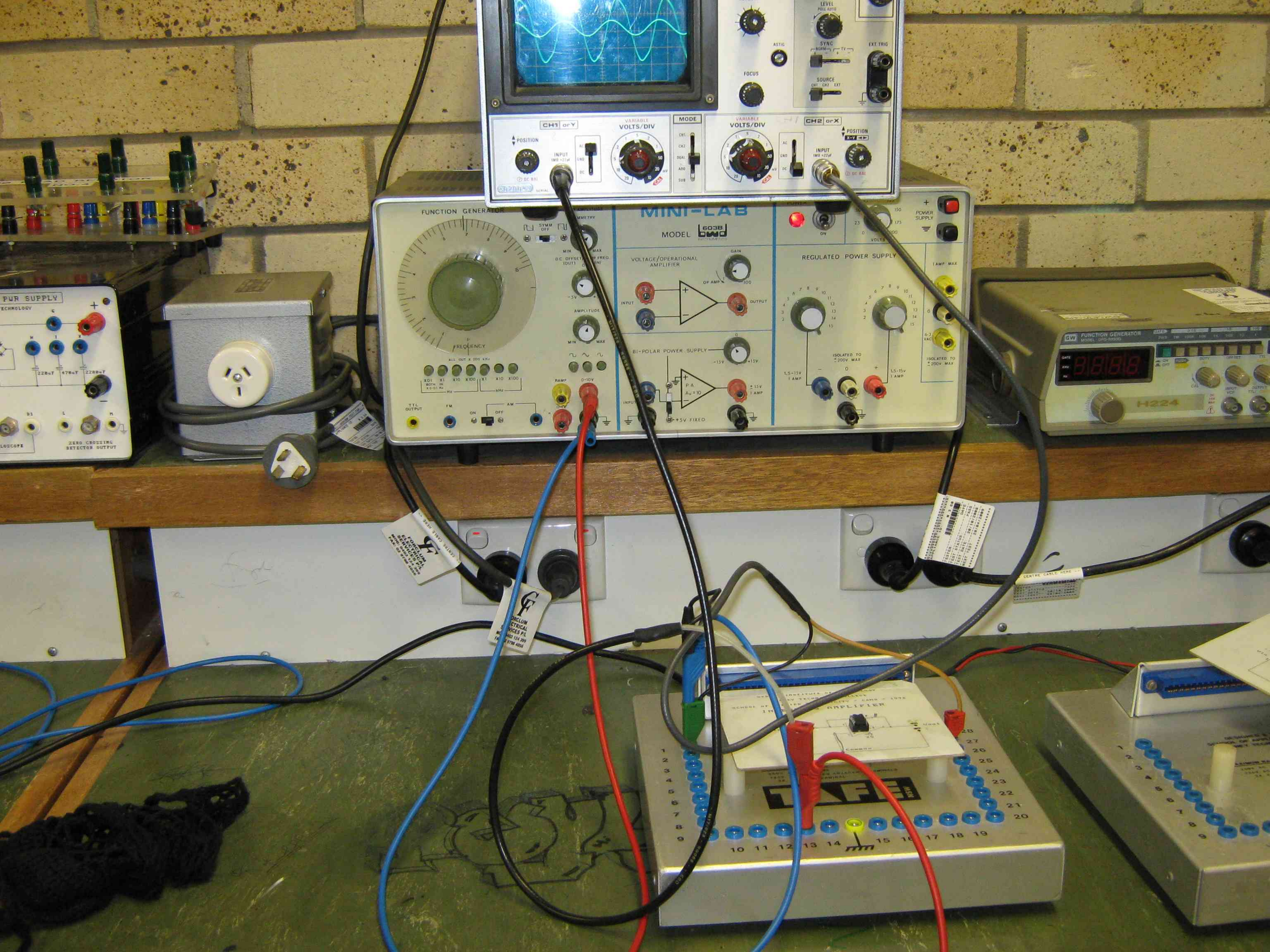
FUNCTI

DC OFFSET (OUT) 0V (IN)
RAMP FREQ.
−5V +5V

VOLTAGE/OPERATIONAL AMPLIFIER
GAIN
OP AMP
OUTPUT

REGULATED POWER SUP

1 AMP MAX
6.3 VAC


CH1 or Y
VARIABLE VOLTS/DIV
MODE
CH2 or X
POSITION
INPUT
FOCUS
ASTIG
SYNC
SOURCE
EXT TRIG
FUNCTION GENERATOR
FREQUENCY
AMPLITUDE
TTL OUTPUT
MINI-LAB
MODEL 603B
VOLTAGE/OPERATIONAL AMPLIFIER
BI-POLAR POWER SUPPLY
REGULATED POWER SUPPLY
POWER SUPPLY
PWR SUPPLY
ZERO CROSSING DETECTOR OUTPUT
H224

Microprocessor

Week ①

Integrated circuit

Small scale Integration (S.S.I) - A few logic gates.

Medium scale Integrated circuit (MSI) - A complete integrated circuit counter

Large scale Integrated circuit (LSI) - more than 10,000 individual transistors on ~~silicon~~ single silicon chip.

Mode of operation

Sequence of operation → computer can perform a number of basic operations called machine instructions. which the user selects and orders in a way which solves a particular problem. This sequential list of operation is referred to as a program

A digital computer utilises the very high speed of execution of each machine instruction usually a few micro seconds by having the required sequence ~~code~~ of instructions (or) Program stored within the computer itself. This is known as the stored Program concept and is the fundamental difference between a basic calculator and computer system.

Input Temperature Output Turn on heating element.

Decimal system

10^4	10^3	10^2	10^1	10^0	
4	9	5	3	6	= 49536

$4\times10^4 + 9\times10^3 + 5\times10^2 + 3\times10^1 + 6\times10^0 = 49536$

Binary system

2^4	2^3	2^2	2^1	2^0	
1	0	1	1	1	$= 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$
					$= 23$

Hex decimal number

4 bit binary pattern	Hexa decimal Symbol
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F

0110 1101 = 6D (Hex)
6 D

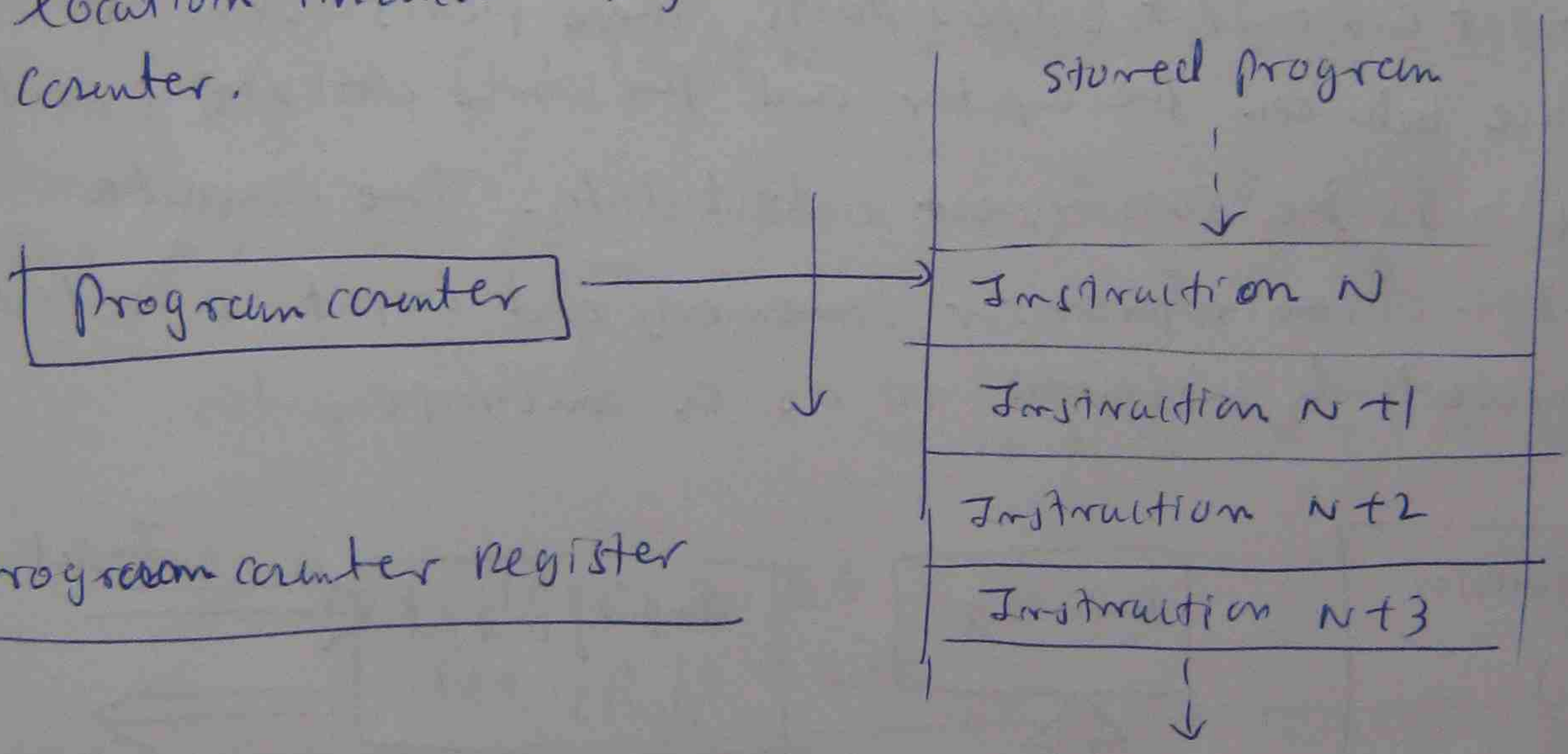
1111 0010 = F2 (Hex)
F 2

1011 0100 1000 1110 = B48E (Hex)
B 4 8 E

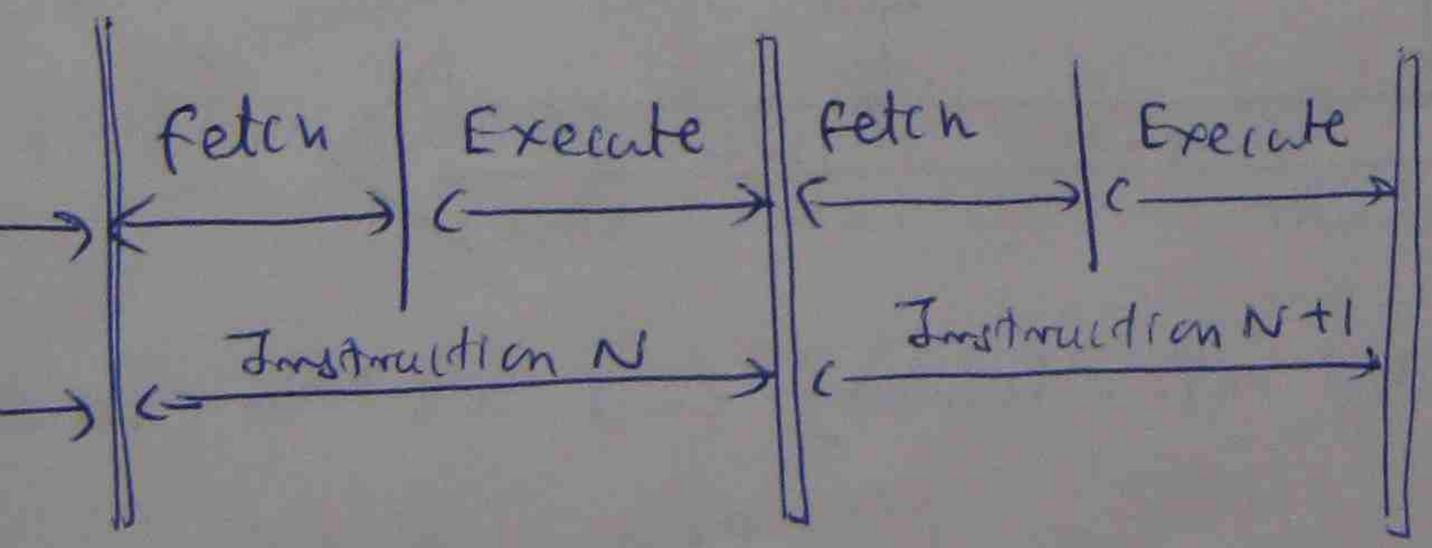
Program counter

In order to remember which program instruction is to be executed next, the microprocessor contains a register (or temporary information storage location) called the Program counter (PC), the contents of which points to the next sequential instruction to be fetched and executed.

Thus during a typical instruction cycle, the next instruction to be executed is read from the memory location indicated by the contents of the Program counter.



Program counter Register



Problem

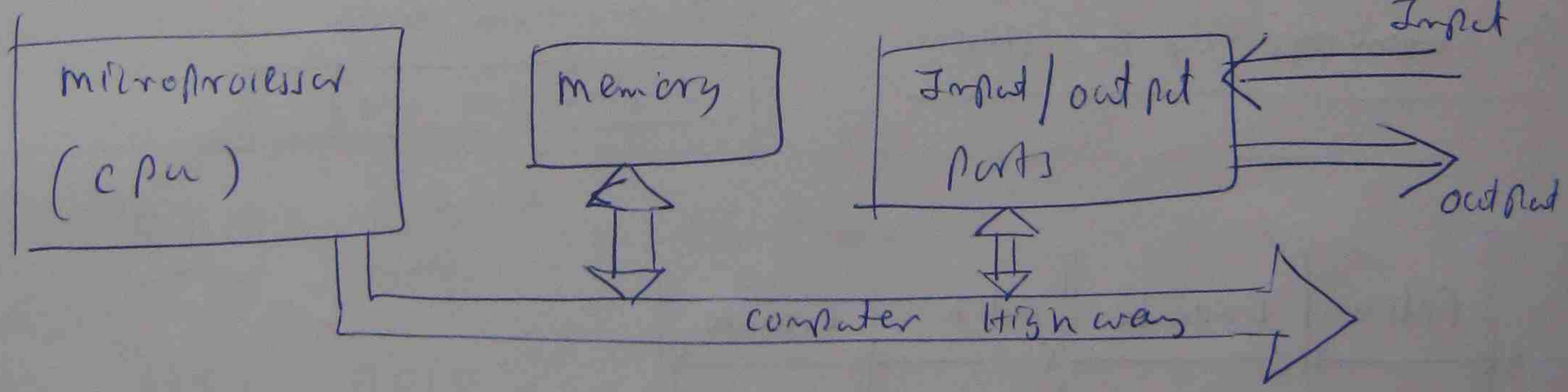
(1) Convert the following decimal numbers into their equivalent binary numbers.

(a) 35, (b) 67 (c) 224

Basic structure and operation

A digital computer executes a list of basic machine instructions (the program) which have been selected and ordered by the user to solve a particular task.
In order to exploit ~~the~~ the intrinsic high speed of execution of each machine instruction, the program is stored within the computer.

A basic digital computer is comprised of a memory which is primarily used to hold (or) store the program, and some input and output (I/O) ports. These ports form the interface between the computer and the source of input data and the subsequent output data. The complete ~~output~~ combination of microprocessor, memory and input & output ports is collectively referred to as a microcomputer.



→ complete program is loaded into memory and is then executed.

microprocessor
- → Phase(1)
  The next instruction is fetched from memory (Fetch cycle)
- → Phase(2)
  The next instruction is fetched from memory; then, in the second phase (or) execution cycle. The microprocessor executes (or performs) the action specified by the instruction

Pb①

35

2	35	1
2	17	1
2	8	0
2	4	0
2	2	0
	1	

Exercise

1000 11 (9)

To make 8 bit put 0, 0

00 1000 11

67 = 01000011

224 = 11100000

Pb② Convert the following binary numbers into their equivalent decimal numbers.

(a) 10 111 (b) 10 10101 (c) 110 110 11

(a) 10 111 = 000 10 111 = $0\times2^7 + 0\times2^6 + 0\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

= 16 + 4 + 2 + 1 = 23

Similarly 10 10101 = 85

110 110 11 = 219

Pb③ convert the following decimal numbers into their equivalent hexadecimal number.

(a) 27 (b) 96 (c) 3334

27 =

2	27	1
2	13	1
2	6	0
2	3	1
	1	

110 11 ⇒ 000 | 10 11

= 1B

(b) 96 =>

2	96 – 0
2	48 – 0
2	24 – 0
2	12 – 0
2	6 – 0
2	3 – 1
	1

(6)

1100000 = 0110, 0000

6 0

60 (Hex)

(c) 3334 => 110, 100000110 = D06 (Hex)

D 0 6

Pb(4) convert the following hexa decimal numbers into their equivalent decimal numbers.

(a) D3 (b) 2F (c) 2D9E

(a) D3 = 1101, 0011 = $1\times2^7+1\times2^6+0\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0 = 211$

(b) 2F = 0010, 1111 = $0\times2^7+0\times2^6+1\times2^5+0\times2^4+1\times2^3+1\times2^2+1\times2^1+1\times2^0 = 47$

(c) 2D9E = 0010, 1101, 1001, 1110

$= 0\times2^{15}+0\times2^{14}+1\times2^{13}+0\times2^{12}+1\times2^{11}+1\times2^{10}+0\times2^9+1\times2^8+1\times2^7+0\times2^6+0\times2^5+1\times2^4+1\times2^3+1\times2^2+1\times2^1+0\times2^0$

$= 11678$

D027

Microprocessor

Week (2)

Microcomputer Architecture

Microprocessor (CPU)

Memory → Hold the stored program.

Input & output ports → Interface the micro computer to the various input & output devices controlled by it.

Microprocessor

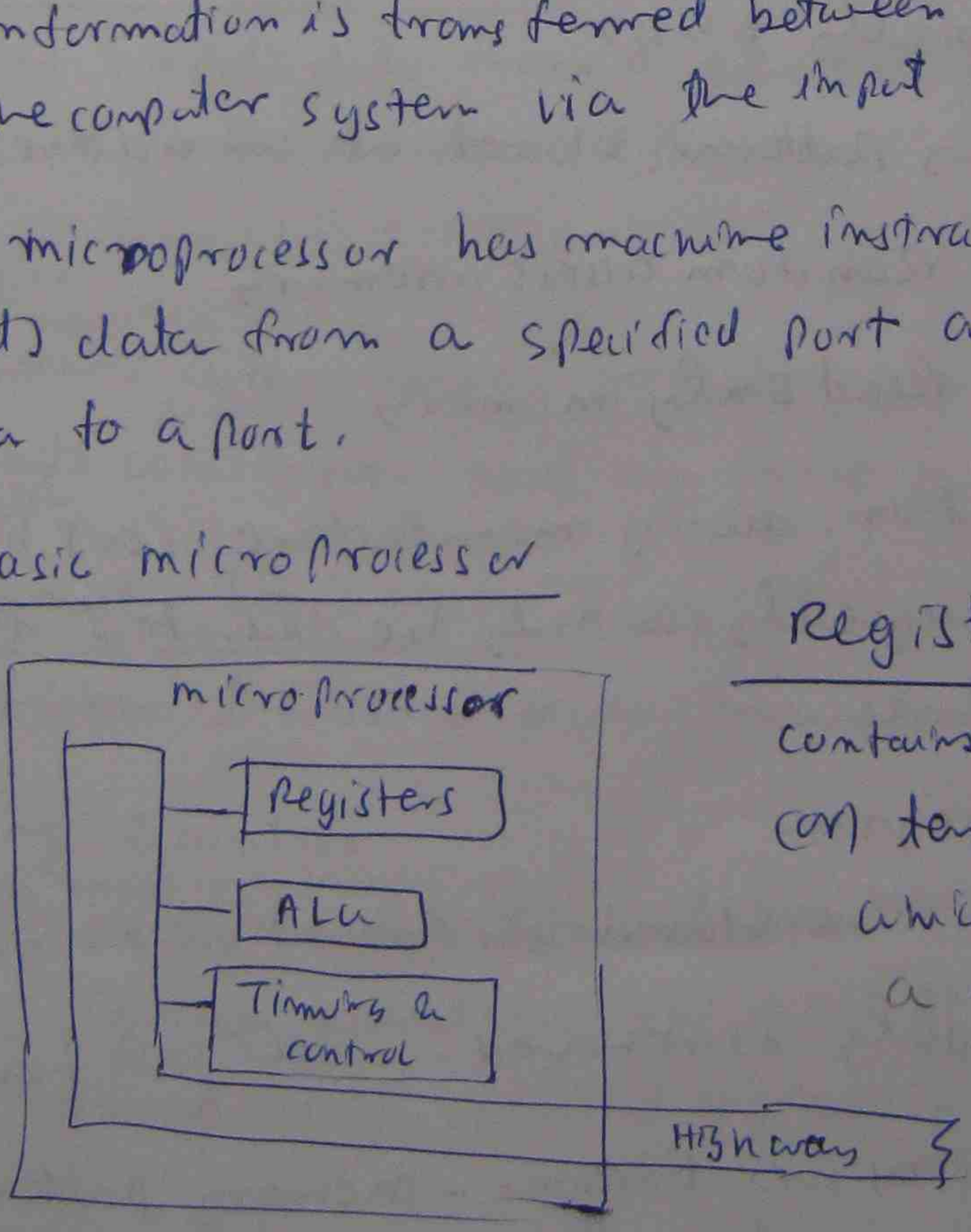
Individual data byte manipulation instructions (add, subtract)

memory transfer instructions (read data byte from memory
write data byte to memory)

Information is transferred between external devices and the computer system via the input & output ports.

The microprocessor has machine instructions to both read (input) data from a specified port and to write (output) data to a port.

Basic microprocessor



Register

contains a number of registers (or) temporary storage elements which can hold or store a single byte or word

(ALU) Arithmetic Logic Unit - Performs the actual data manipulation.

Timing & control section - co-ordinates the internal operation of the ALU and registers so that the desired action specified by an instruction is performed.

The microprocessor communicates with the memory, both to obtain the individual instructions which make up the program and to access & store data and to transfer data to and from input and output using a highway (or) bus.

Memory

Locations ⟷ address. - Each location contains a binary pattern with a number of bits corresponding to the word length of the computer (typically 8 bits).

Content - The binary pattern stored at an address.

Memory → RAm - Random access memory
→ Rom - Read only memory.

Rom - Fixed information during manufacture (or) by the user and consequently can only be operated in a read only mode.
- Hold fixed program
- Non volatile. The stored information is not lost when power supply is removed. low cost / and

Erasable Programmable Roms (or) EPRom - memory pattern can be changed by the user in a controlled manner.

Week 4 (2)

Microcomputer Architecture

~~#5~~ (7)

Functional Units of a microcomputer

microprocessor — cpu
The memory — It is used to hold the stored program.
Input/output Ports — To interface the micro computer to the various input and output devices controlled by it.

Microprocessor

- Execute a number of basic machine instructions.
- Data byte manipulation, instruction (add, subtract)
- memory transfer instruction (Read data byte, transfer between external devices and the computer system via the input and output ports,)
- machine instructions

  Read (input) data from a specified port
  write (output) data to a port.

Register

- contains a number of registers or temporary storage elements which can hold or store a single byte or word.

Arithmetic Logic Unit (ALU)

- Perform the actual data manipulation operations.

Timing & control

- co-ordinate the internal operation of the microprocessor and controls operation of the ALU and registers so that the desired action specified by an instruction is performed.

(16) (2)

Microprocessor communication

- The microprocessor communicates with the memory both to obtain the individual instructions which make up the program and to access and store data.
- To transfer data to and from input and output ports using a highway (or) bus.

Erase

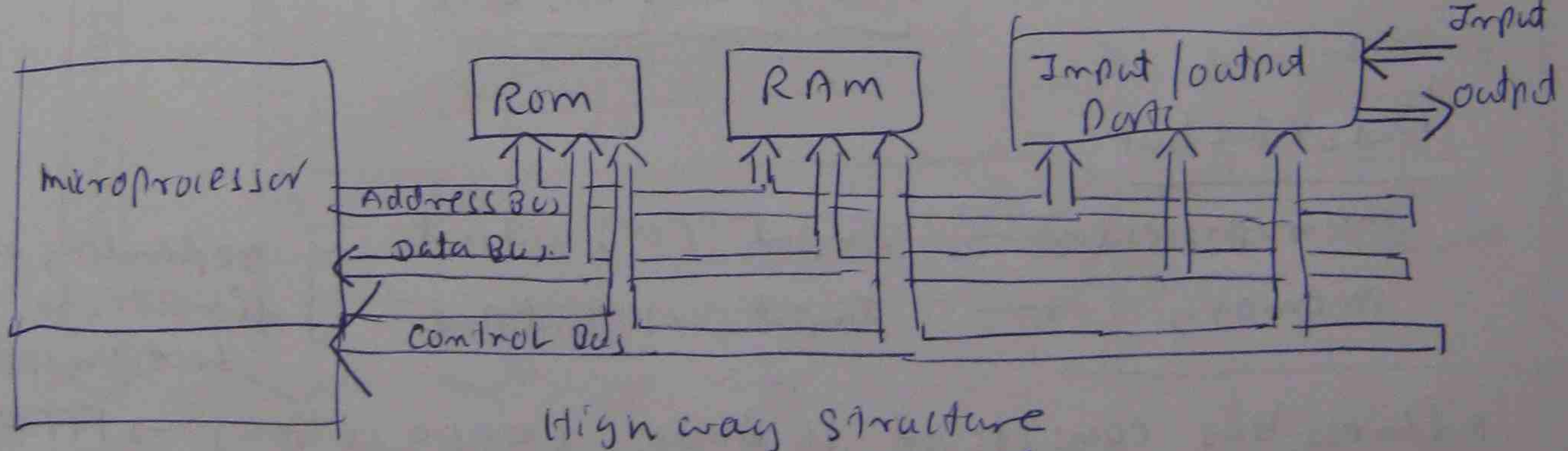
- Exposure to intensive altra violet light
- Applying voltage to ~~chip~~ specific pins on the integrated circuit.

Highway Structure

Data bus - carry the data associated with a memory or input/output transfer & typically 8 bit wide

Address bus - To specify the memory location (or) Input output port involved in a transfer

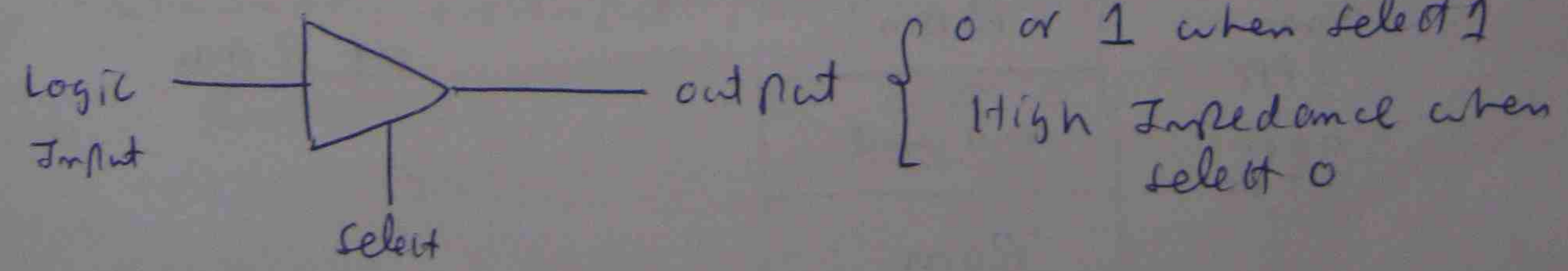
control bus - made up of the various control lines generated by the microprocessor and other system components to synchronise transfer.



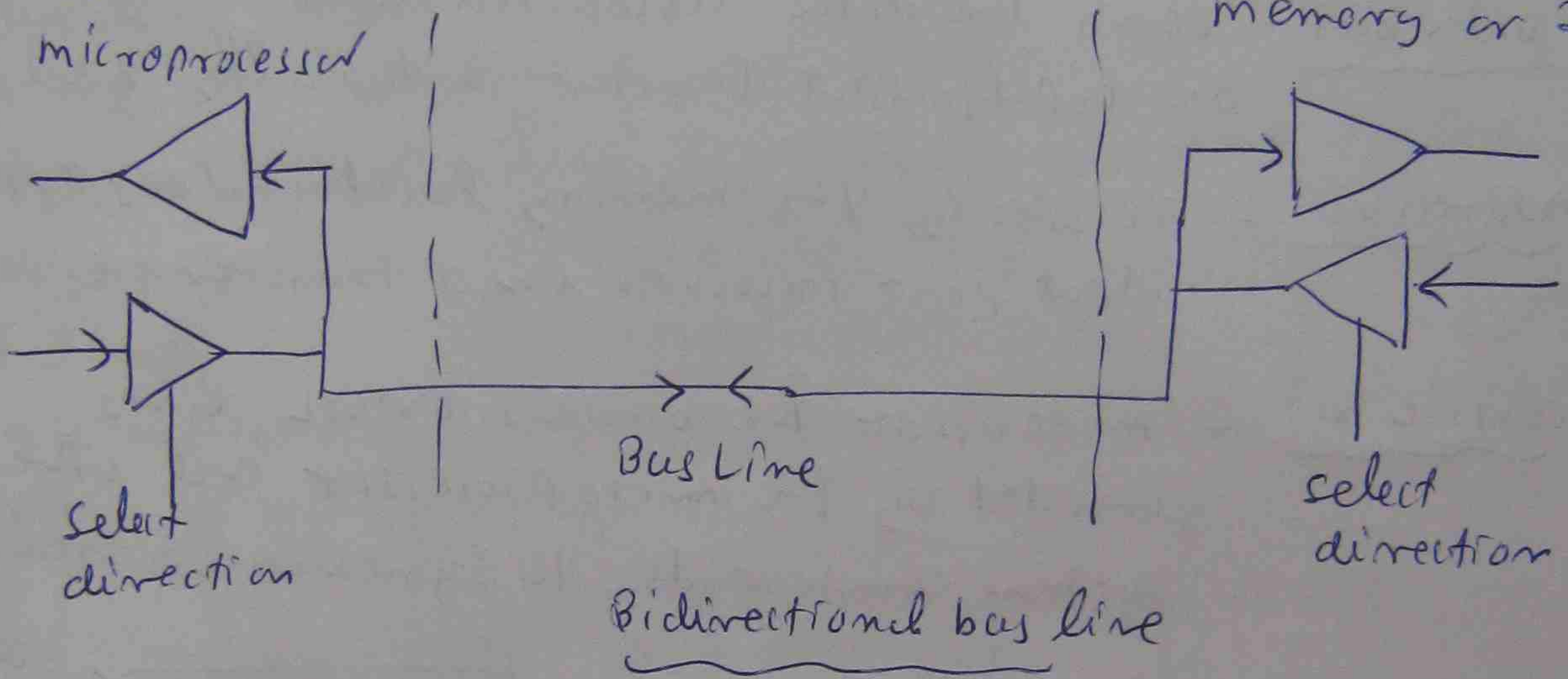
Highway Structure

Data bus - Bi directional data flow

The processor can write data on to the bus lines to be read by a memory device (or) it can read data from the bus presented by such a device.



It becomes possible to make a single pin a logic input and output by incorporating within the microprocessor logic output gates, a third output state in addition to normal 0 and 1 signals. This third state is a high impedance condition where the output is effectively switched off.

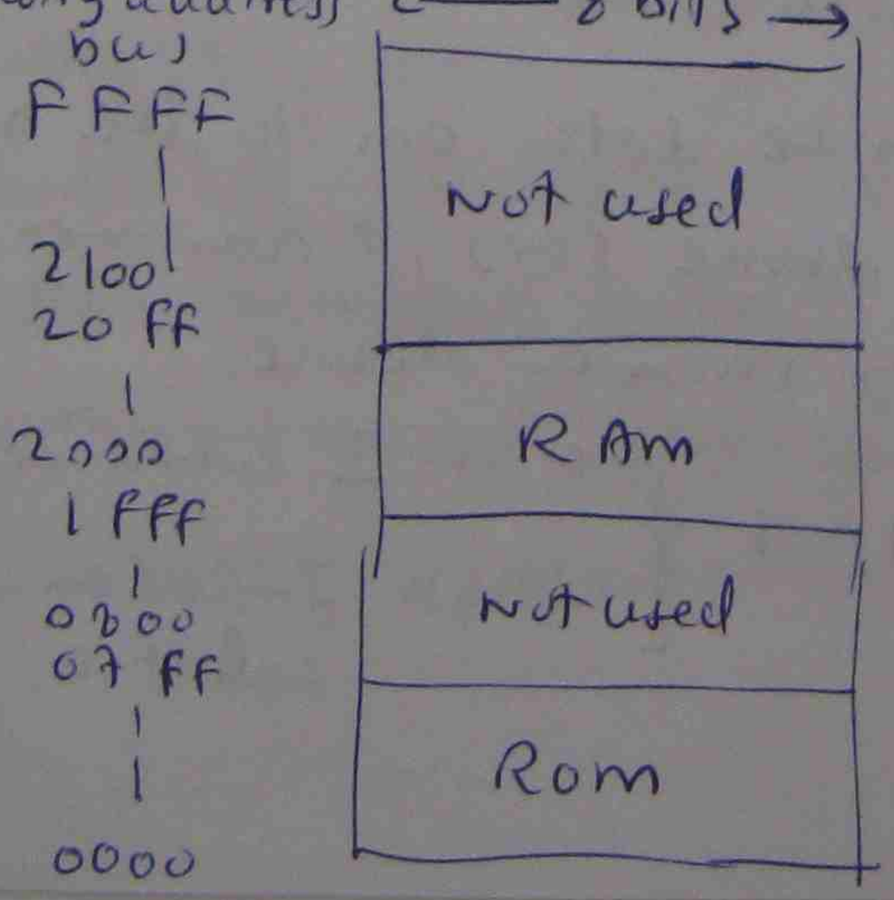


End of the bus

microprocessor — Input (or) output
memory — Input (or) output
} depending on direction control

Address bus consists of 16 lines. (0000)(Hex) → FFFF (Hex)

memory map



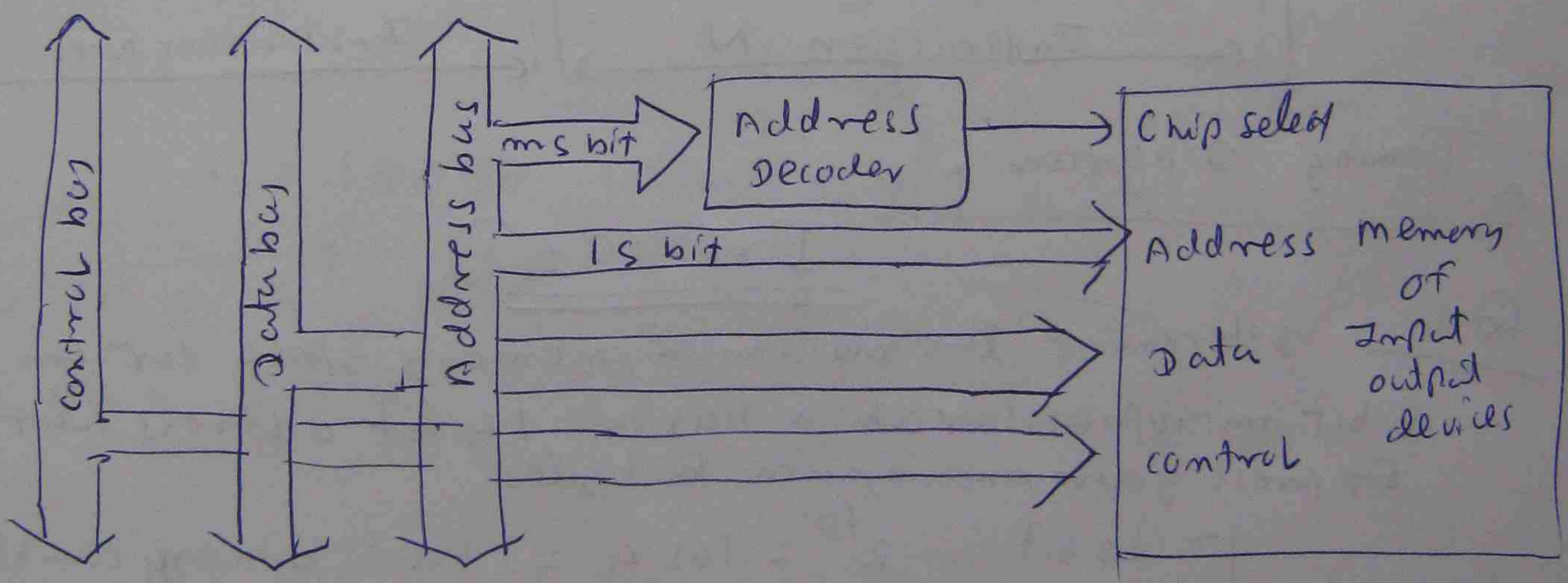
2K => 0000 → 07FF bytes of Rom

256 (2000 → 20FF) bytes of RAM

Address decoding

Since there are a number of devices connected to the computer highways - Rom and Ram chips, input/output devices etc - it is necessary to ensure that only the device intended for the data transfer responds when a request is made by the microprocessor.

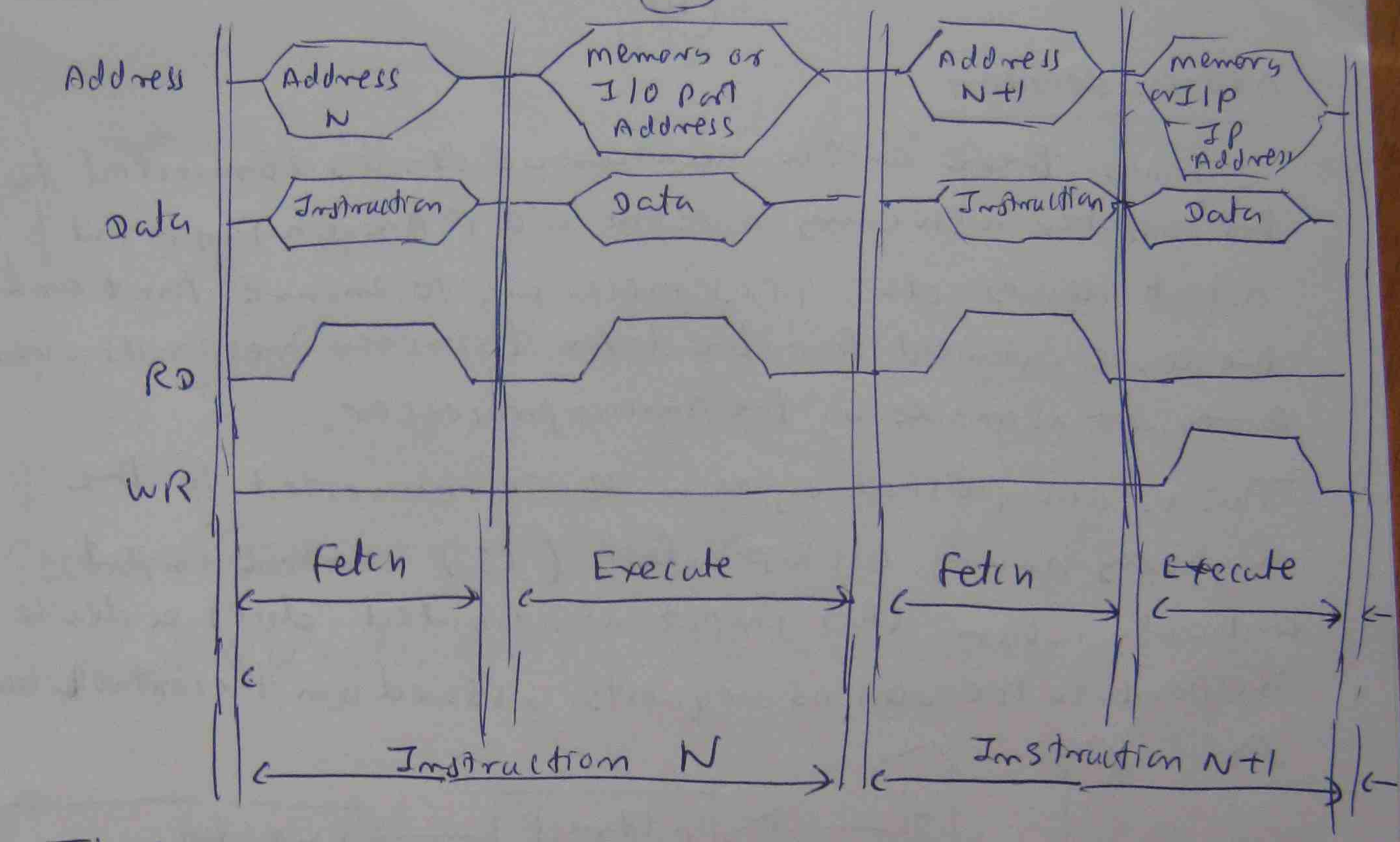
This is accomplished by each device connected to the highway having a chip select (CS) control input, and only when this input is activated does a device respond to the various requests issued on the control bus



Bus control

The control bus incorporates the timing signals which are generated by the microprocessor to synchronise information transfers between the microprocessor & a memory or input/output port

Read - RD, write WR



Timing Diagram

EXERCISE (2)

Q1 Determine the maximum memory space for an 8 bit microprocessor which has a 14 bit address word Express your answer in K bytes.

14 bit → 10 bit = 2^{10} = 1024 = 1K binary combinations
or,
4 bit = 2^4 = 16 binary combinations
(0000 → 1111)

2 | 16
2 | 8
4

∴ 14 bit = 1K × 16 = 16K bytes of memory

Q2 A microcomputer system requires 4K bytes of Rom and 256 bytes of RAM. Determine the start and end addresses of each memory block if the two memories are to occupy continuous blocks of memory starting at address (0000) hex. Express your answer in hex notation.

1K = 2^{10} = 1024 locations. 10 bit

4K = 2^{12} = 4096 locations 12 bit

256 = 2^8 locations 8 bit

Rom = $2^{10} \simeq$ 1000 ~~= 0000 ⟶ 1000 Rom~~

1000 × 4 = 4 Kbytes Rom = 0000 ← Zero byte

↓

1000

∴ Rom 0000 ↓

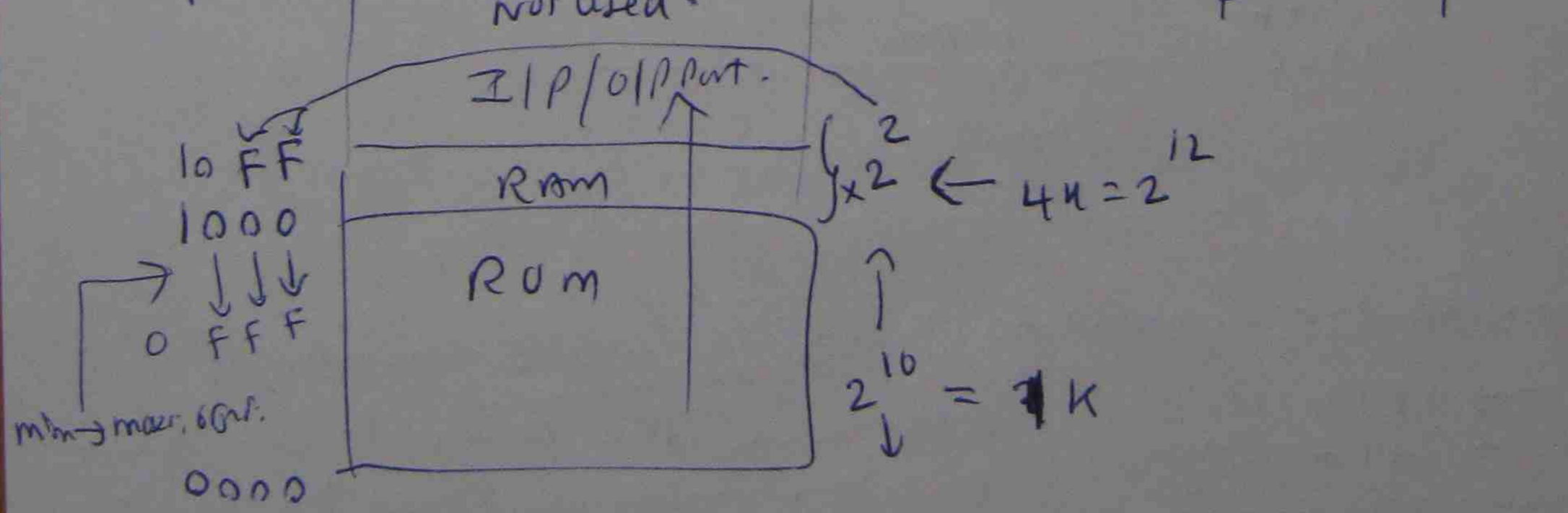
4 × 1000 4 Kbyte. Base line ⟶ 4 × 1000

$1K = 2^{10}$

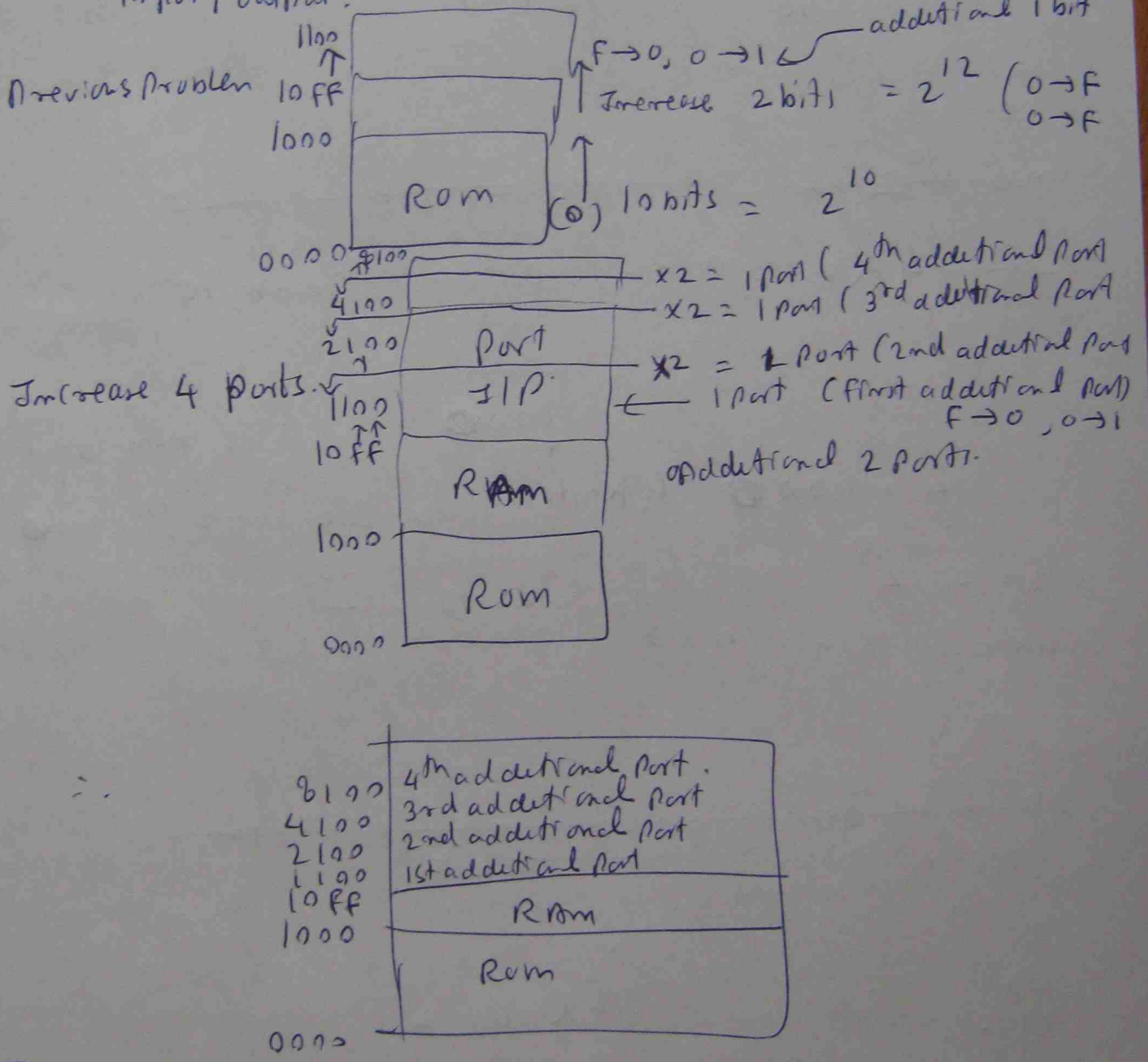
$4K = 4 \times 1K = 4 \times 2^{10} = 2^2 \times 2^{10} = 2^{12}$

8 bit = 0000 0000 ⟶ 1111 1111

FFFF 8 bit ← 256 byte F F



Q3 If the microcomputer in above question requires four additional input / output ports. Define suitable addresses for the ports assuming memory mapped Input / output.



∴

Q4 A micro computer system has the following memory map

0000 → 0FFF ROM
2000 → 21FF RAM
4000 → 400F I/O

Determine the amount of ROM & RAM memory and the number of I/O ports in the system.

4bit → 1111

0000 → 0FFF ← 1111 = 4bit, 1111 = 4bit

2000 → 21FF ← 1111 4bit, 1111 = 4bit, 0001 = 1bit

Total = 12 bit

$2^{12} \simeq 4k$

9 bit = 2^9 = 512 byte byte

4000 → 400F

1111 = 4bit

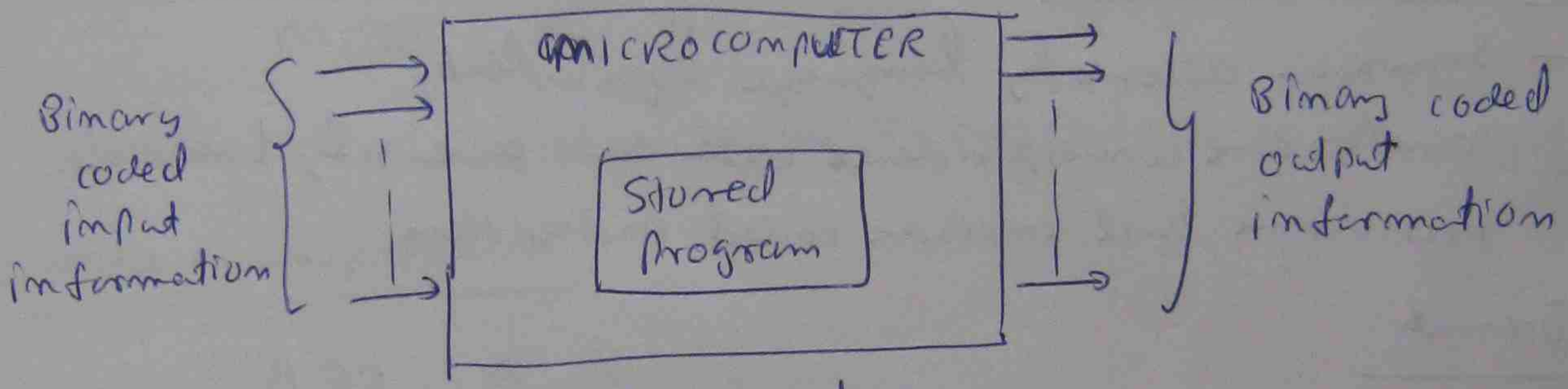
2^4 = 16 port

Week (3)

Introduction to Programming and data transfer

microcomputer

- Read binary coded information from its input ports
- manipulate this information according to a program stored within its memory
- subsequently produce output information at its output ports



※ It is necessary to become familiar with the different types of machine instructions which a typical microprocessor executes and to investigate their effect on the total system.

Intel 8085

A	
B	C
D	E
H	L
F	Im
Stack Pointer SP	
Program counter PC	

A = 8 bit arithmetic register (accumulator)

BCDE - 4 bit general purpose registers.

F = 8 bit flag register controlled by ALU

Im - 8 bit interrupt control register

HL - two 8 bit registers used to form a 16 bit memory pointer.

SP – Stack Pointer register.
16 bit memory address which displays points to the top of a system stack.

PC – Program counter register which contains a 16 bit memory address which points to the next instruction to be executed.

Assembly Language

– Symbolic assembly language equivalent.
– one to one correspondence between assembly language instruction and machine coded instruction.

Format

LABEL : OPERATION MNEMONIC , OPERANDS COMMENTS.

LABEL – Operational symbolic address for the instruction

OPERATION MNEMONIC – Tell the programmer the specific operation to be performed

OPERANDS – A value on which this operation is to be carried out
(OR)
The memory locations where the value can be found.

COMMENTS – optional comments.
To facilitate understanding & enhance the readability of the complete program
– Does not influence the machine code resulting from the assembly instruction

Classification of instructions

- Data transfer
- Data manipulation
- Transfer of control
- Input / output
- machine control

1 Data Transfer

MOV A, B

Results in B register being transfered to A register

2 Data manipulation

ADD A, B

A register (accumulator) containing the sum of its previous contents and the contents of the B register.

3 Transfer of control

- unconditional
- Subroutine call
- conditional (flag dependent) Jump instructions.
- Return instruction.

Jmp LABEL1

- microprocessor breaks its normal mode of sequential instruction execution.
- Jump unconditionally to symbolic address LABEL1 for next instruction to be executed.

4 Input / output

move data between the various input / output ports of the system and an internal processor register - usually A - register.

OUT 05

The content of the A register being transferred to output port 05 (hex).

5 machine control

Instructions in the machine control group affect the state (or) mode of operation of the processor itself.

Interrupt, enable, disable, processor halt, no operation instructions.

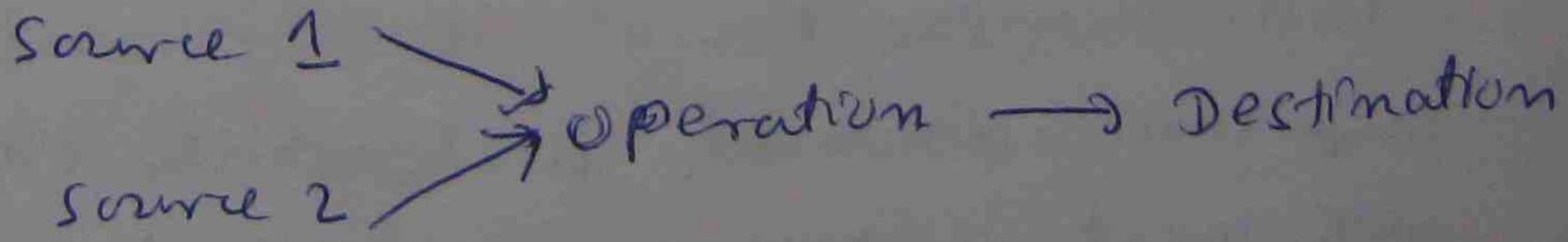
Operand addressing mode

machine instruction

2 addresses

Specify the location of the values to be manipulated (source addresses)

The third to specify the location where the result is to be stored (destination address)



Instruction addressing mode

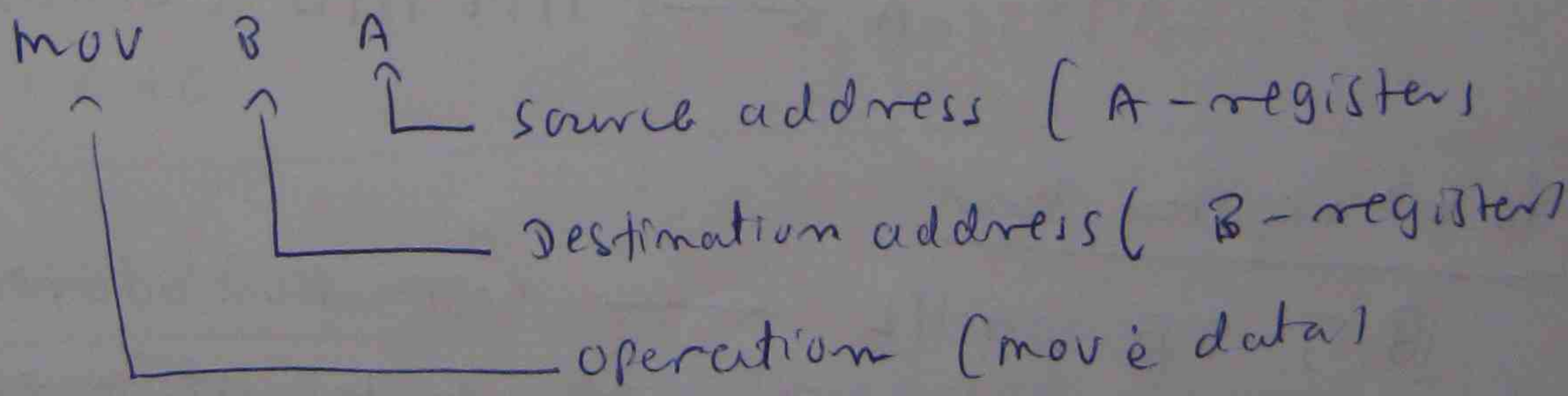
4 main types of addressing modes

Register Addressing
Immediate Addressing
} These are used primarily for data transfer and manipulation instructions which involve only the internal processor registers.

Direct (Extended) Addressing
Register Indirect Addressing
} These are used primarily for data transfer and manipulation instructions which involve the system memory.

Register Addressing

- move data between the internal processor registers.
- The instruction source and destination addresses specify which of these registers are involved in the transfer.



This results in the contents of the A register being transferred to the B-register. The contents of the A-register remain unchanged. (B) ← (A)

Data transfer involving 16 bit register pairs

XCHG

(DE) ⟷ (HL)

The contents of register pair DE being exchanged with the contents of register pair HL.

Immediate Addressing

The source address does not specify a register (or) memory location but instead the actual source data is contained with in the instruction itself. and is therefore immediately available.

MVI A FE(hex)

- FE(hex) — source data
- A — Destination address
- MVI — operation.

The data value FE(hex) being transferred to the A register

(A) ← FE(hex)

(or)

A ← 1111 1110 (binary)

16 bit register pair

B, D, H or SP —— Destination address.

These instructions require two byte of immediate data

LXI H, 8020 { 16 bit register pair HL being loaded with immediate data 8020 (hex)

(H)(L) ← 8020 (hex)

LXI D, E627 { (D)(E) ← E627 (hex)

REGISTER DATA TRANSFER

Pb(1)

1 – The Program loads a value into A register using immediate addressing

– Then loads this value into two further registers

2 B and C using ~~two~~ register addressing.

– Finally Register Pairs HL and DE are loaded

3 a/b with 802D & E627 using immediate addressing. ~~Then their~~

4 – Then their contents are exchanged using register addressing.

Assembly Instructions.

MNEMONIC	OP1	OP2	comments
1 MVI	A	FE	(A) ← FE (hex)
2 MOV	B	A	(B) ← (A)
2 MOV	C	B	(C) ← B
3a LXI	H	802D	(H)(L) ← 802D (hex)
3b LXI	D	E627	(D)(E) ← E627
4 XCHG			(D)(E) ⟷ (H)(L)

Direct Addressing

An operand may be either read from (or) written to a memory location, the address of which is specified in the instruction itself.

Load

Ex LDA 20EA (OR) (A) ← (20EA)

Register A loaded with content 20EA (hex)

~~A loaded~~ (Loaded) ← (content)

Store

Ex STA 20f2 (OR) (20f2) ← (A)

Register A being stored in memory location

20 f 2 (hex)

(Memory location) ← (Register)

Hold

Simultaneously

LOAD

Ex

LHLD 20A2 (OR) (L) ← (20A2)
(H) ← (20A3)

Register

content

~~Register~~ Register Pair, Pair ~~I~~

LOAD HOLD

Register 2
~~Register~~ content
~~pair 2~~

Register Pair H and L is frequently used to hold a combined 16 bit memory address.

This address for it address increase 1 for reg: L

~~address~~

Ex Store simultaneously

SHLD 20AF ← Store

20A~~F~~2

Register

content

~~20AF~~ content of Register ~~(H)~~ is stored in 20A~~F~~2

(L) is stored in 20A3

(L) ← (20A2)
(H) ← (20A3)

Ex SHLD 20AF
(20AF) ← (L)
(20B0) ← (H)

DIRECT ADDRESSING (23)

Pb(2)

- Immediate and direct addressing of register A at FF and EE

- ~~Data 20A2 and another consecutive memory location~~

  Data in A is stored at address 20A2 and another consecutive memory location.

- Load register pair HL with these data

- Store the same data loaded at HL into a two consequent memory locations.

Immediate Address(L) Store(L) Immediate address Store(2)

	Assembly Instructions		Actions
1(a)	MVI A, FF		(A) ← FF(hex)
2(a)	~~MVI~~ STA 20A2		(20A2) ← (A)
1(b)	MVI A, EE		(A) ← EE(hex)
2(b)	STA 20A3		(20A3) ← (A)
3	LHLD 20A2	Load	(L) ← (20A2) FF (hex) (H) ← (20A3) EE (hex)
4	SHLD 20A4	Store	~~(L) (H) (20~~ (20A4) ← (L) (20A5) ← (H)

Register Indirect Addressing

Direct Addressing	Indirect addressing
	more efficient method
– Only the A register may be used to store or load a value to and from memory	– Data may be transferred between any of the Processor registers and the system memory
– If a value were to be stored in a memory location, the B register	– Operand is either read from (or) written to memory location
(1) Transfer contentents from B to A	– Instruction does not contain actual memory address
(2) Then store operation is performed	– Operand is either read from (or) written to the memory location, the address of which is currently stored in the register pair HL

Ex mov A M

A register being loaded with the contents of the memory location whose address is specified in register H and L (A) ← (H)(L)

Ex mvI M, FF

The value FF(hex) being stored in the memory location whose address is in register H & L ((H)(L)) ← FF(hex)

Indirect Addressing

Ph (3)

1 The memory address of A which contains data value AA is 2040 and it is loaded into register H and L by indirect addressing mode.

LXI H

2 Then a value is moved to memory location using register indirect addressing.

MVI M, AA

3 The value is loaded into two further registers B & C using register indirect addressing

MOV B, M

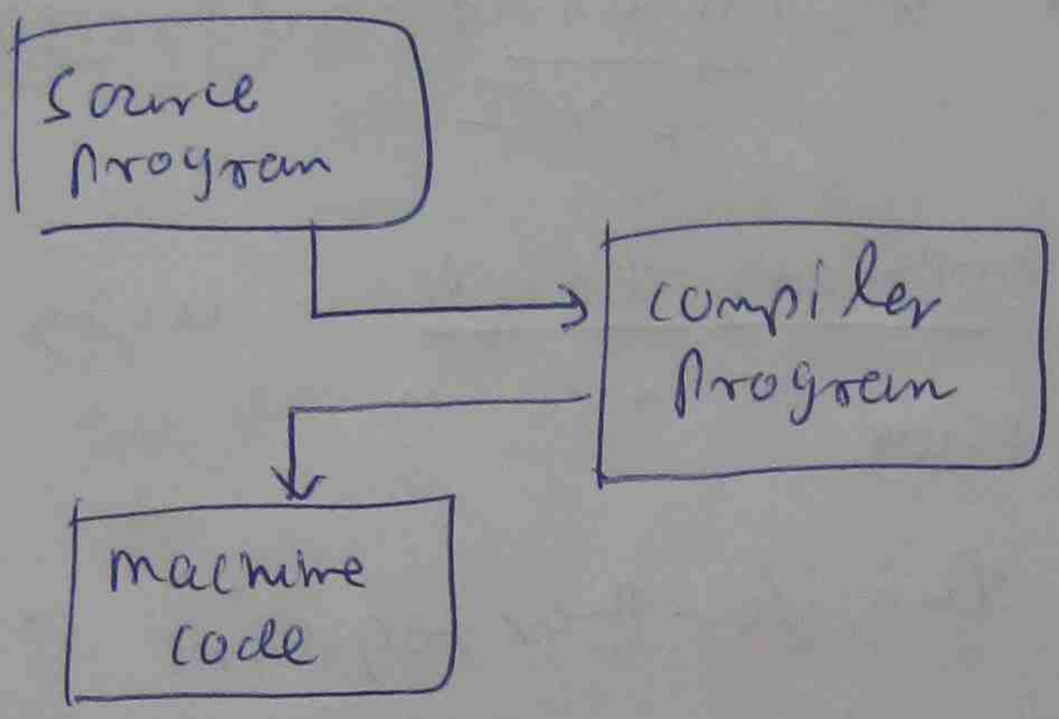
MOV C, M

	Assembly Instruction	Action
1	LXI H, 2040	(L) ← 40 (hex) (H) ← 20 (hex)
2	MVI M, AA	(H)(L) ← AA (hex) (OR) 2040 ← AA (hex)
3	MOV B, M MOV C, M	(B) ← (2040) (OR) (B) ← AA (hex) (C) ← (2040) (OR) (C) ← AA (hex)

The Assembly Process

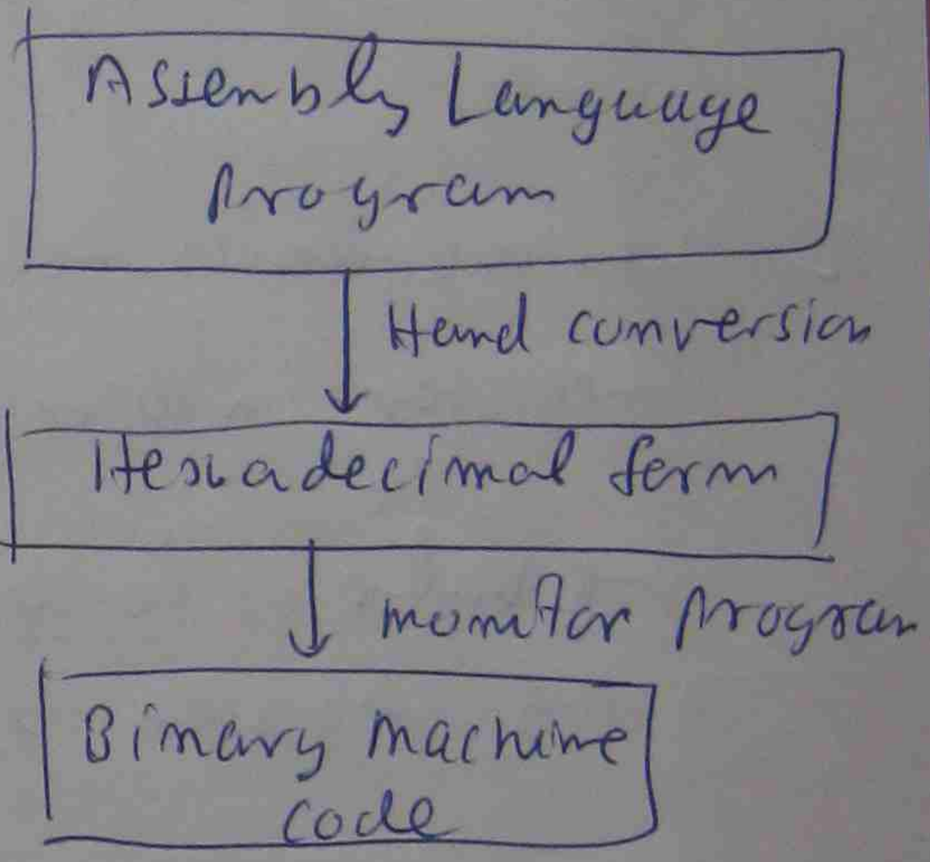
compiler

The compilation process



Assembler

The hand assembly process



8085 Instruction set

MOV	A, A	7f
	A B	78
	A C	79
	A D	7A
	↓	
	M L	75

Appendix (1) Page 140 → 144

move Immediate

MVI	A byte	3E
	B byte	06
	m byte	36

Load Immediate

(Reg, Pair)

LXI	B dble	01
	D dble	11
	H dble	21
	SP dble	31

Load / Store A direct

LDA	addr	3A
STA	addr	32

Load / Store A indirect

LDAX B	0A	STAX B	02
LDAX D	1A	STAX D	12

Load / Store HL direct

LHLD	addr	2A
SHLD	addr	22

Exchange HL/DE

XCHG	EB

Ex MOV A B

(A) ← (B) requires 7, 8

Ex MVI A, FE

(A) ← FE(hex) requires 3 E, FE

operation

Immediate data

Ex STA 2052

32 operation

most significant byte of memory address

Least significant byte of memory address

Pb write the hand coding for the following operation

① the Program load, a value into A register using Immediate addressing. at the address 2000

② Then loads this value into two further registers B and C using register addressing.

3 a/b Finally register pair HL and DE are loaded with 3020 & E027 using immediate addressing

④ then Their contents are exchanged using register addressing.

(28)

LINE NO.	Memory		Assembly Instruction			Comment
	Address	content	MNEMONIC	OP1	OP2	
~~2000~~ 1	2000	3E	MVI	A	FE	(A)←FE hex
	2001	FE	3E for 3085			
	2002	47	MOV	B	A	(B)←(A)
	2003	4F	MOV	C	B	(C)←(B)
		Although data finally reached	MOV CB is 48	A	MOV C,A	EA 4F
	2004	21 for LXI H	LXI	H	202D	(H)(L)← 202D
	2005	2D				
	2006	80				
	2007	21	LXI	D	E027	(D)(E)← E027
	2008	27				
	2009	E0				
	200A	EB	XCHG			(D)(E)↔ (H)(L)

Nvt 2010

200(A) ← 10

Week (4)

Data manipulation

Microprocessor - Represent data within an number of different form

Typical Arithmetic instruction.

Data representation

In general, numbers may be represented in unsigned binary, signed binary or binary coded decimal (BCD) form.

Unsigned binary

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	= weightings
0	0	1	0	1	0	1	1	= 43 (Decimal)
0	1	0	0	0	1	1	0	= 70
1	0	1	0	0	0	0	1	= 161
1	1	0	0	1	1	0	0	= 204

Signed binary

26 =

2	26	0
2	13	1
2	6	0
2	3	1
	1	

= 11010

for 8 bit

0 00011010 = +26

7 bit

100 =

2	100	0
2	50	0
2	25	1
2	12	0
2	6	0
2	3	1
	1	

= 1100100

for 8 bit

1 1100100 = −100

This form is not possible to perform Arithmetic operation

Two's complement form of representation

S	6 5 4 3 2 1 0

S=0 for positive number and zero

S=1 for negative numbers.

Ex1 +15

2	15	1
2	7	1
2	3	1
	1	

1111

for 8 digit = 00001111

Invert) 11110000

+1) Increase 1

−15 = 11110001

Ex2 +89 =

2	89	1
2	44	0
2	22	0
2	11	1
2	5	1
2	2	0
	1	

1011001

FOR 8 bit = 01011001

Invert) 10100110

+1)

−89 = 10100111

Table Two's complement Representation

Decimal Number	Two's complement
+127	0 1111111
⋮	
+3	0 0000011
+2	0 0000010
+1	0 0000001
0	0 0000000
−1	1 1111111
−2	1 1111110

$-3 \longrightarrow$ 1 1 1 1 1 1 0 1

1 0 0 0 0 0 0 1

-127

-128 1 0 0 0 0 0 0 0

Binary coded Decimal (BCD)

It the input data is from a decimal key pad and the subsequent output data drives a decimal display, use decimal number representation and arithmetic → provide instructions for performing arithmetic on binary coded decimal (BCD) number.

BCD representation is a subset of the hexadecimal system

BCD code

decimal digit	BCD code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

8 bit binary code may be used to store two BCD numbers.

86

1000 0110

(decimal)

$\therefore$ 86 = 1000 0110

51 (decimal)

0101 0001

$\therefore$ 51 = 0101 0001

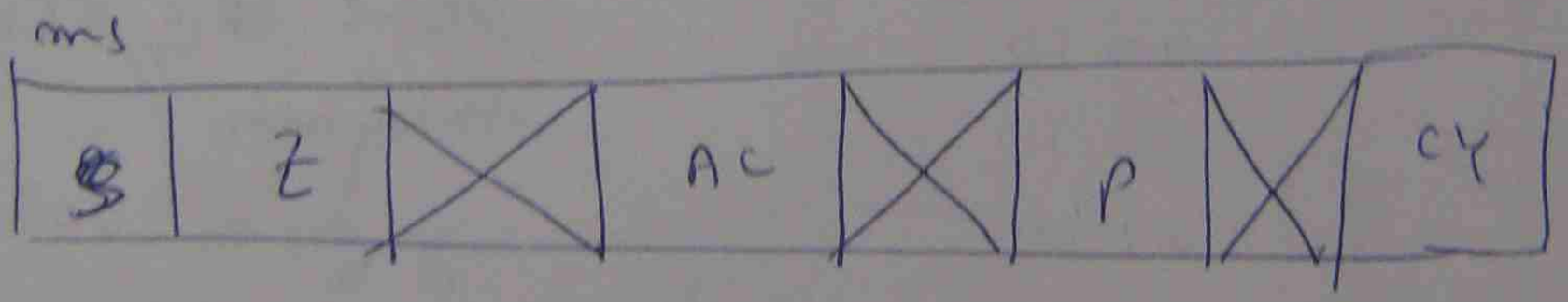
Arithmetic instructions

Arithmetic instructions → add, subtract, increment, decrement

8085 The instructions always involve A register and either another processor register (or) a memory location.

- A microprocessor contains a number of different forms of these instructions so that data can be manipulated in the selected manner.

Flag - Status (or) condition bit
- which are either set (or) reset depending on the particular arithmetic instruction being carried out and the programmer is able to use and interpret these flags to manipulate data in the selected way



F register of 8085

S =	Sign flag	If is set when the result of an arithmetic operation is negative
Z	Zero flag	- The flag is set if the result of an arithmetic operation on the register is zero, otherwise it is reset. - This is used with transfer of control function
AC	Auxiliary Carry	BCD number representation is being used If is set when the result of an arithmetic operation produces a carry out from the least significant half of the A-register

P	Parity flag	This is used with logical operation. The flag is set if the result of a logical operation (AND, OR, XOR) produces an even number of 1's
CY	Carry Flag	CY is set after an ADD instruction if a carry out was generated from the A-register

Addition of two bits

Bit 1	Bit 2	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

1st 2nd 1st 1st Bit Bit
0 + 1

A = 1001 1010 = 154 decimal

B = 0101 0111 = 87 decimal

241 decimal

100 ← carry in

001 ← sum

Addition of two bits & a carry in

	Bit 1	Bit 2	Carry in	Sum	Carry out
(1) 1st	0	0	0	0	0
(2) 2nd	0	0	1	1	0
(3) 1st	0	1	0	1	0
(4) 2nd	0	1	1	0	1
(5) 1st	1	0	0	1	0
(6) 2nd	1	0	1	0	1
(7) 1st	1	1	0	0	1
(8) 2nd	1	1	1	1	1

1001 1010

\+ 0001 0010

0101 0111

1100 1101

A = 1 0 0 1 1 0 1 0 = 154 decimal

B = 0 1 0 1 0 1 1 1 = 87 decimal

1 0 0 1 0 0 sum

0 0 1 0 0 1 0 carry

A :- 1 0 0 1 1 0 1 0 = 154 decimal

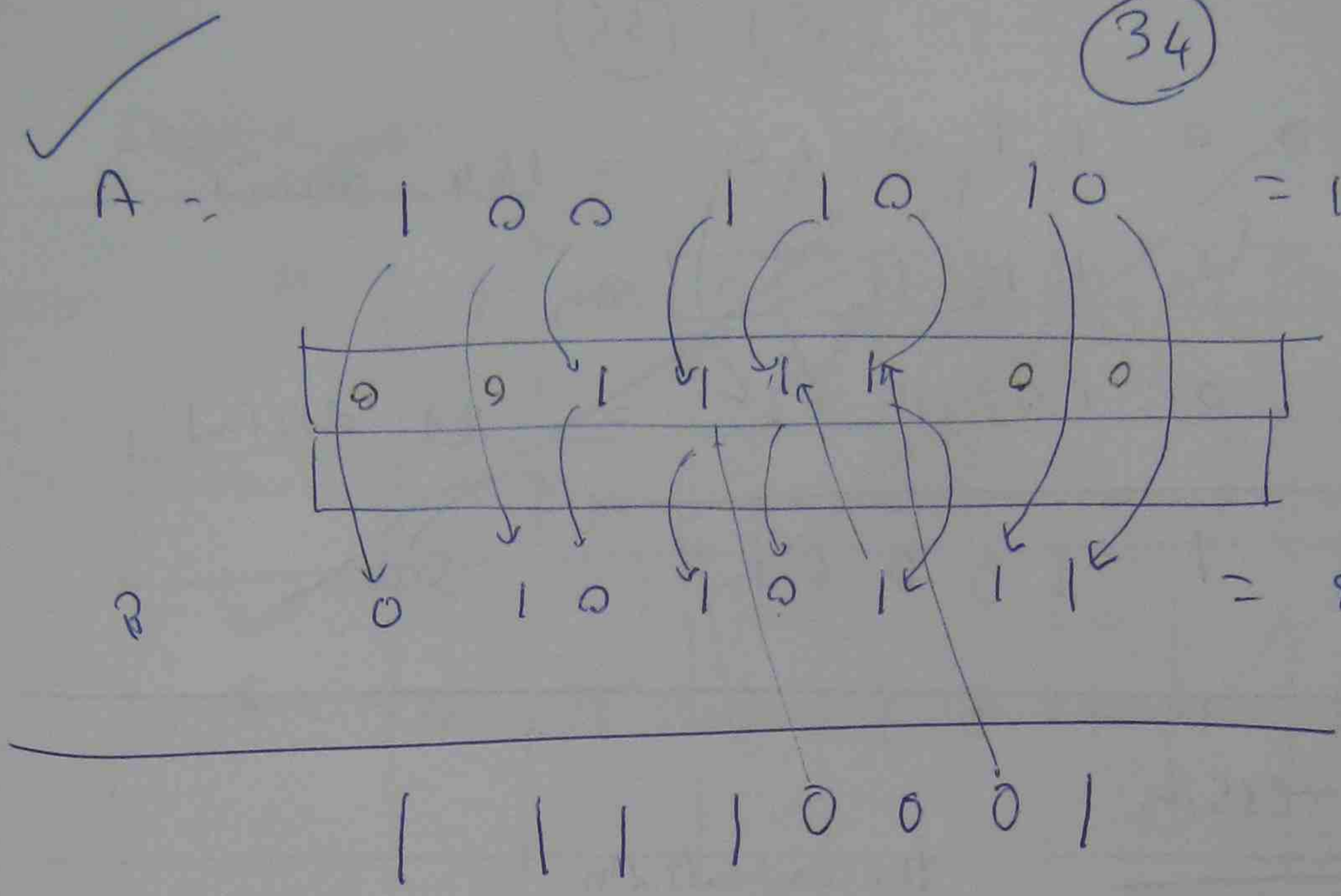
B :- 0 1 0 1 0 1 1 1 = 87

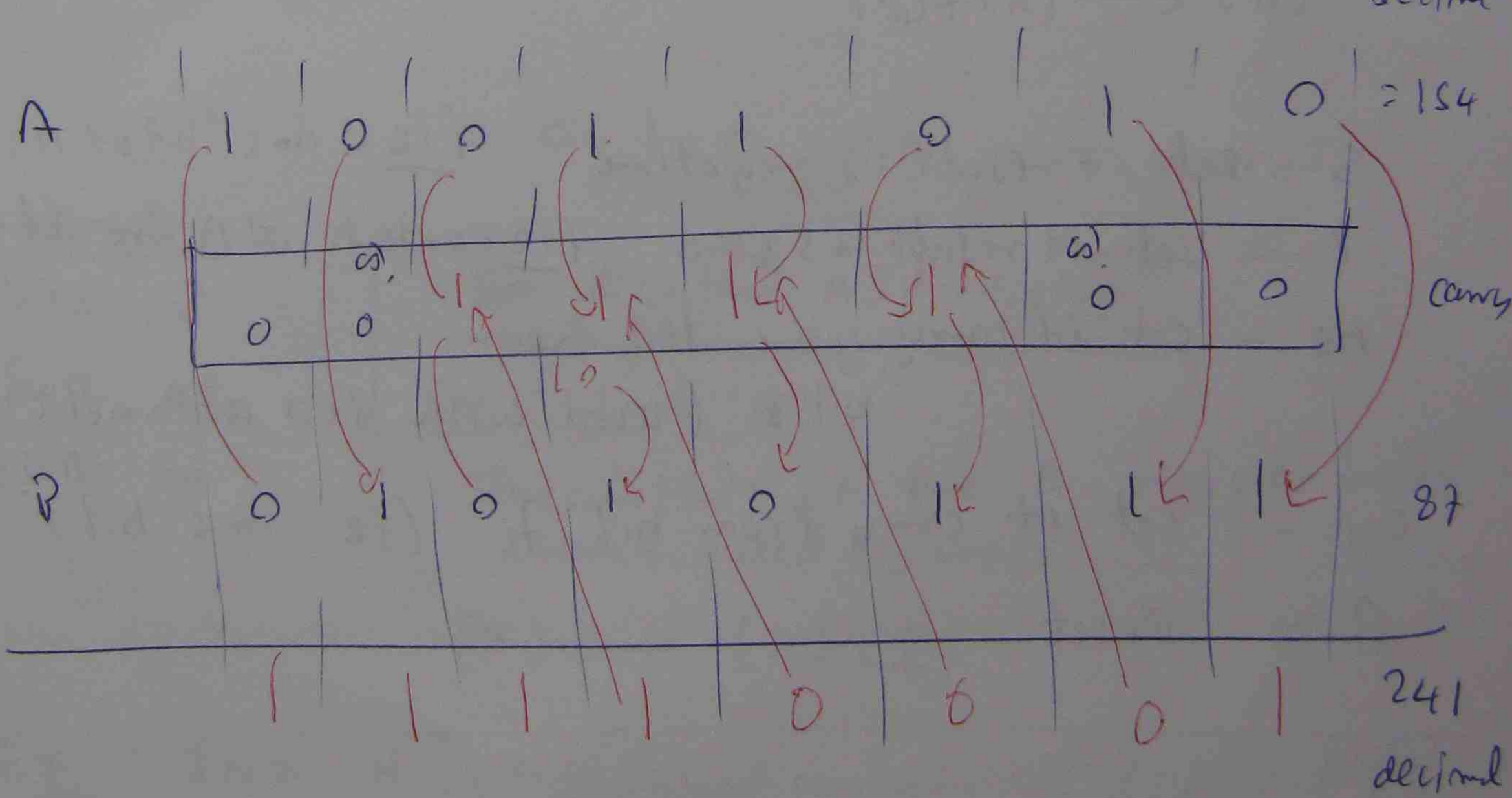
1 1 1 1 0 0 0 1 = 241 decimal.

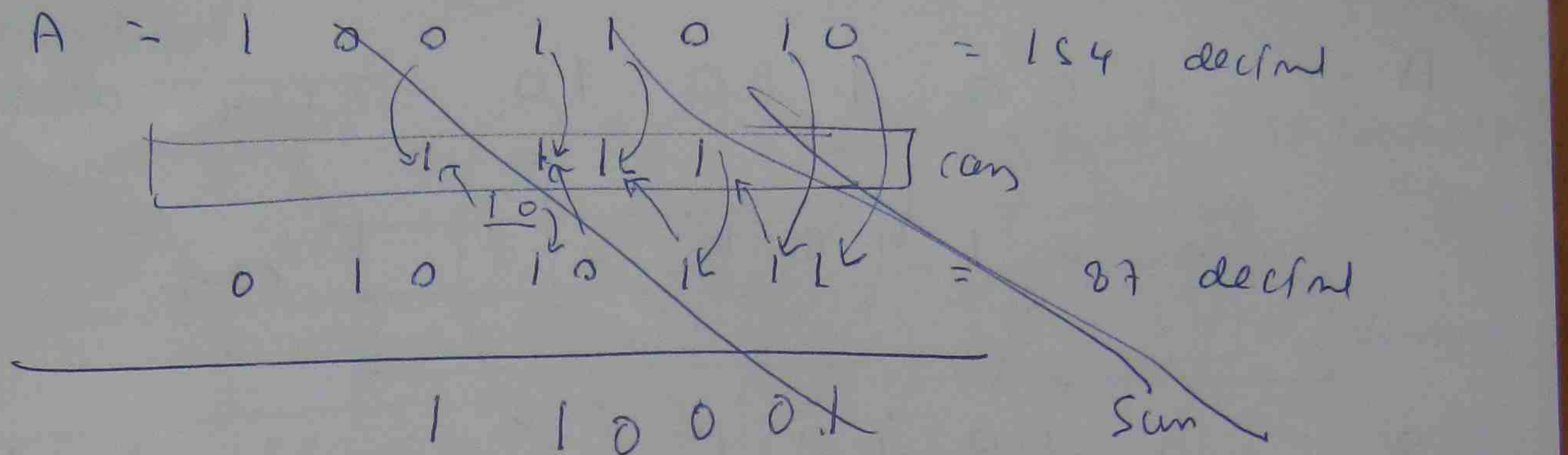
1 0 0 1 1 0 1 0

0 1 0 1 0 1 1 1

1 0 0


A = 1 0 0 1 1 0 1 0 = 154 decimal
0 0 1 1 1 0 0 carry.
B 0 1 0 1 0 1 1 1 = 87 decimal
1 1 1 1 0 0 0 1


A 1 0 0 1 1 0 1 0 = 154
0 0 0 0 carry
B 0 1 0 1 0 1 1 1 87
1 1 1 1 0 0 0 1 241
decimal



Register Addressing

Ex ADD B — the contents in B register is being added to current contents of the A register

(A) ← (A) + (B)

S = set if result is negative (ie ms bit of A is 1
Z - set if result is zero (ie contents of A are all 0s)
AC - set if carry generated from bit 3 (used with BCD arithmetic)
Cy - set if carry from bit 7 (ie ms bit)
P - Reset

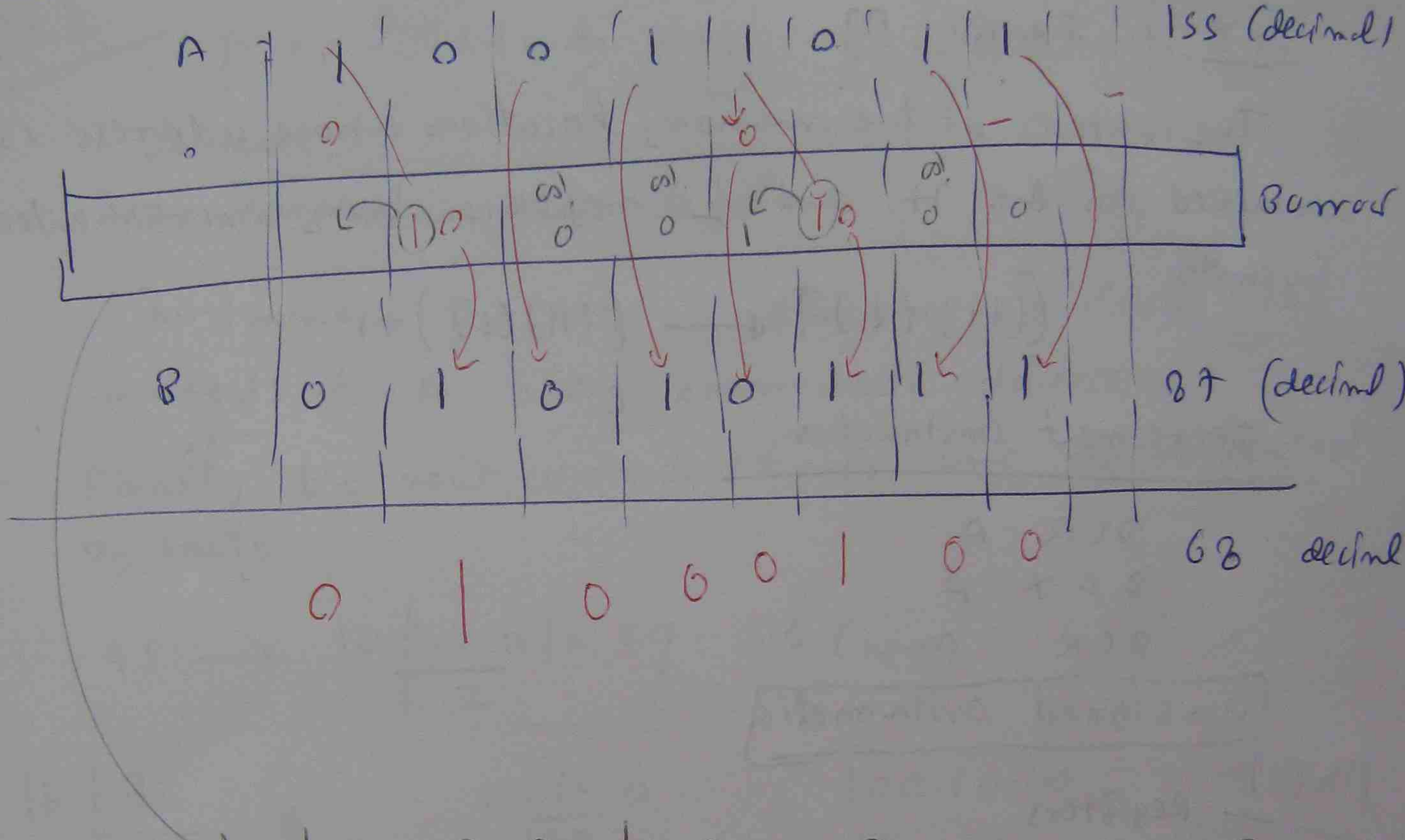
Ex ADI 0F

The immediate data 0F (hex) being added to the current contents of A register

(A) ← (A) + 0F

all Flags are affected.

Substract



Substract

SUB	B
SUI	0f
SUB	m

~~Re~~ Register Addressing

EX INR A

The contents of the A-register being incremented by units.

(A) ← (A)+1

Ex INX H combined contents of register pair H & L being incremented by units

(H)(L) ← (H)(L)+1

Register Indirect Addressing

Ex INR M

The contents of the memory location whose address is contained in the H and L registers being incremented by units.

((H)(L)) ← ((H)(L))+1

Decrement instruction

DCR A
DCX H
DCR M

Unsigned arithmetic

Pb(1) Registers

- A is loaded with immediate data 53 (hex), B is loaded with immediate data 3A.
- Their contents are added together.
- A third numeric is then substracted from the contents of A using immediate addressing. 8C (hex)
- Finally the new contents of A are decremented by units.

Assembly Instruction	Action
MVI A, 53	(A) ← 53 (hex)
MVI B, 3A	(B) ← 3A (hex)
ADD A	(A) ← (A)+(B)
SBI 5C	(A) ← (A)−8C (hex)
DCR A	(A) ← (A)−1

Signed Arithmetic

Ph 2 A is loaded with +35 decimal, B is loaded with −72 decimal.

Their contents are added together

Third numeric DB (hex) is substracted from the contents of A using immediate addressing.

Finally the new contents of A are decremented by unity.

$35 \longrightarrow 16 \left|\frac{35-32}{2}\right| = 3 \rightarrow 23$ (hex)

16	72
	64
	8

2	72	0
2	36	0
2	18	0
2	9	1
2	4	0
2	2	0
	1	

$1001000 = +72$

8 bit

0100, 1000 → Invert

1011 0111 → +1

1

$1011\ 1000 = -72$

$-72 = B8$ (hex)

Assembly Instruction	Action
MVI A, 23	(A) ← 23 (hex)
MVI B, B8	(B) ← B8 (hex)
ADD B	(A) ← (A) + (B)
SBI DB	(A) ← (A) − DB (hex)
DCR A	(A) ← (A) − 1

16 bits arithmetic

8 bit – only B
16 bit B & C → BC

8 bit only D
16 bit only D & E → DE

8 bit – only H
16 bit – H & L → HL

EX INX B

(B)(C) ← (B)(C) + 1

EX DCX D

(D)(E) ← (D)(E) − 1

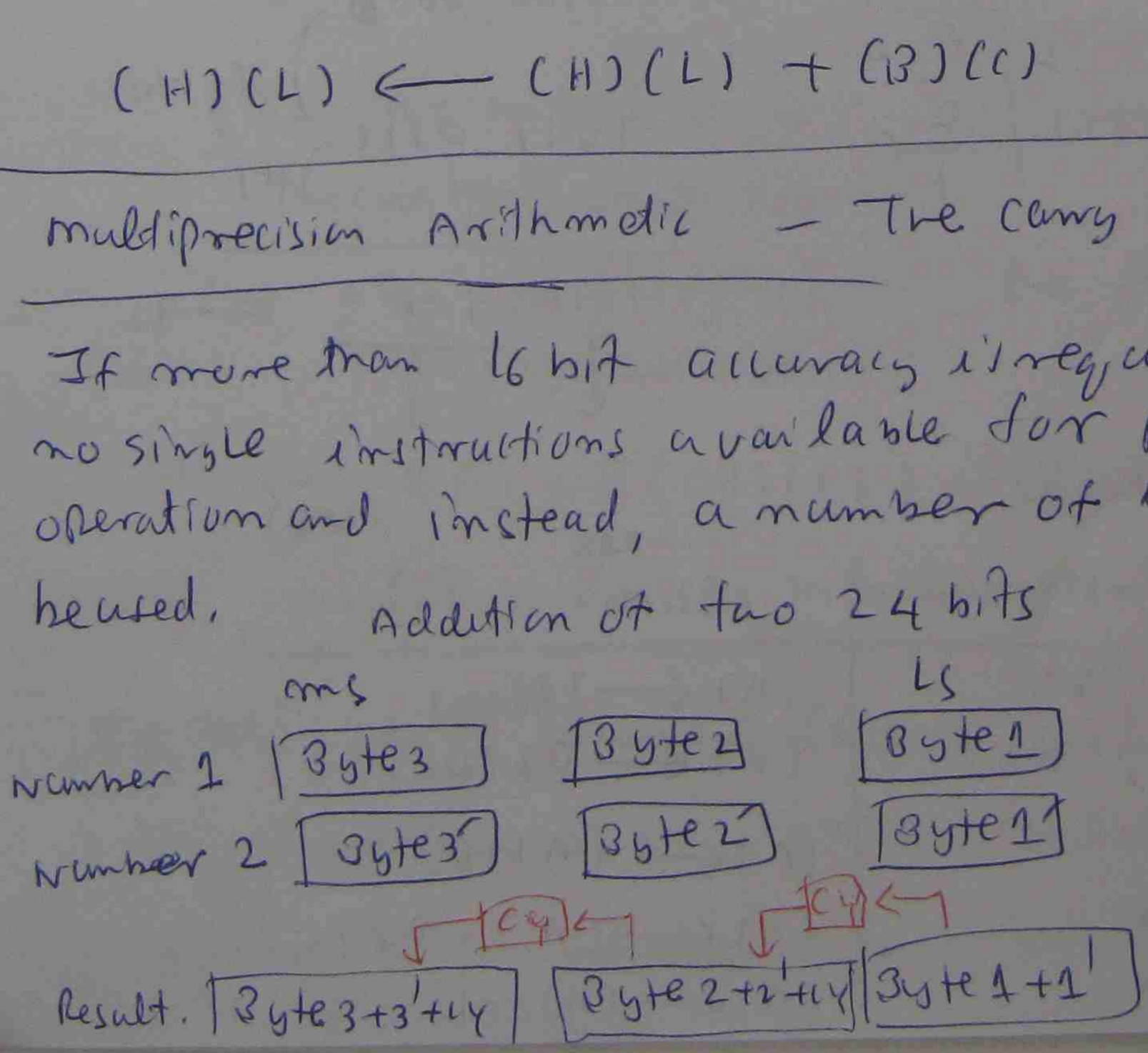
EX DAD B

16 bit contents of the register pair B,C being added to the 16 bit contents of the register pair H,L.

(H)(L) ← (H)(L) + (B)(C)

multiprecision Arithmetic – The carry flag

If more than 16 bit accuracy is required, there are ~~no~~ no single instructions available for performing arithmetic operation and instead, a number of instructions must be used. Addition of two 24 bits



~~Pb~~ EY ADC M

(1) The contents of the memory location whose address is contained in registers H & L

and

(2) The contents of CY Flag being added to the contents of A Register

(3) The result is placed in the A register and all flags are affected.

(A) ← (A) + ((H)(L)) + (CY)

EY SBB M

(1) The contents of the memory location whose address is contained in registers H & L

and

(2) The contents of CY Flag being substracted from the contents of A register

(3) The result is placed in the A register and all flags are affected.

(A) ← (A) − ((H)(L)) − (CY)

Pb 3 multiprecision Arithmetic

24 bits (3 byte) number. first is loaded in 2030, 2nd is loaded in 2031, 3rd is loaded in 2033.

< 24 bits (3 byte) number which is stored in the three consecutive memory location starting at address

2080 to 2082)

They are added together at 2083.

2 Last bit result result replaces first number.

1st → LXI H 2083 (Initialize to contain 2083

1	Load the number at 2080	LDA 2080	Add 1st pair
	Add memory	ADD M	of byte
	Stored	STA 2080	

INX H Increment HL

2	Load the number at 2081	LDA 2081	Add 2nd pairs of byte, together with carry
	Add memory contrive to add 2 carry	ADC M	
	Stored	STA 2081	

INX H Increment HL

3	Load the number at 2082	LDA 2082	Add 3rd pairs of bytes together with carry
	Add memory 2 carry	ADC M	
	Stored	STA 2082	

Replace

Add at , the result is added to 2083 which replace the first number

Program

```
LXI   H, 2083
LDA   2080
ADD   M
STA   2080
INX   H
LDA   2081
ADC   M
STA   2081
INX   H
LDA   2082
ADC   M
STA   2082
```

BCD Arithmetic

Ph (1) Add 62 BCD and 25 BCD

62 = 6 → 0110, 2 → 0010

25 = 2 → 0010, 5 → 0101

∴ 62 + 25 = 0110 0010 + 0010 0101

1000 0111

No carry beyond 4 bits

$C_y = 0$

1000 → 8, 0111 → 7 BCD

Ph (2) Add 79 BCD & 16 BCD

79 = 7 → 0111, 9 → 1001

79 + 16 = 0111 1001 + 0001 0110

1000 1111

Pb 3 Add 39 BCD & 48 BCD

39 = 3 → 0011, 9 → 1001

48 = 4 → 0100, 8 → 1000

39 + 48 =

```
  0011   1001
  0100   1000
  ------------
  1000 (1)0001
         ↑ carry
```

Carry ∴ CY = 1

no bit to put

1000 0001 ← Normal BCD

But 39 + 48 = 87 ∴ Reqd. BCD = 87 → 1000 0111

Required BCD 1000 0111 > Normal BCD 1000, 0001

∴ AC = 1

∴ CY = 1, AC = 1

To correct it 1000 0001 will needs to be added with 0110

∴

```
  1000, 0001
  +     0110  ← (6)
  ----------
  1000  0111
```

to get 1000, 0111 or 87 hex

$(+6)_{hex}$ is corrected BCD sum (0110)

∴ Decimal Adjust Accumulator DAA is utilized.

<AC Flag is set 6 is added to A register>

Ph Add the followings BCD nubers & make comection.

(a) 47 BCD + 32 BCD ~~(b) 81 BCD + 69 BCD~~

(a)

47 BCD = 0100 0111

32 BCD = + 0011 0010

7a BCD 0111 1001 ← normal BCD

↓ 1001 + ∴

1001, ~~0001~~ 0000 0000 corrected BCD

↑ The same

Required BCD ←

~~(b) 81 BCD =~~

Ph Substract the following BCD numbers and make the correction

(a) 47 BCD − 32 BCD (b) 81 BCD − 69 BCD

(a)

47 BCD = 0100 0111

32 BCD = 0011 0010

---

15 BCD = 0001 0101 ← Normal BCD

Reqd BCD: 0001, 0101 ← The same

(b) 81 BCD = 1000 0001

69 BCD = 0110 1001

---

12 BCD = 0010 1000 ← Normal BCD different

↓

0001, 0010 ← Reqd.

∴ Normal BCD ≠ Reqd. BCD different

∴ AC = 1

Correction:?

Required = 1111 0010 − 0001 0010 = 0000

Normal = 0001 1000

ADD → 1111 1010

F A

0001 0010
− 0001 1000
1111 1010

Hexi decimal numbers

4 bit binary	Hexi decimal symbol
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

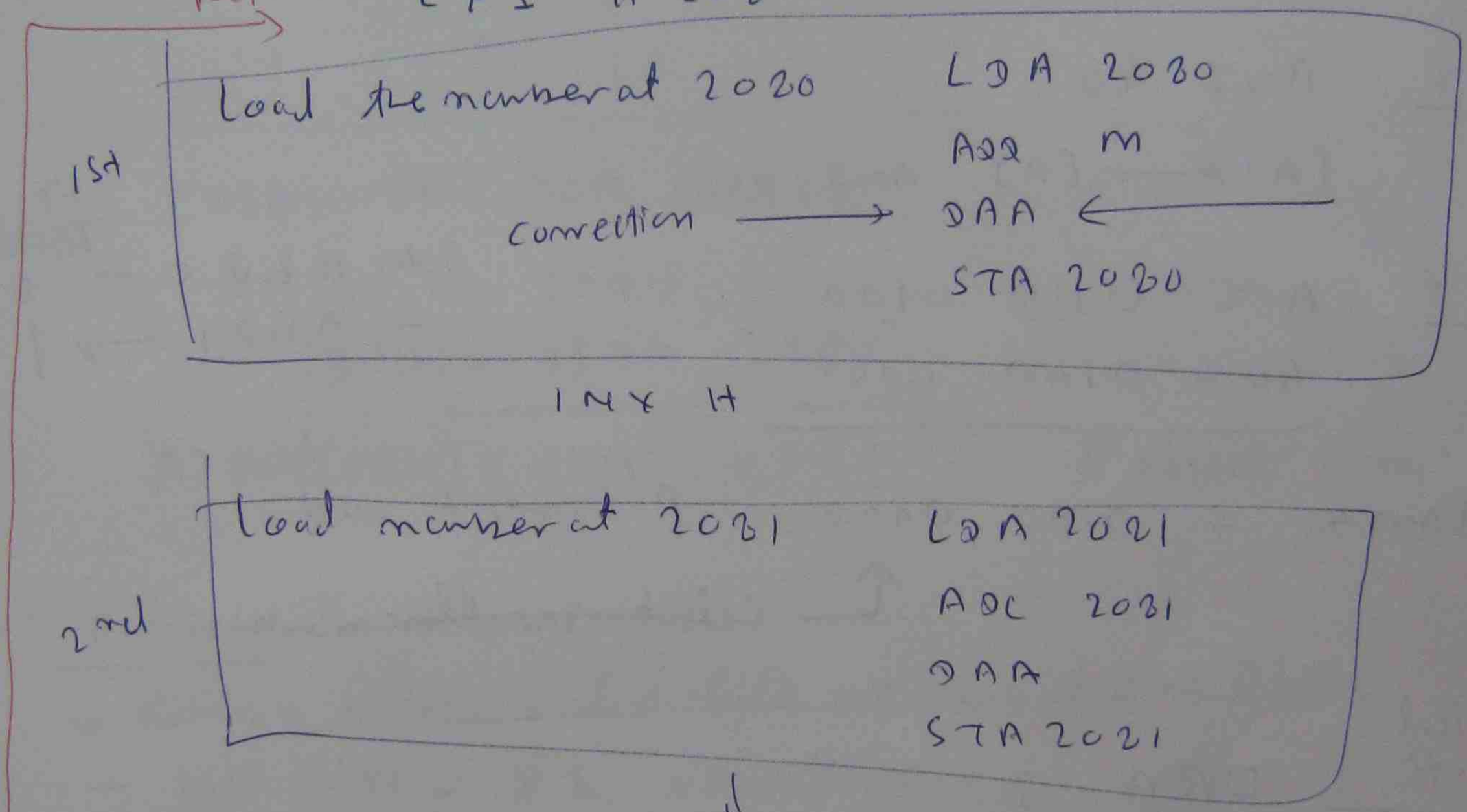
Logical operation

Ex First number 2nd number

A is loaded at 2020 and B is loaded at 2021

They are added together and correction is made

The result is put into 2082 and 2082 replaces first number



Program

```
LXI   H   2082
LDA       2020
ADD       M
DAA
INX   H
LDA       2021
ADC       2021
DAA
STA       2021
```

Carry CY	Upper hex digit b7 7-4	Aux carry AC	Lower Hex b7 3-0	correction
0	0-9	0	0-9	00
0	0-8	1	6-F	FA
1	7-F	0	0-9	A0
1	6-F	1	6-F	9A

Logical operations

Logical AND

The Logical AND instructions perform the bit by bit AND operation between the contents of the A-register and either immediate data (or) the contents of another Processor register (OR) a memory location.

Ex ANI 40

C

(A) ← (A) AND 40

Ex A = 0110 0100
40 = 0100 0000

3OM A & B 0 → Result 0
— A & B 1 → 1

(A) AND 40 = 0100 0000

↑ ~~Select smaller~~ number

P is reset odd.

Logical OR

ORA B

The contents of A register are ORed with those in B register

(M) ← (A) OR (B)

Ex (A) = 0110 0100
(B) = 1001 0101

Either A or B(1)
Result (1)

~~Select bigger~~

(A) XOR (B) = 1111 ~~0000~~ 0101

(A) + (B)

OR		
Bit 1	Bit 2	Bit(1) (or) Bit(2)
0	0	0
0	1	1
1	0	1
1	1	1

~~1110~~
~~1110 0001~~

XOR function truth table

Bit 1	Bit 2	Bit 1 XOR Bit 2
0	0	0
0	1	1
1	0	1
1	1	0

1st

2nd

Ex (A) ← (A) XOR (H)(L)

(A) = 1001 1011

(H)(L) = 1100 1101

(A) XOR (H)(L) = 0101 0110 P is set even

Rotate

- Rotate a binary value left or right one place
  - Left shift = X 2 operation
    Right shift = /2 operation
  - useful for performing multiplication & division

Ex (A) = 1001 0110

RLC(A) = 0 0 1 0 1 0 1 1 (CY) ← 1

1 carry.

∴ CY ← 7 ← 0

↻ Rotate left

Compare

Compare two values - the contents of the A-register and either immediate data (or) the contents of a processor register or memory location.

CMP B

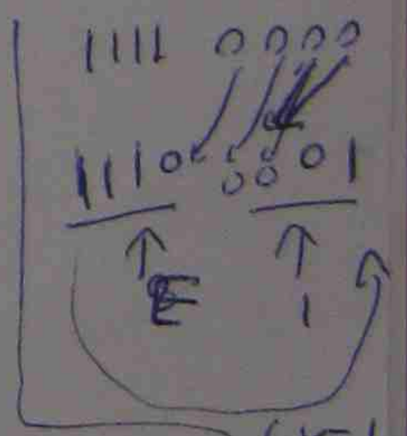
The contents of B registers are compared with the contents of A register.

Z flag is set ~~zero~~ to 1 if the contents are equal and the cy flag is set to 1 if contents of A are less than contents of B

Ex The Program Instrd to & of
First loads immediate data into registers A and B and then performs a series of logical operation

First the contents of and B are compared
2nd the ~~rotated~~ contents of A are then rotated left
3rd the new contents of A are then AND-ed with a constant 81
4th the resulting contents of A are OR-ed with the contents of B

(A) F0 = 1111,0000
(B) 0F = 0000 1111



Instrd	MVI A, F0	(A) = F0 (hex)
	MVI B, 0F	(B) = 0F (hex)
1st	CMP B	Z ← 0 CY ← 0
2nd	RLC	(A) ← E1 CY ← 1
3rd	ANI 81	(A) ← 81, CY ← 0, P ← 1
4th	ORA B	(A) ← 8F

MVI A F0 A = 1111, 0000

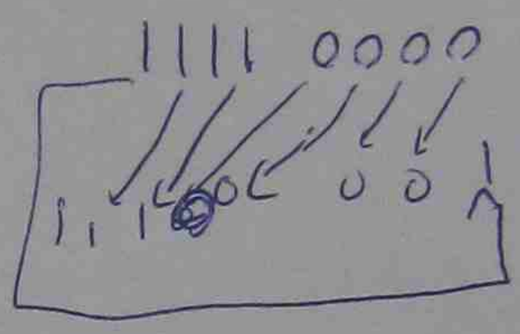
MVI B 0F B = 0000, 1111

1st CMP B A 1111, 0000, B 0000 1111

∴ A > B ∴ A = 1111, 0000

2nd RLC

A = 1111 0000



= 1110 0001

↑ CY ↑ 1

3rd ANI 81

∴ (A) CY = 1

Compare

1110 0001

81 => 1000 0001

Smaller Thesae

1000 0001 = 81

4th ORA B

A = 81 = 1000, 0001

B = 0F = 0000 1111 > OR

1000 1111

↓ ↓

8 F

Exercise Look at exercise questions 4.1 to 4.8
check the answer provided for exercise 4.1 to 4.8
Determine the ways to approach the solution.
Take exercise with microprocessor Training software

week(5)

Transfer of control

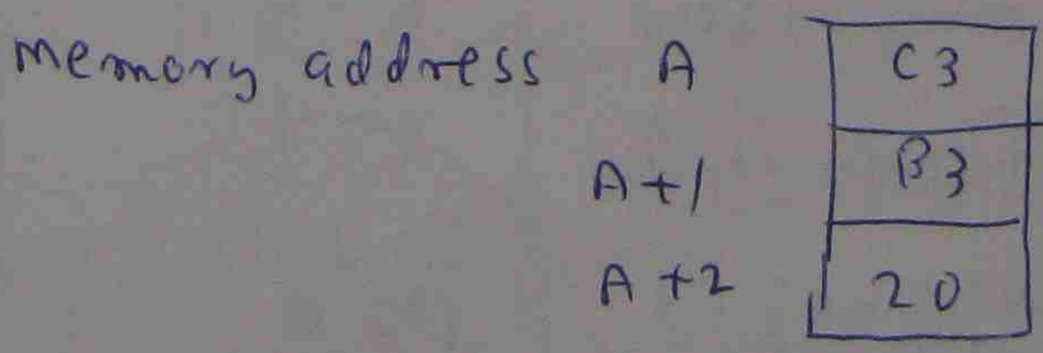
- To achieve the ability to transfer control, branch or to an instruction that is not in sequential order, it is achieved with instructions from the transfer of control group.
- All these instructions act on the program counter.
- It is possible to execute a block of instructions many times over with the number of times determined either by Program data or the state of a Processor flag.

Jump instructions

- Break normal sequential execution
- Branch to a different part of the Program.
- Loading the address of the next out of sequence instruction in to the Program counter.
- Forcing the Processor to fetch the contents of this new location for its next instruction
- The new address is specified in the instruction.

JMP 2083

<an conditional jump to memory location 2083 for the next instruction>.



unconditional Jump

Jmp LAB 1

↑ to indicate the destination address of jump instruction

- use label to indicate jump during assembly language development

Absolute address	Instruction
2003	Jmp LAB 1
2083	LAB1: Destination Instruction

Symbolic Addressing

conditional Jump

JNZ LAB1

Jump if zero flag not set to LAB 1.

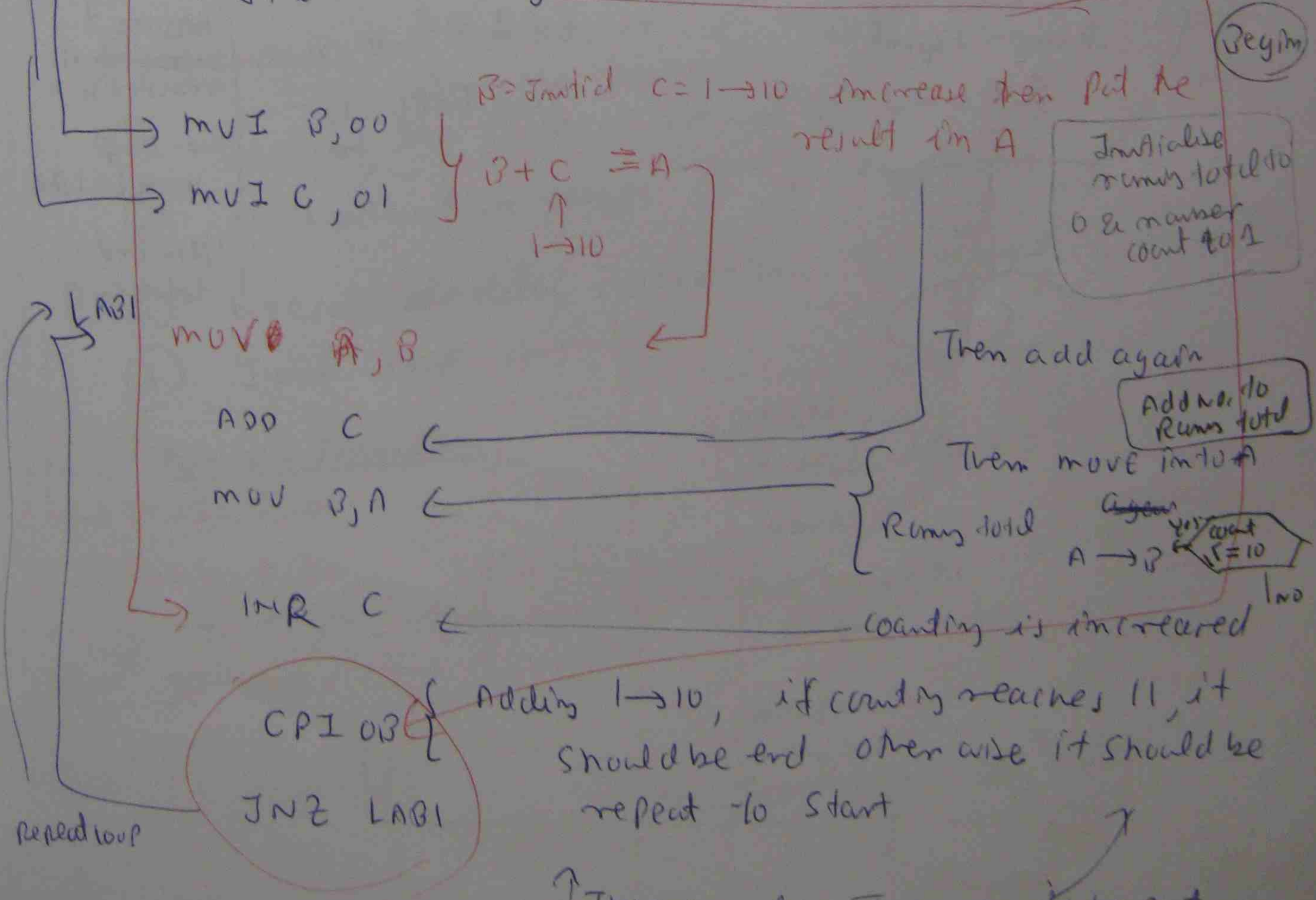
OP-code	condition	Flag status
JNZ (LAB1)	Jump if zero flag is not set to (LAB1)	Z = 0
JZ	Jump if zero flag is set to (LAB1)	Z = 1
JNC ( )	Jump if carry is zero	C = 0
JC	Jump if carry is 1	C = 1
JPO	Jump if parity is odd	P = 0
JPE	Jump if parity is even	P = 1
JP	Jump if S = +	S = 0
JM	Jump if S = −	S = 1

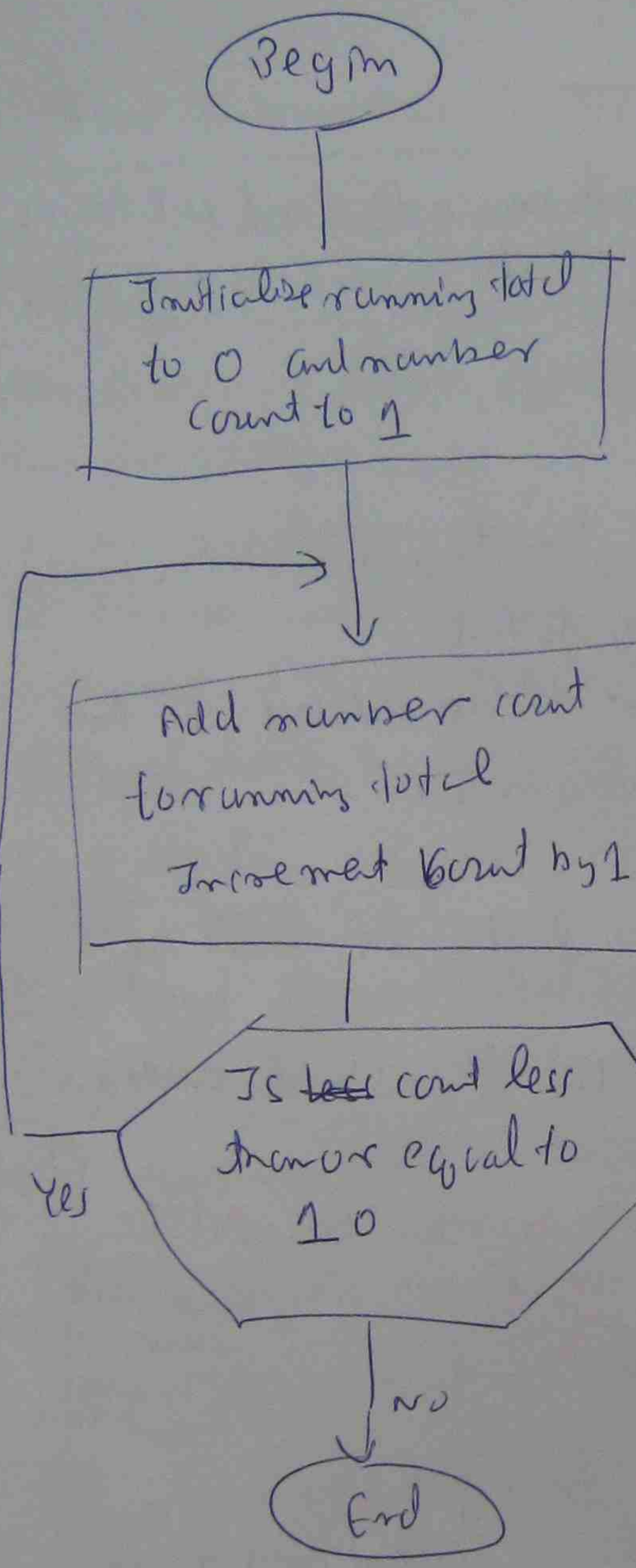
Flow Chart

A diagram which indicates the sequence of, and actions required in, a program and the points where branching is required.

Pb1 Construct the program to add together ten numbers 1→10

- B register contains the running total
- C register contains the number to be added to the running total. (~~Incs~~ start from 1)
- Increment by 1 & add
- If count is less than or equal to it will be repeated
- If the count is greater than 10 it will be ended.





	Assembly Instruction	Comment
	MVI B, 00	Initialise running total
	MVI C, 01	Initialise number count
LAB1	MOV A, B	Bring running total to A
	ADD C	Add number count
	MOV B, A	Restore running total in B
	INR C	Increment count
	MOV A, C	Bring count to A
	CPI 0B	Has count reached 11?
	JNZ LAB1	No- Jump back to LAB1
		Yes- End total in B

Imp loop ... Loop ... register

Jump Instruction

Time delay in the Program

- A common requirement when a microcomputer is interfaced to other equipment is to compute a time delay in the program.
- A micro computer-based road traffic light controller, for example, would need to compute a time delay to implement the sequencing of the light changes.
- Desired delay & loop count to be held in C-register.

Time delay NOP- No-operation instruction.

---

write a Program

① Load desired delay time parameter into C-register.
delay parameter is 02

② Clear A

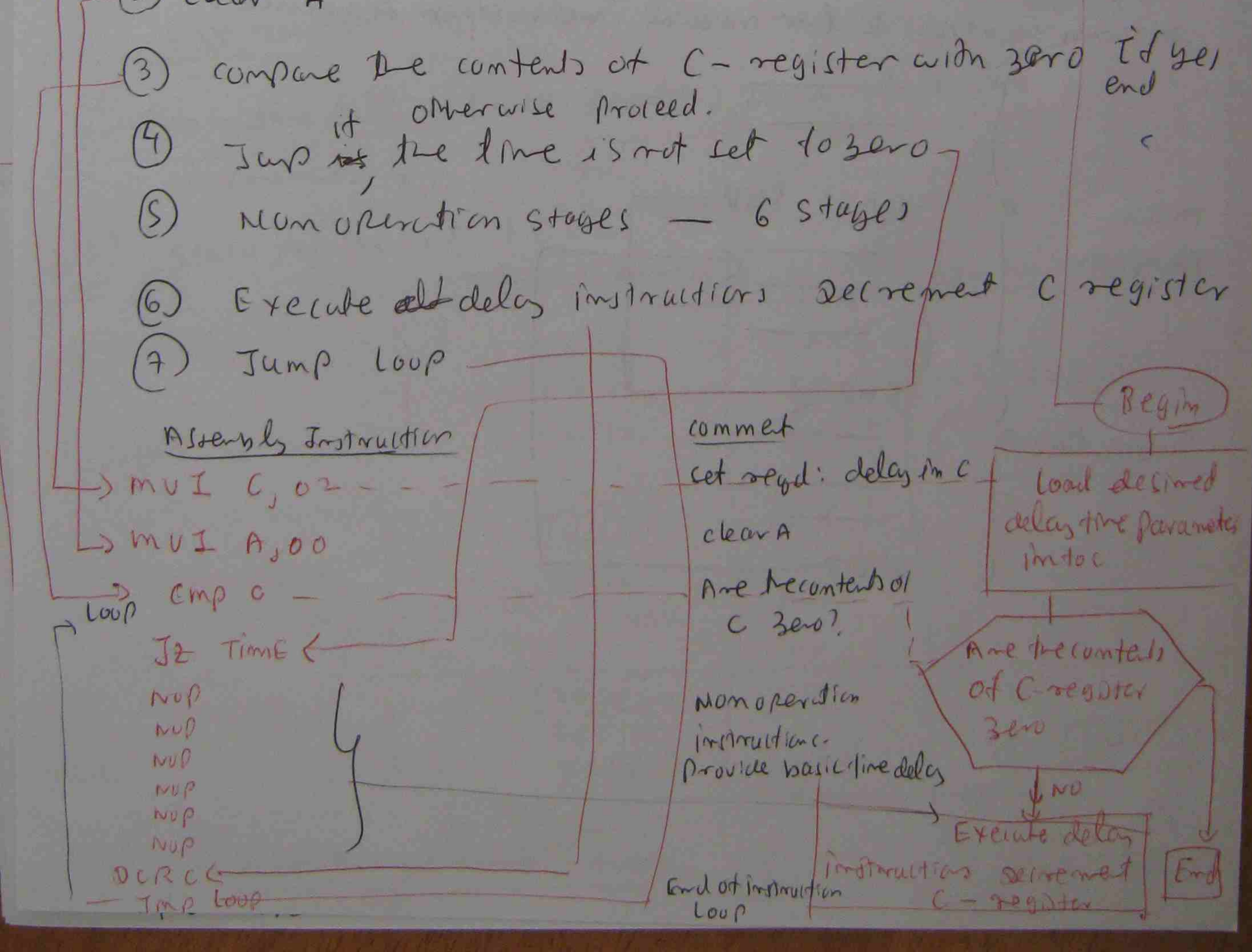
③ Compare the contents of C-register with zero If yes end

④ Jump if not, the time is not set to zero

⑤ Non operation stages — 6 stages

⑥ Execute delay instructions Decrement C register

⑦ Jump Loop



subroutines

pb 2 ← A short program to compute a time delay

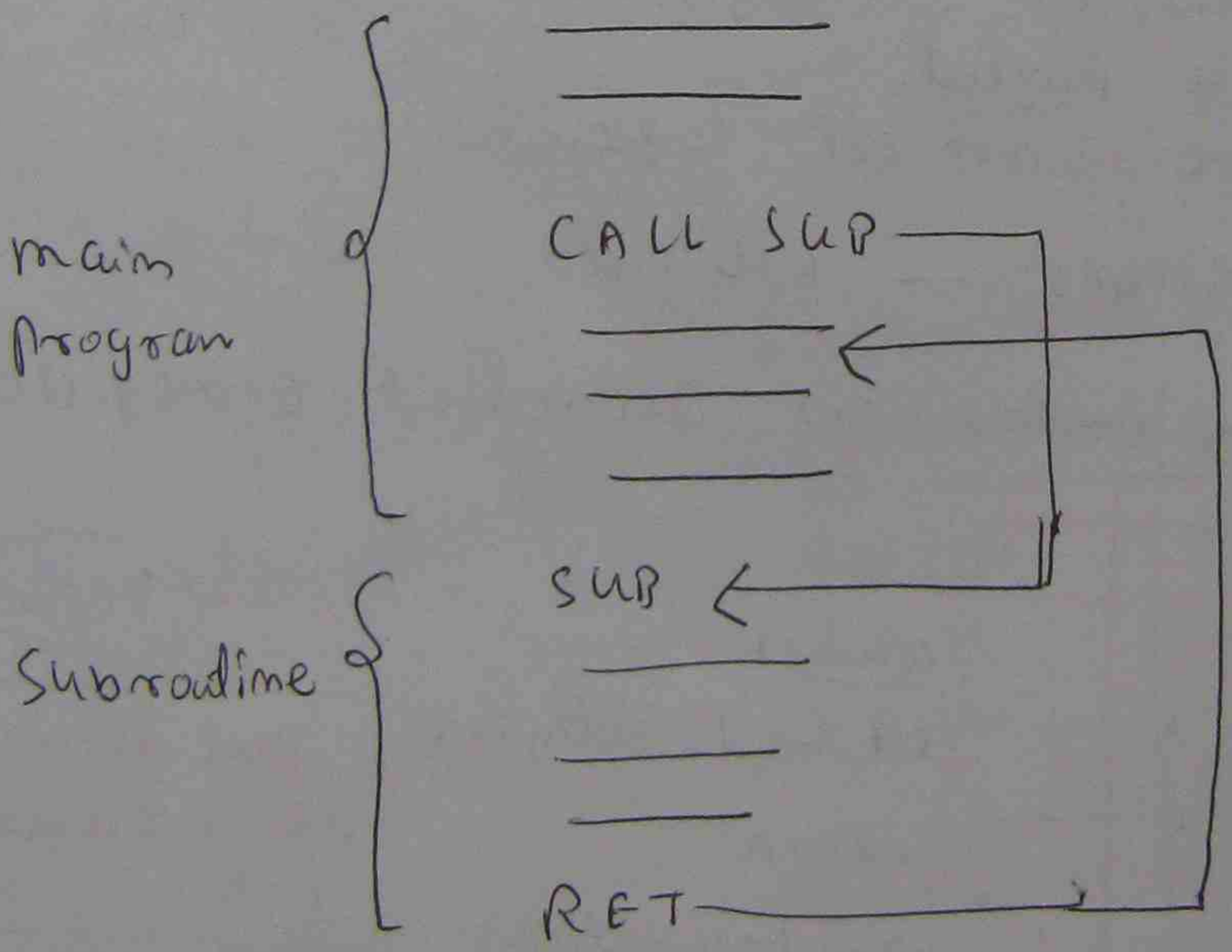
— This can be made into a subroutine by placing a RET instruction with the label TIME at the end of the program.

subroutine

Frequently within in a program, it may be necessary to perform a particular sub-task many times over.

~~It is highly desirable~~

subroutine is designed to perform the sub task and then return control to the main instruction sequence.



Stack

A last in first out queue which is implemented as a set of successive locations either within the processor itself or more usually, in the system memory.

Stack pointer register (SP)

it points to the address which currently holds the entry at the top of the stack, and consequently its contents change as each subroutine call and return instruction is executed.

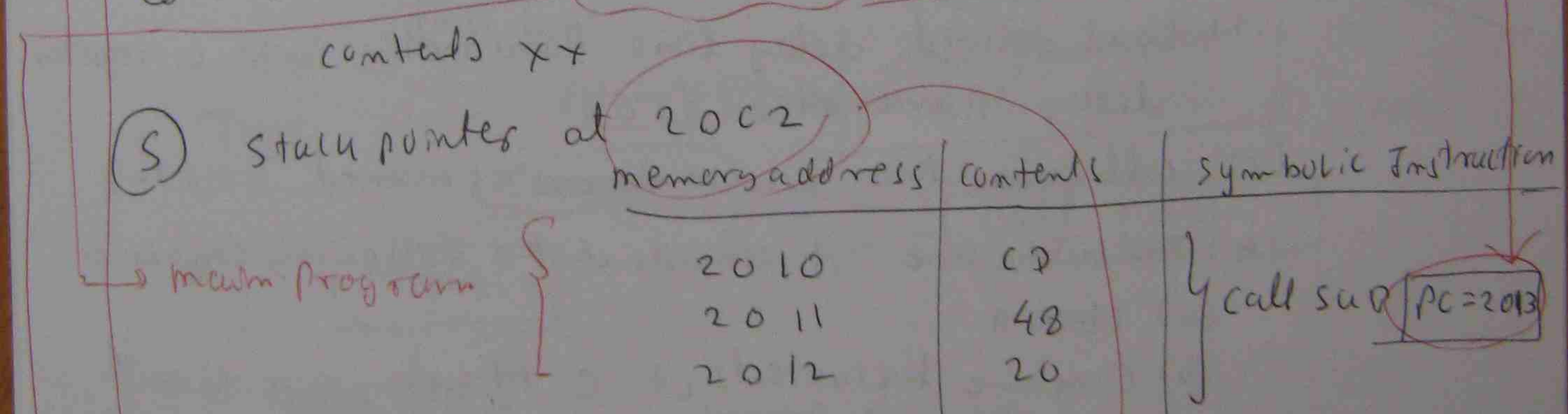
Ex ① main program memory address 2010 to 2012
contents CD, 48, 20 respectively

② Program counter PC - two bytes at 2013 call subroutine

③ Sub routine instruction 2048 → 2049 contents XX

④ Sub routine call (Stack ~~pointer~~) 2000 to 2002
contents XX

⑤ Stack pointer at 2002



Pb 3

memory Address		contents	symbolic instruction
main Program	2010	CP	CALL SUB PC = 2048
	2011	48	
	2012	20	
Subroutine	2048	XX	SUP
	2049	XX	
Stack	2010	13	SP = 2010
	2011	20	
	2012	XX	

Subroutine

Pb 3

(1) Initialize stack pointer at 2012
(2) Load desired delay time parameter into C-register
delay parameter is 02
(3) call subroutine which ~~is~~ ~~one~~ is named TIMEDLY
(4) In subroutine it consists of the following sequence
(a) Clear A
(b) Compare the contents of C if yes, goes to end
otherwise proceeds
(c) Jump if the time is not set to zero
(d) Non operation stages → 6 stages
(e) Execute delay instructions. Decrement C register
(f) Jump loop
(g) Return subroutine.

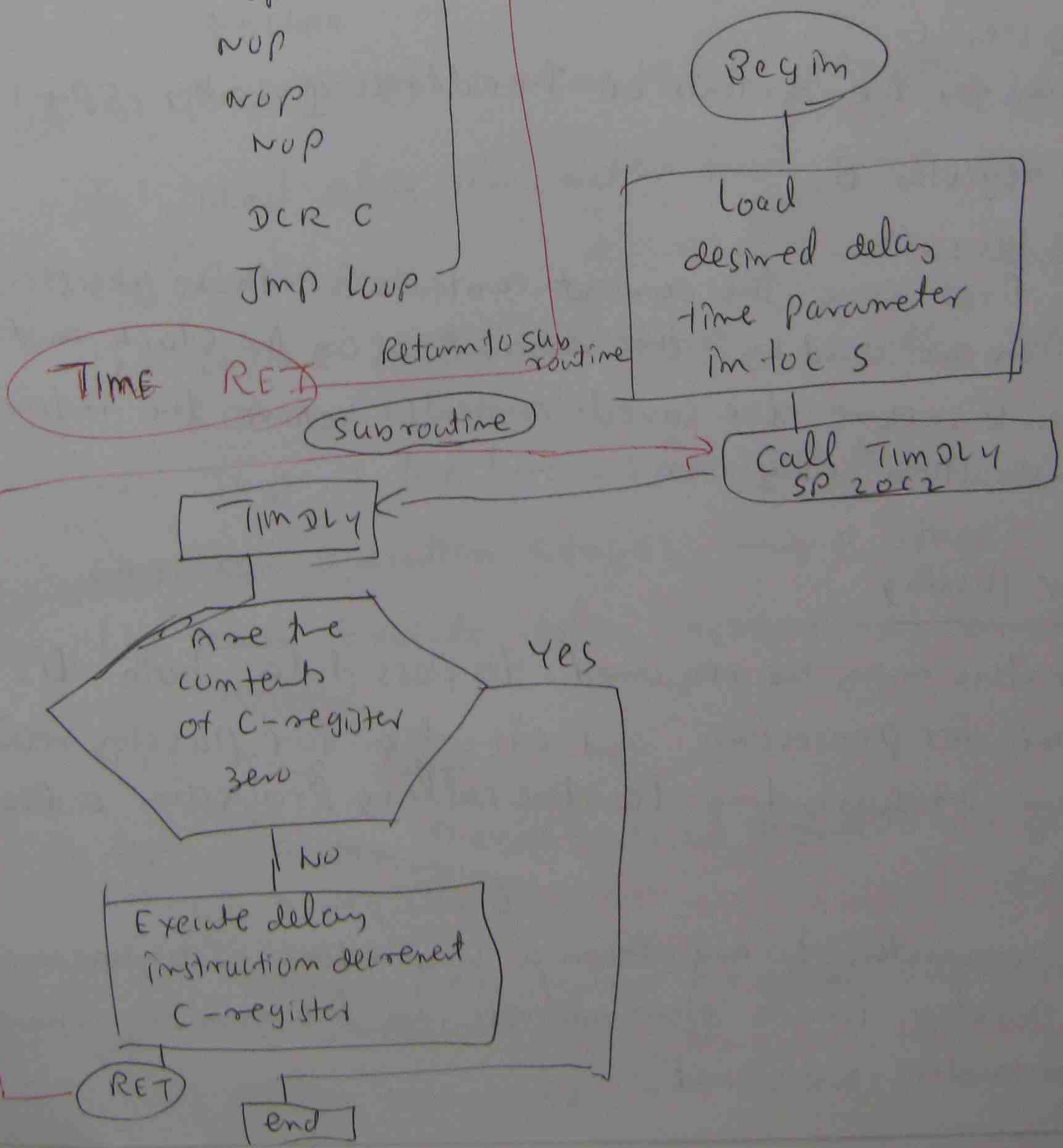
ORIGINAL PROGRAM

Additional for subroutine

```
LXI SP, 2002    <- Initialize Stack Pointer
MVI C, 02       Load TIMDLY Parameter
CALL TIMDLY     <- call subroutine
TIMDLY  MVI A, 00   Time delay subroutine
LOOP    CMP 0
        JZ TIME
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        DCR C
        JMP LOOP
TIME    RET     Return to sub routine
```

delay routine

Flow Chart



Stack operation

The stack ones alsobe used as a temporary deposit for thecontents of processor register.

PUSH PSW (Push Processor status word)

This push (saves) the combined 16 bit contents of the A-register and flags register on the top of the stack.

POP BC

(i) Transfer the contents of the address given by SP to register C
(ii) Transfer the contents of the address given by SP+1 to register B

Writing subroutine

* To first save the current contents of those registers which are used by the subroutine on the stack and then to restore the saved contents before the return instruction is given.

Parameter Passing

- Parameters may be required to pass data both (i) to subroutine for processing and (ii) also for passing results back from the subroutine to the calling program after processing.

- Passing parameters to and from a subroutine is by means of a pointer to the start address in the memory where the parameters are stored.

Pb 4

(1) Initialize stack pointer at 20C2

(2) Store delay parameter in memory location 2080

(3) Load delay time parameter into C register
delay parameter is 02

(4) call subroutine which is named TIMDLY

(5) In subroutine, it consists of the following sequence.
(a) Push (save) the combined 16 bit contents of the A register and flags register on the top of the stack
<Push PSW>

(b) Push (save) the contents of register C on stack
<Push B>

(c) Read delay parameter from memory C

(d) Clear A

(e) compare the contents of C, if yes, goes to end otherwise proceeds

(f) Jump if the time is not set to zero

(g) Nom operation stages ⟶ 6 stages

(h) ~~Decre~~ Execute delay instructions - Decrement C register

(i) Jump loop

(6) ~~(j)~~ Transfer the contents of the address given by
(a) sp to register B <Restore contents of register C>

(b) Transfer the contents of ~~the given~~ address given by
sp to register A <Restore contents of register A>

(c) Return subroutine

Original Program with subroutine

Address for parameter passing

```
LXI  SP, 20C2      Initialise stack pointer
LXI  H, 2080   }
MVI  M, 02     }   ——  Store delay parameters in memory location 2080
CALL TIMDLY        ——  call subroutine
```

MVI M, FF →

```
TIMDLY  PUSH PSW        Sa
        PUSH B          Sb      }  Save content of registers A2C on Stack
        MOV C, M        Subroutine 1
        MVI A, 00       Sc
LOOP    CMP C                   Read delay parameter from memory
        JZ  TIMC
        NOP
        NOP
        NOP
        NOP
        NOP
        DCR C
        JMP LOOP
TIMC    POP B           Ga      }  Restore contents of registers A2C from Stack
        POP PSW         Gb
        RET                     Return subroutine
```

LXI H, 2081
MVI M, FF
CALL TIMDLY2 →

Store delay parameter in memory location 2081

Remove.

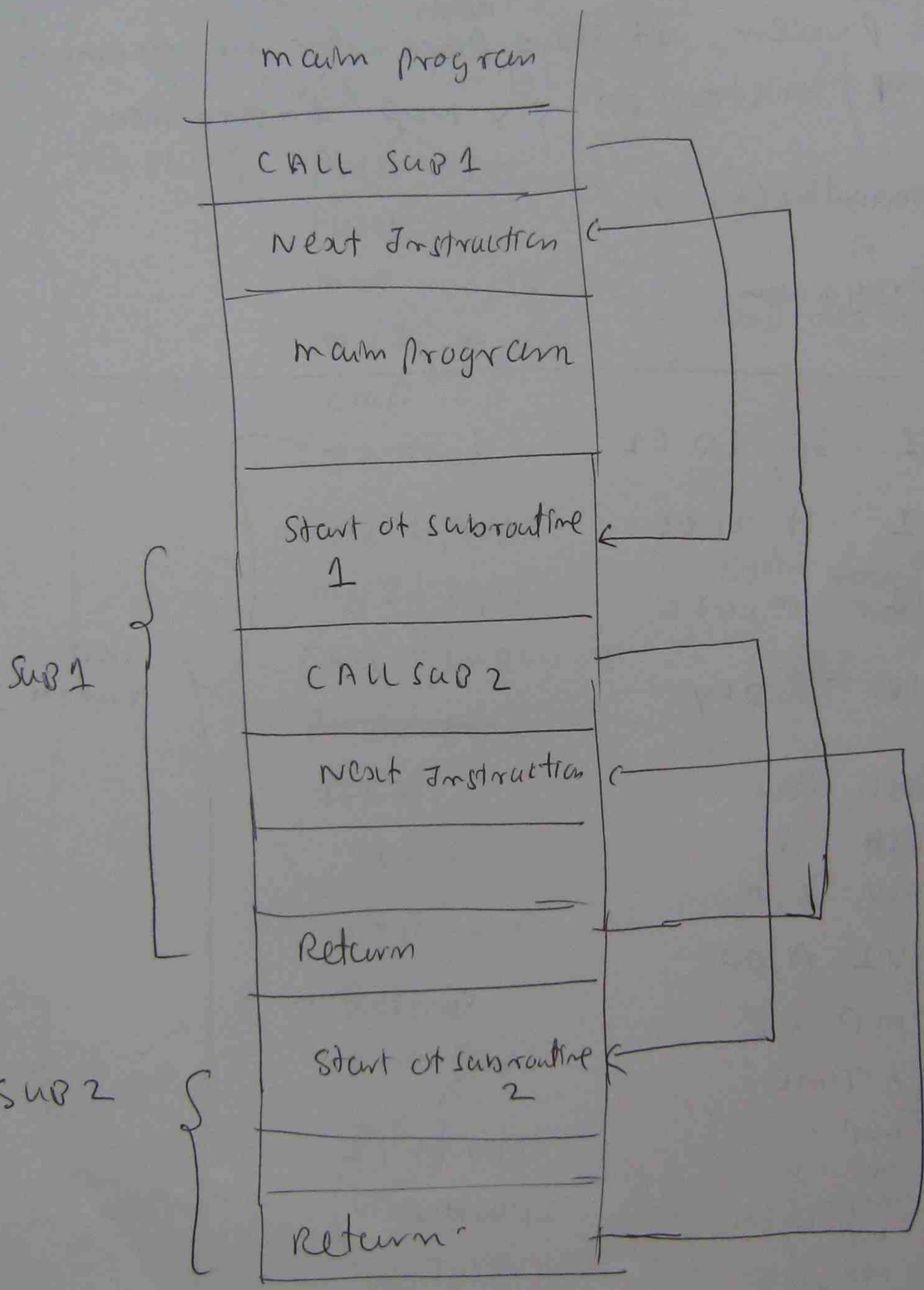
Delay routine

Sub routine 2:

```
TIMDLY2  PUSH PSW
         RET
```

Nested subroutine

A sub routine calls another subroutine with in itself



Pb 5 — modification of Pb 4

In above problem, it store delay parameter in memory location 2080 instead of six NOP instructions, as subroutine (2)

write the program.

```
        LXI   SP, 20C2
        LXI   H, 2080
        MVI   M, 02
        CALL  TIMDLY
TIMDLY  PUSH  PSW
        PUSH  B
        MOV   C, M
        MVI   A, 00
LOOP    CMP   C
        JZ    TIME
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        DCRC
        JMP   LOOP
TIME    POP   B
        POP   PSW
        RET
```

MVI M, ff

LXI H, 2081
MVI M, ff
CALL TIMDLY2

sub routine 1

Subroutine 2

```
TIMDLY2 PUSH PSW
        RET
```

main program

```
LXI  SP, 20C2      ← Initialise stack pointer
LXI  H, 2030       } ← store delay parameter in memory location 2030
MVI  M, FF         }
CALL TIMDLY1       ← call subroutine 1
```

Sub routine 1

```
TIMDLY1  PUSH PSW      } ← save contents of register A & C
         PUSH B        }
         MOV  C, M     } ← Read delay parameter from memory
         MVI  A, 00
LOOP     CMP  C
         JZ TIME
         LXI H, 2031   } ← store delay parameter in memory location 2031
         MVI M, FF     }
         CALL TIMDLY 2 ← call subroutine 2
         ~~PUSH PSW~~
         ~~RET~~
         DCR C
         Jmp LOOP
         POP B
         POP PSW
         RET
```

Sub routine 2

```
TIMDLY2  PUSH PSW   } ← As for TIMDLY1 except MVP instructions are replaced by subroutine 2 call
         RET
```

Exercise

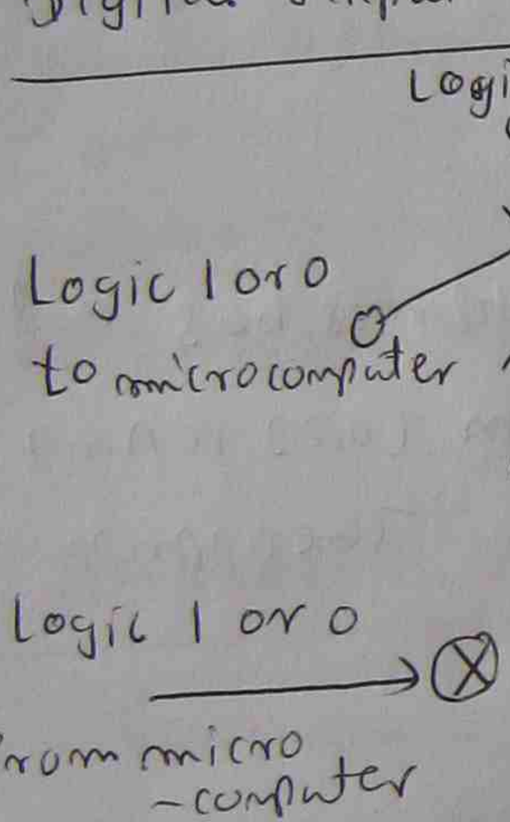
- Look at questions in Exercise 5.1 to 5.7
- Study the answers provided for exercise 5.1 to 5.7
- Observe the way to find the solution
- Take practice with microprocessor training software

Week ⑥

Digital Input and output

- A microcomputer is basically a digital component that can examine digital input signals and perform functions as a consequence of these inputs to yield digital output signals.
- The external devices outside the micro computer produce (or) accept signals which are not necessarily digital in nature, special interface circuitry is often required to transform these external signals into a form suitable for the microcomputer

Digital Input and output

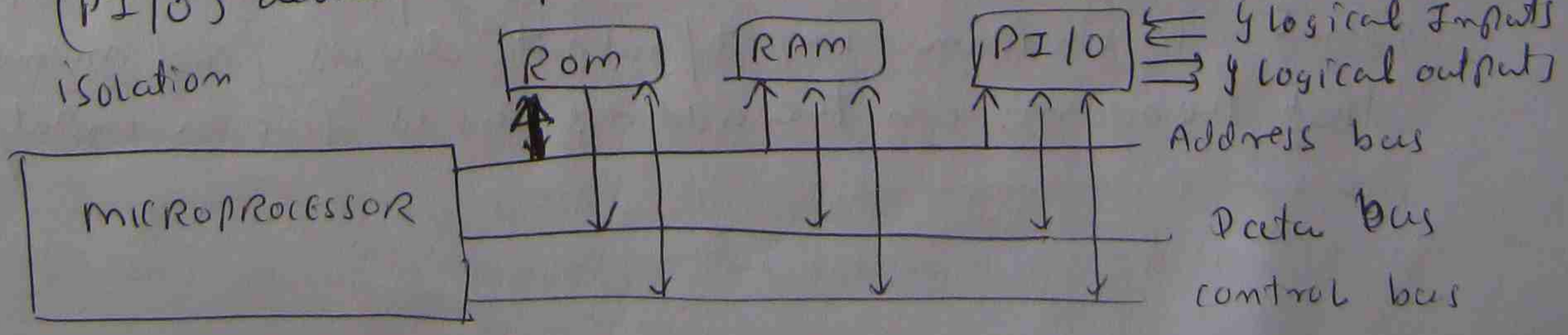


A simple digital input to a microcomputer can be produced by a single pole switch.

Logic 0, 1 depends on switch position

LED ON for logical 1
LED off for logical 0

- The information intended for an output indicator must be latched by a suitable circuit.
- The processor can then send data to the output latching device which captures the data at the appropriate time determined by bus control signals and then provides a continuous output until new data is sent to it.
- The system incorporates a programmable input/output (PI/O) device to provide the necessary latching and isolation

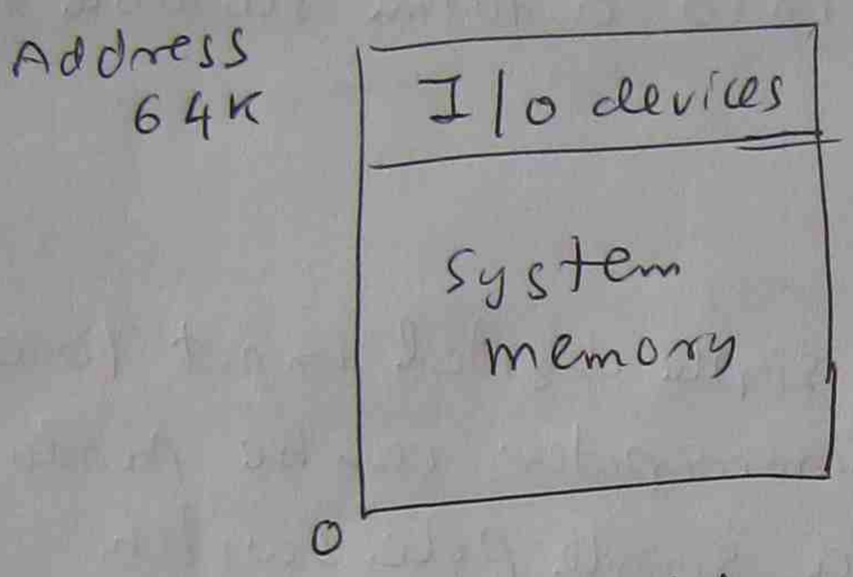


Data transfer of input/output data between a microcomputer bus and input/output device

---

- memory mapped input/output
- Programmed input/output.

i memory mapped input/output

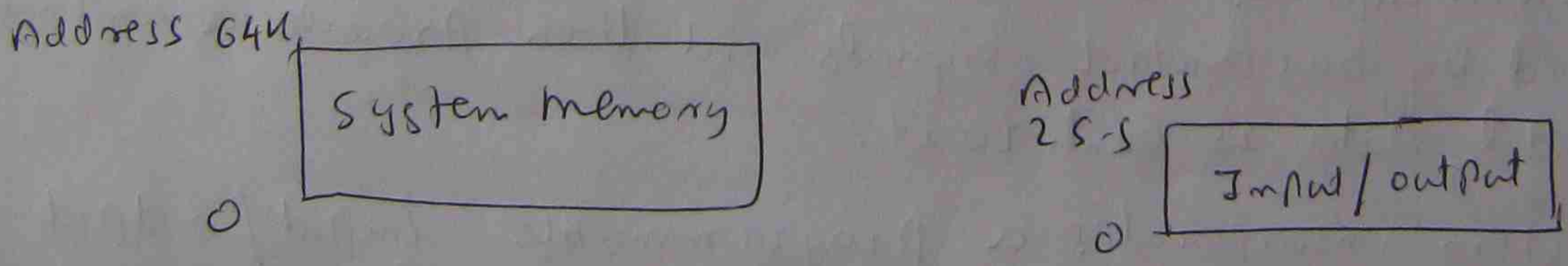


- The same instructions are used for both memory and input/output data transfer

The appropriate address is output on the address bus and recognised either by a memory device (ROM (or) RAM) or input/output device (PI/O) or port. The appropriate data is transferred on data bus.

ii Programmed input/output

- Input/output data transfers are accomplished by means of special instructions executed by the Processor - IN and OUT for the Intel 8085.



- microprocessor generates an input/output request signal to inform input/output devices (and memory) that the address on the address bus is for an input/output device.

iii Programmable input/output

Digital input and output in most microprocessors is controlled by Programmable input/output devices and Programmed input/output is normally used. A PI/O device can control a number of individual input and output lines. These are normally grouped into a number of ports, each comprised of eight lines which may be Programmed to operate either as inputs or as outputs.

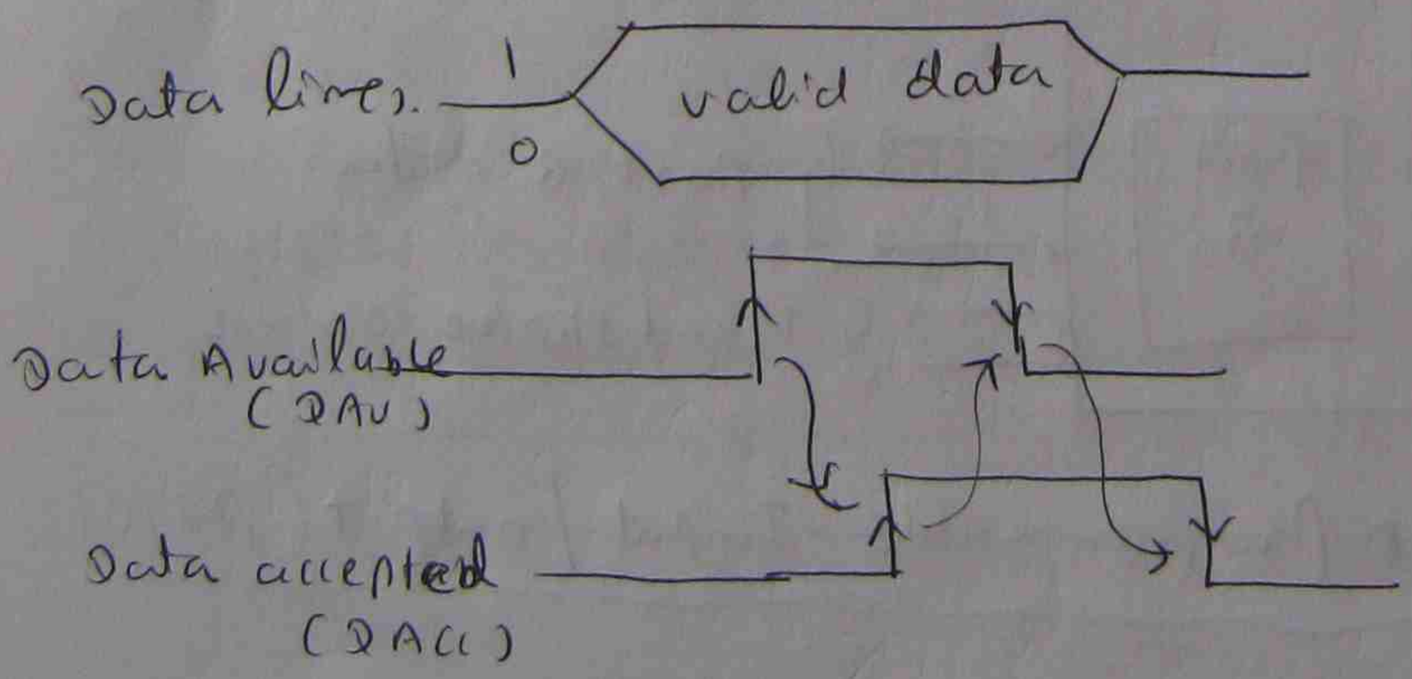
Steps

- write appropriate command information into a specific addressable registers within the device when the system is being initialised
- Data is read from (or) written to a port.

Handshake control

Synchronise the transfer of data between PI/O and external device and consequently most PI/Os provide control lines for this function.

Hand shake – Typical transfer sequence.

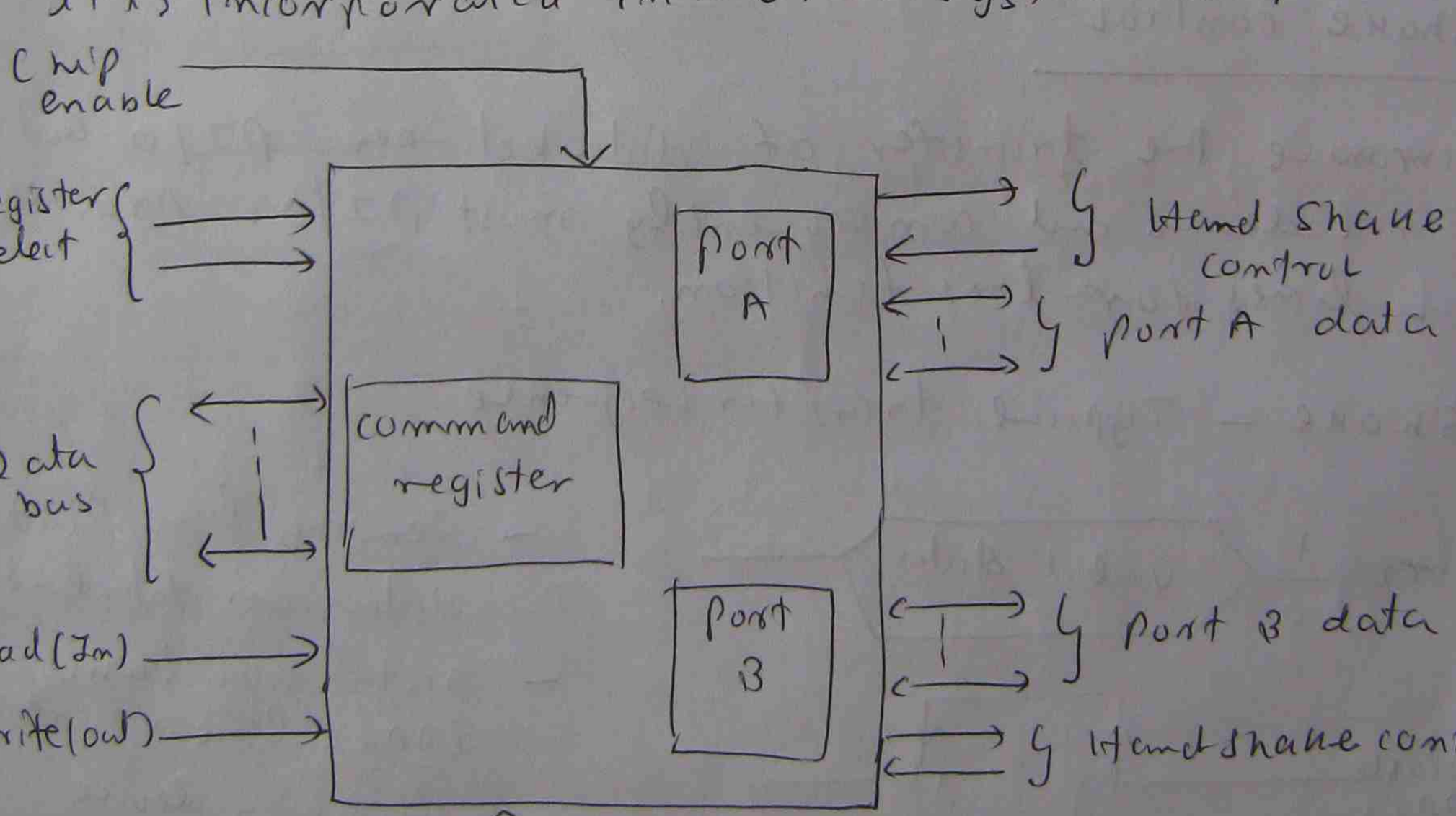


- Sending line Places data on data line
- Data available (DAV) line is set
- Receiving device detects the setting or DAV line
- Then responds by setting the data accepted "DACC" line
- Sending device interprets the setting or DACC line

- Receiver detects that DAV line has been reset
- Reset DACK line to permit further data transfer.
- There is a chip enable input on all the devices which are connected to the microprocessor bus.
  - RAMs, ROMs, PIOs - They are used to ensure that only one device responds to each data transfer on the bus.
- Port A, B select the appropriate register within PIO itself - command.

Port Initialisation

Some microprocessor, each programmable input/output device is a separate integrated circuit but in others. it is incorporated into other system components.



A Programmable Input/output (PIO)

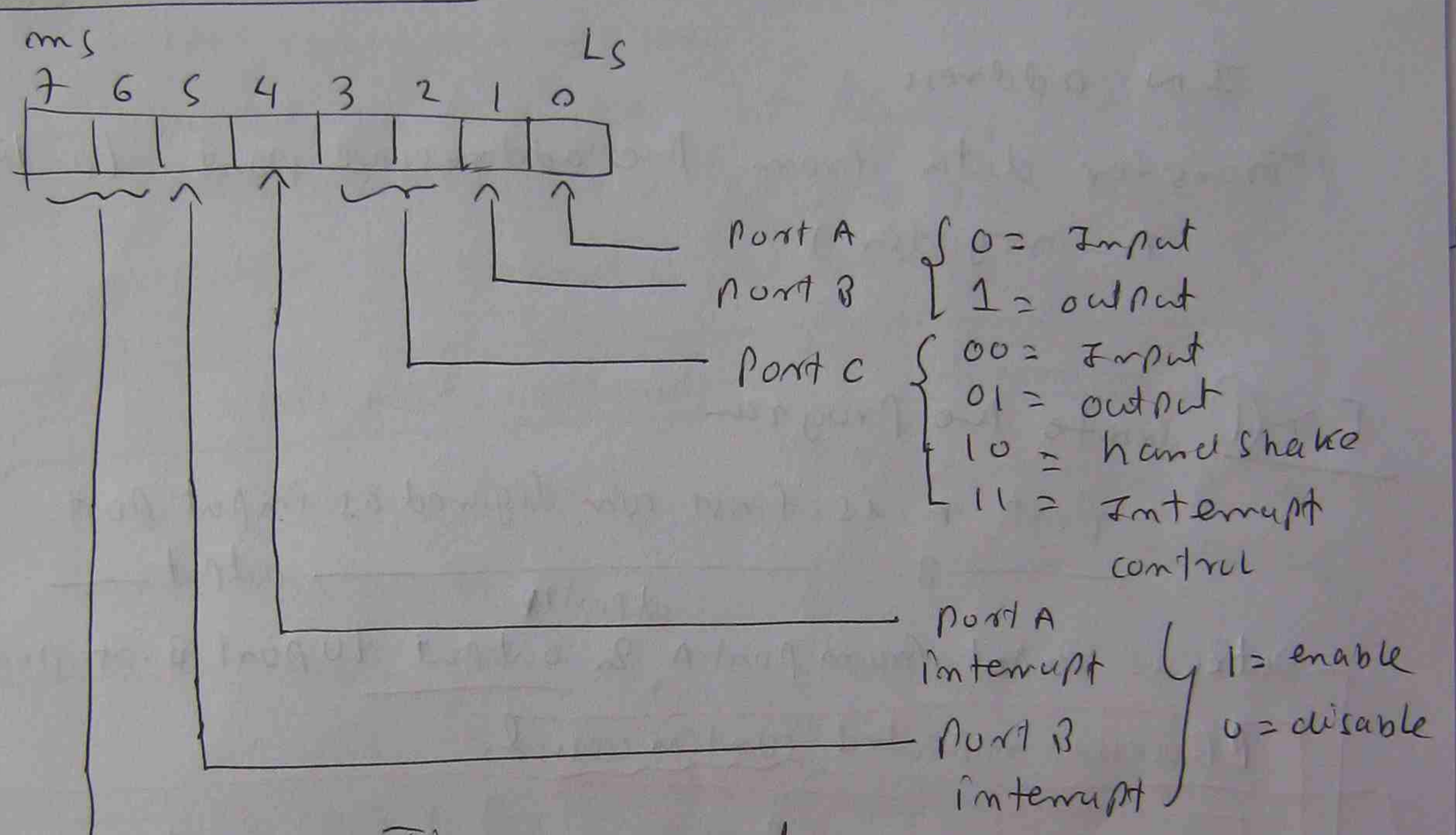
Command information

- Initialise the port, Timer and input/output ports are programmed together

OUT address

(Transfer the contents of the Processor A register to the addressed input / output device)

8155 command byte

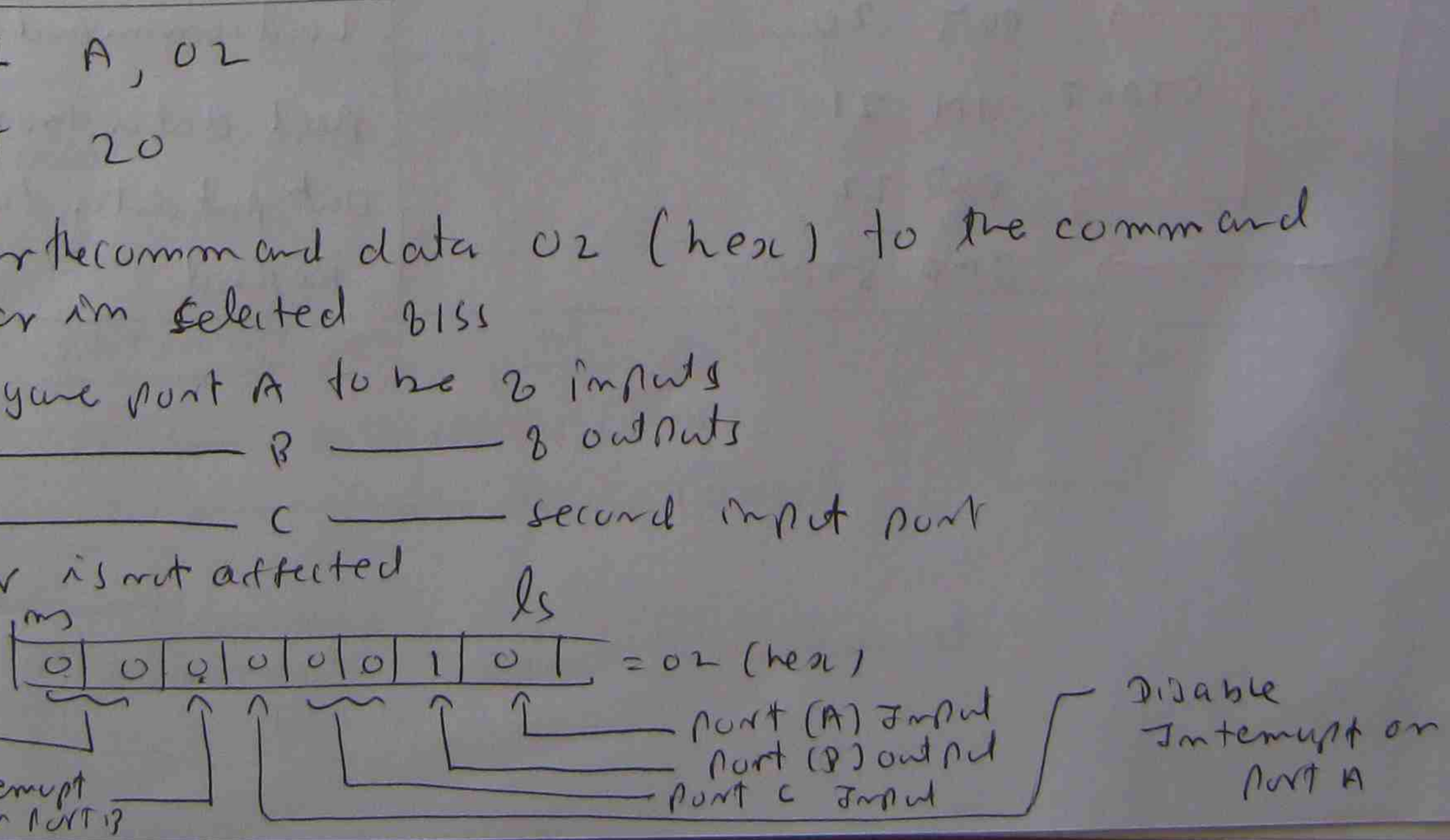


MVI A, 02

OUT 20

- Transfer the command data 02 (hex) to the command register in selected 8155
- configure port A to be 2 inputs
- ——— B ——— 8 outputs
- ——— C ——— second input port

Timer is not affected



Address (hex)	Port / Register
20	command / status register
21	Port A
22	Port B
23	Port C

Typical Port / register addressing

IN address
(Transfer data from the addressed port to the A - register)

Ex① write the Program
- Port A is first configured as input port
- B ——— output ———
- Data is input from port A & output to port B. at port 22
- Process is repeated continuously

out port 22

Assembly Instruction		comments
	MVI A, 02	Load command byte in A
	OUT 20	Load command register
START	IN 21	Read data from Port A
	OUT 22	out put data to Port B
	JMP START	Repeat

Ex2 Initialize port A, B, C as output ports.
then output decimal number 24937 in BCD form.

then
- load command byte in A
- Load command register
- output 24 to port A at line 21
- output 93 to port B at line 22
- output 07 to port C at line 23

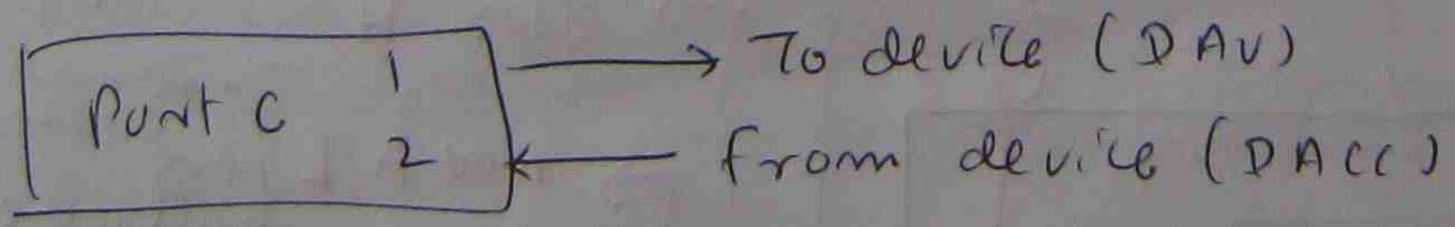
Assembly instruction	comments
MVI A, 07	Load command byte in A
OUT 20	Load command register
MVI A, 24	output 24 to port A
OUT 21	
MVI A, 93	output 93 to port B
OUT 22	
MVI A, 07	output 07 to port C
OUT 23	

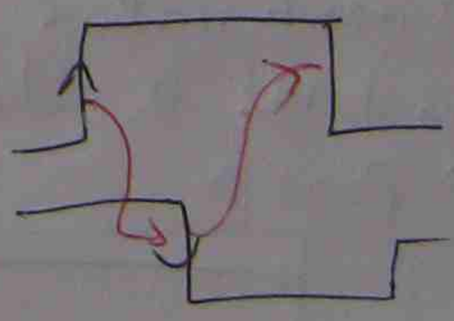
Port A
0010 0100
2 4

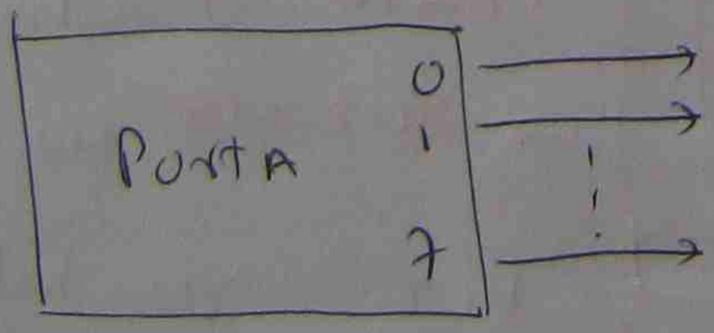
Port B
1001 0011
9 3

Port C
xxxx 0111
7

Handshake control







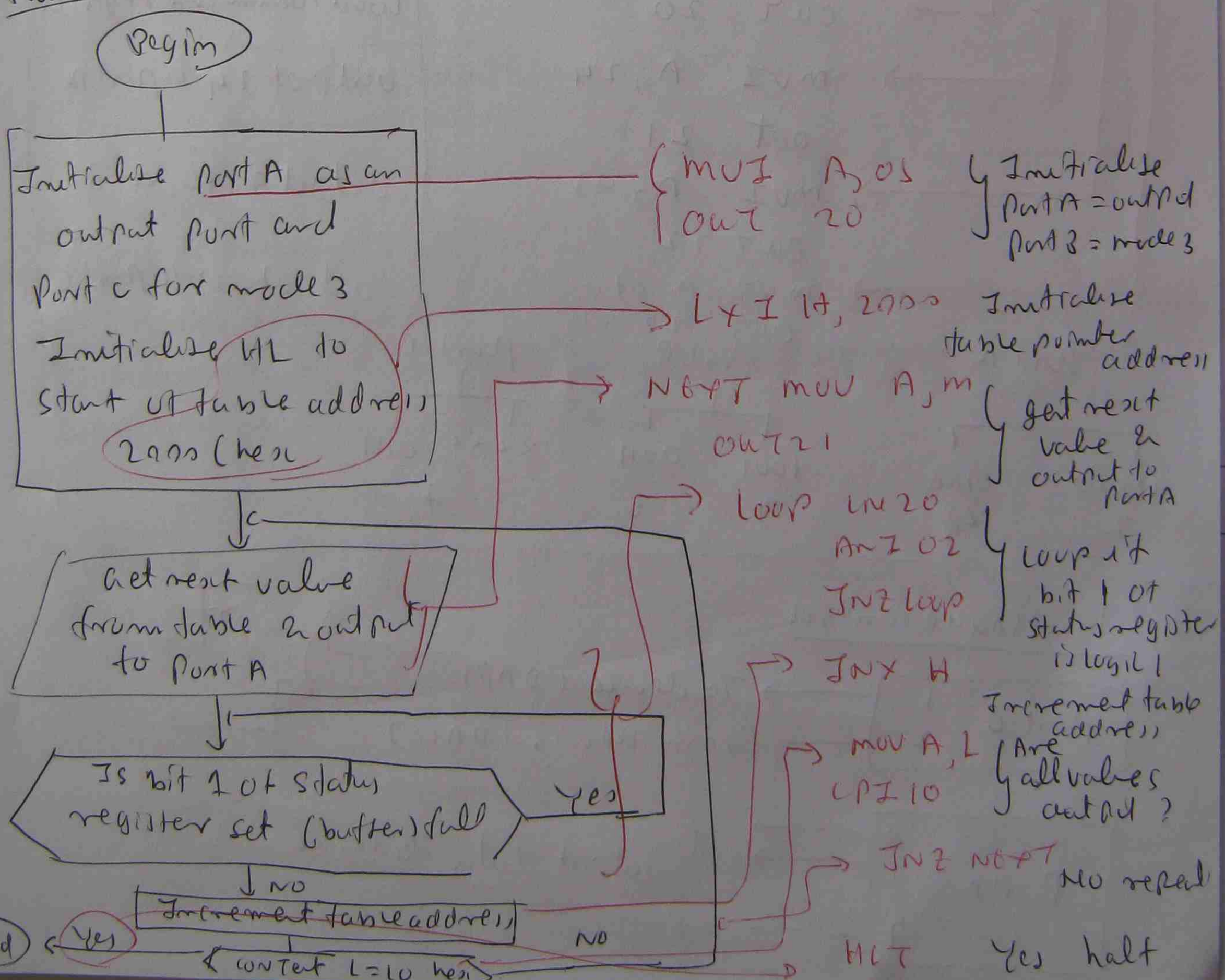
3 & 4M

E+3

Handshake control lines of Port C to output 16 values from a table in memory - starting address 2000 (hex) to an external device connected to Port A.

- Port C has been initialised to operate in mode 3 (hand shake mode)
- After data has been output to Port A bit 1 of Port C will automatically go to logic 1, indicating to the external device that new data available.

Flow

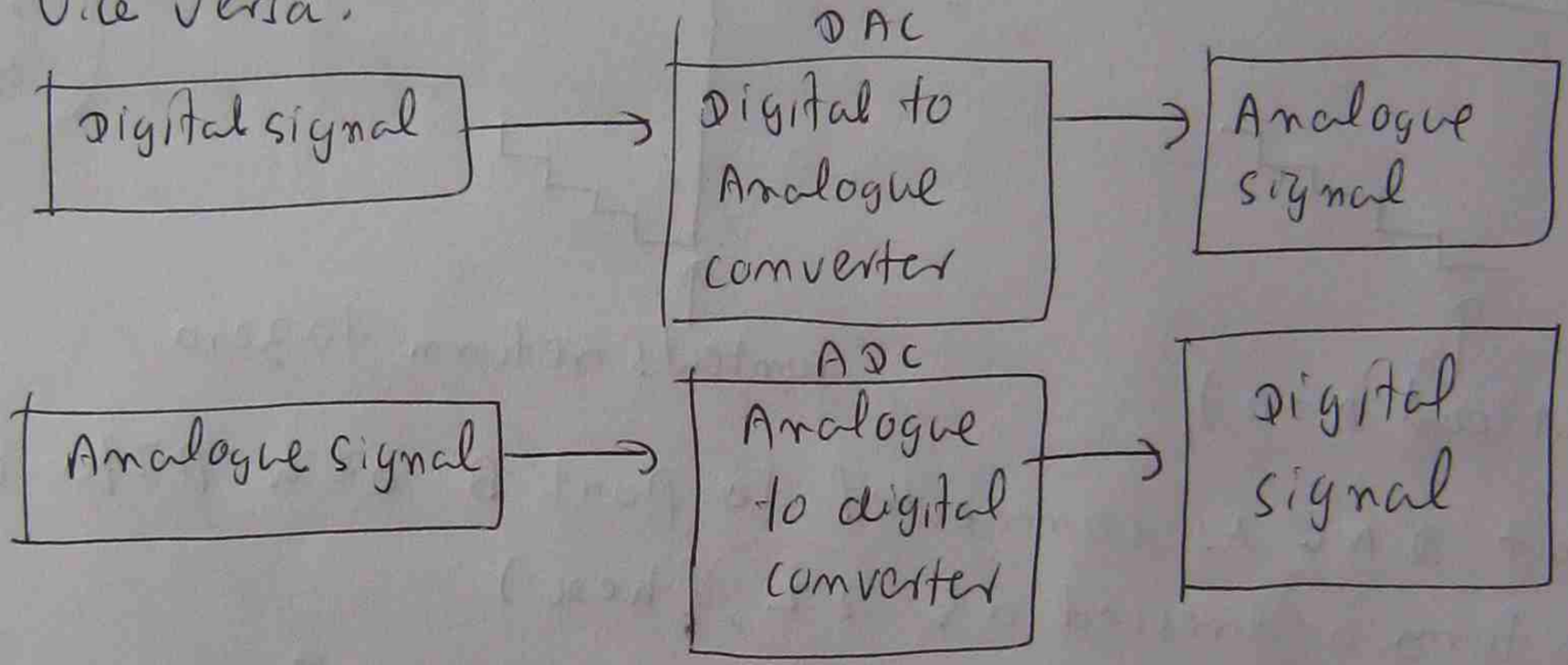


WEEK (7) ANALOG INPUT/OUTPUT & INTERRUPT

Input data - varying analogue signal, output voltage from a temperature transducer.

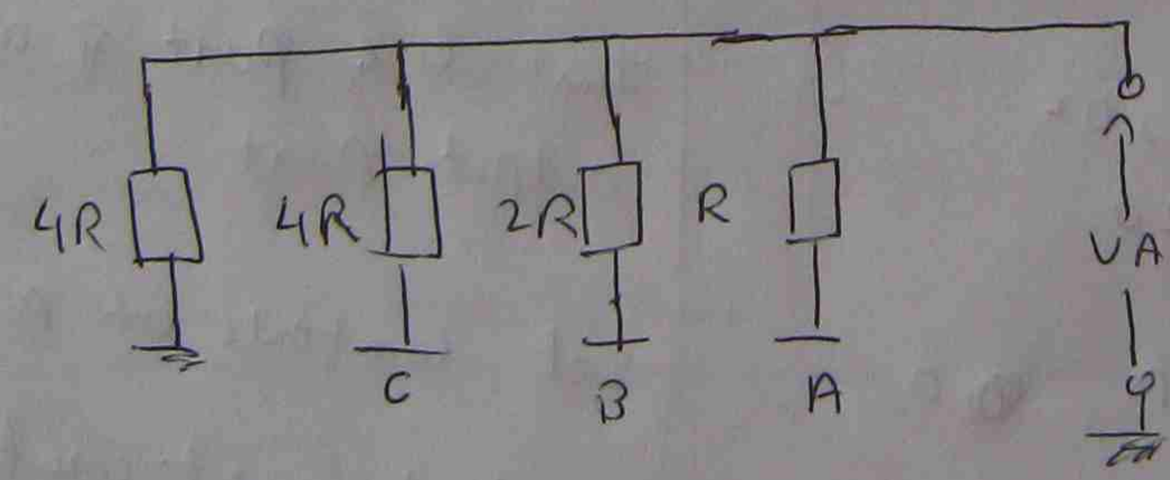
Output data - The output data from the microprocessor is often required in analogue form. For example to drive a motor.

It is necessary to have additional interface circuitry between the input/output ports of the microprocessor and the controlled peripheral devices, both to convert analogue signals in to digital form and vice versa.

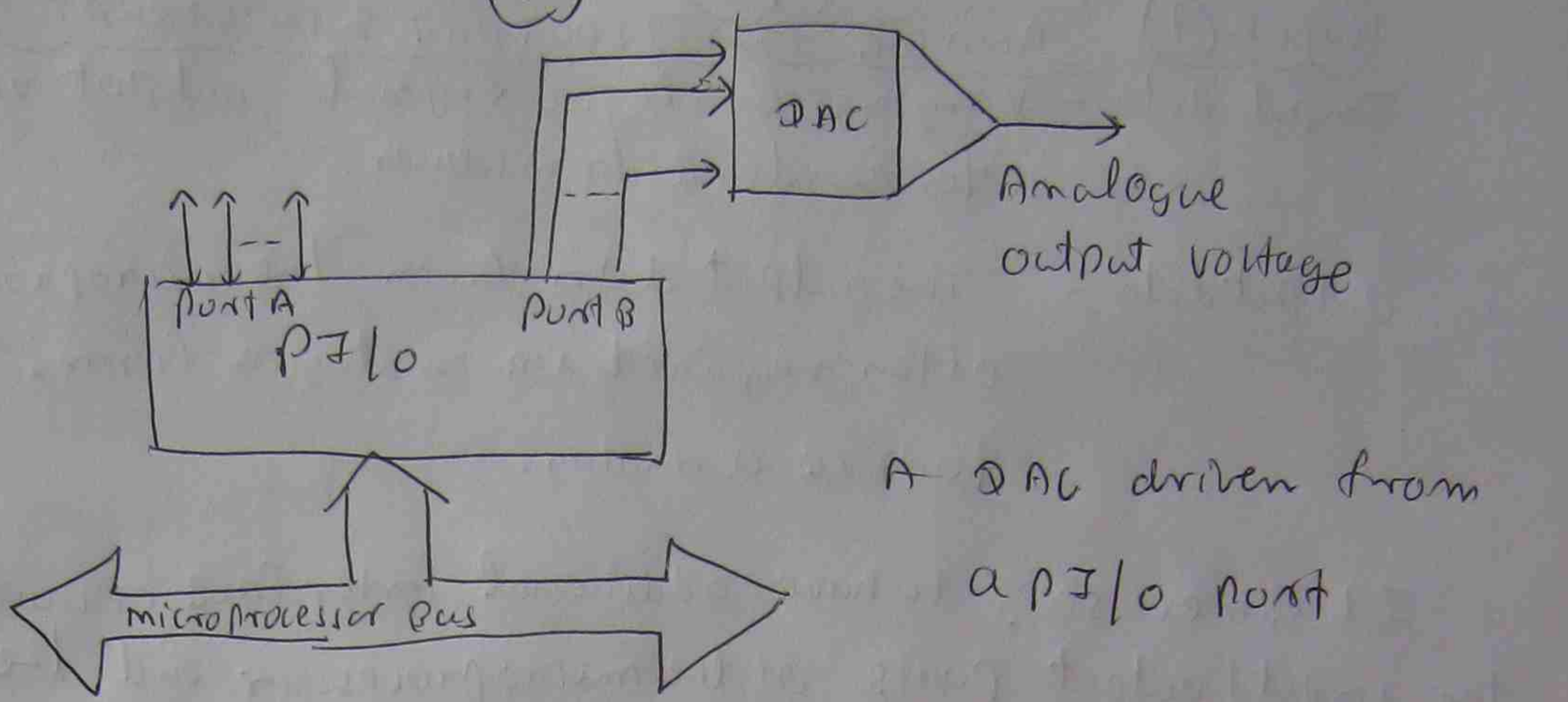


Digital to analogue conversion

A digital number can be converted to an analogue voltage by selectively adding voltages which are proportional to the weighting of each binary digit.

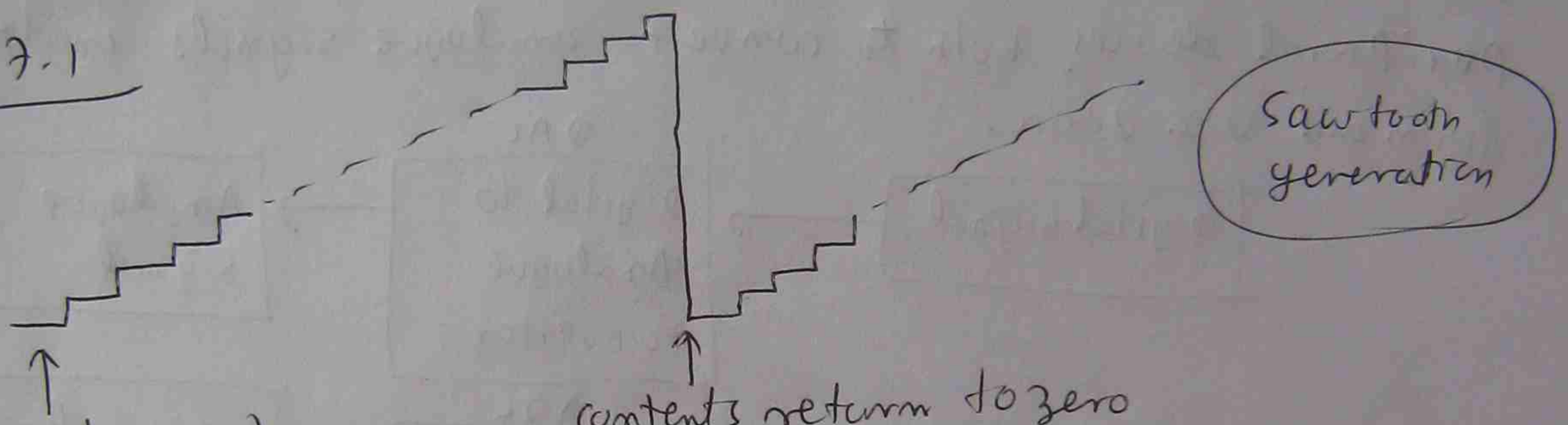


A	B	C	VA
0	0	0	0
0	0	1	V/8
0	1	0	V/4
0	1	1	3V/8
1	0	0	V/2
1	0	1	5V/8
1	1	0	3V/4
1	1	1	7V/8



A DAC driven from a PIO port

Ex 7.1



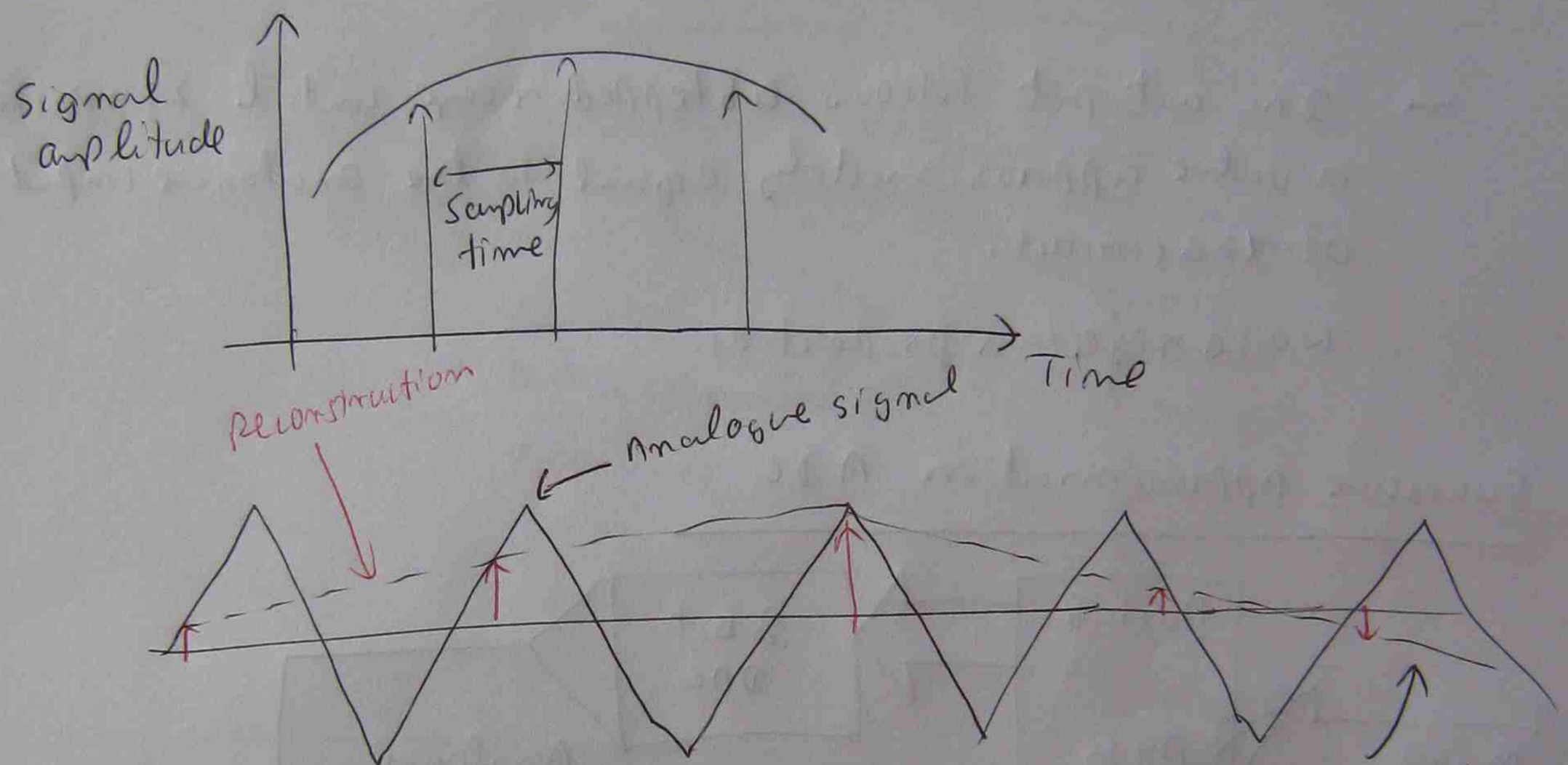
8 bit DAC is connected to Port B of a PIO which is in turn addressed as 22 (hex)

A saw tooth wave form is then readily generated by using the A-register as a counter. & outputting its contents after each increment. Write a program

Assembly Instructions	comments
MVI A, 02 OUT 20	Initialize Port B as an output port
MVI A, 00	Set contents of A to zero
COUNT OUT 22	Output current count
INR A	Increment count
JMP COUNT	Loop back

Analogue to digital conversion

The conversion of an analogue signal to a digital number implies a process of signal sampling. A digital number can only accurately represent a changing analogue signal for a short period of time.

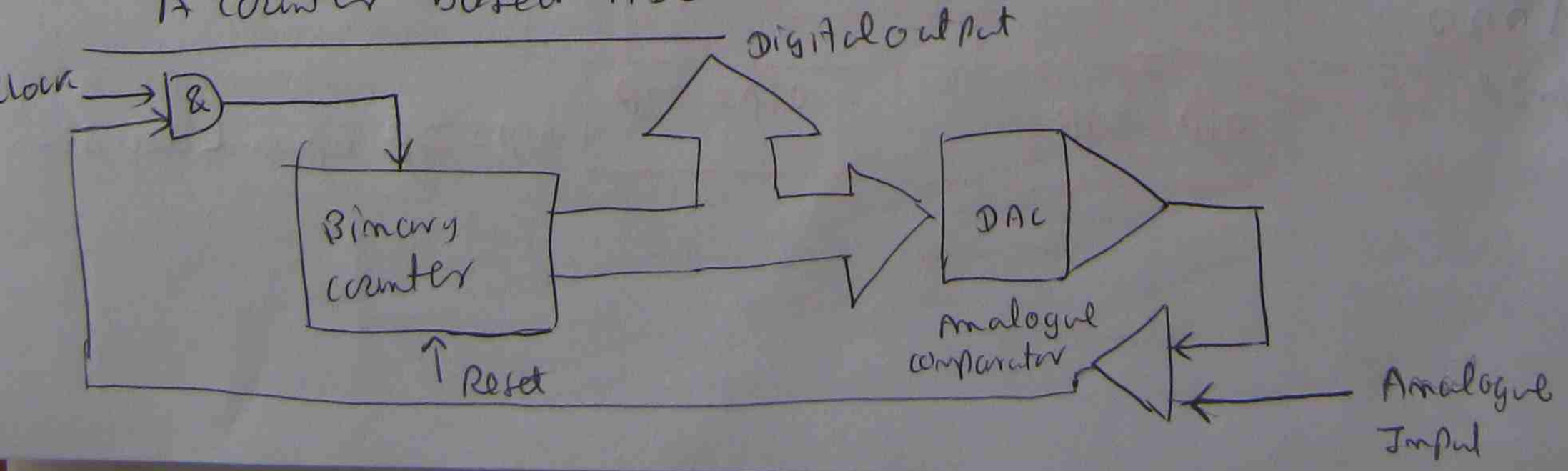


The shannon sampling theorem

It is possible to severely distort the digital representation of an analogue signal by sampling it too infrequently.

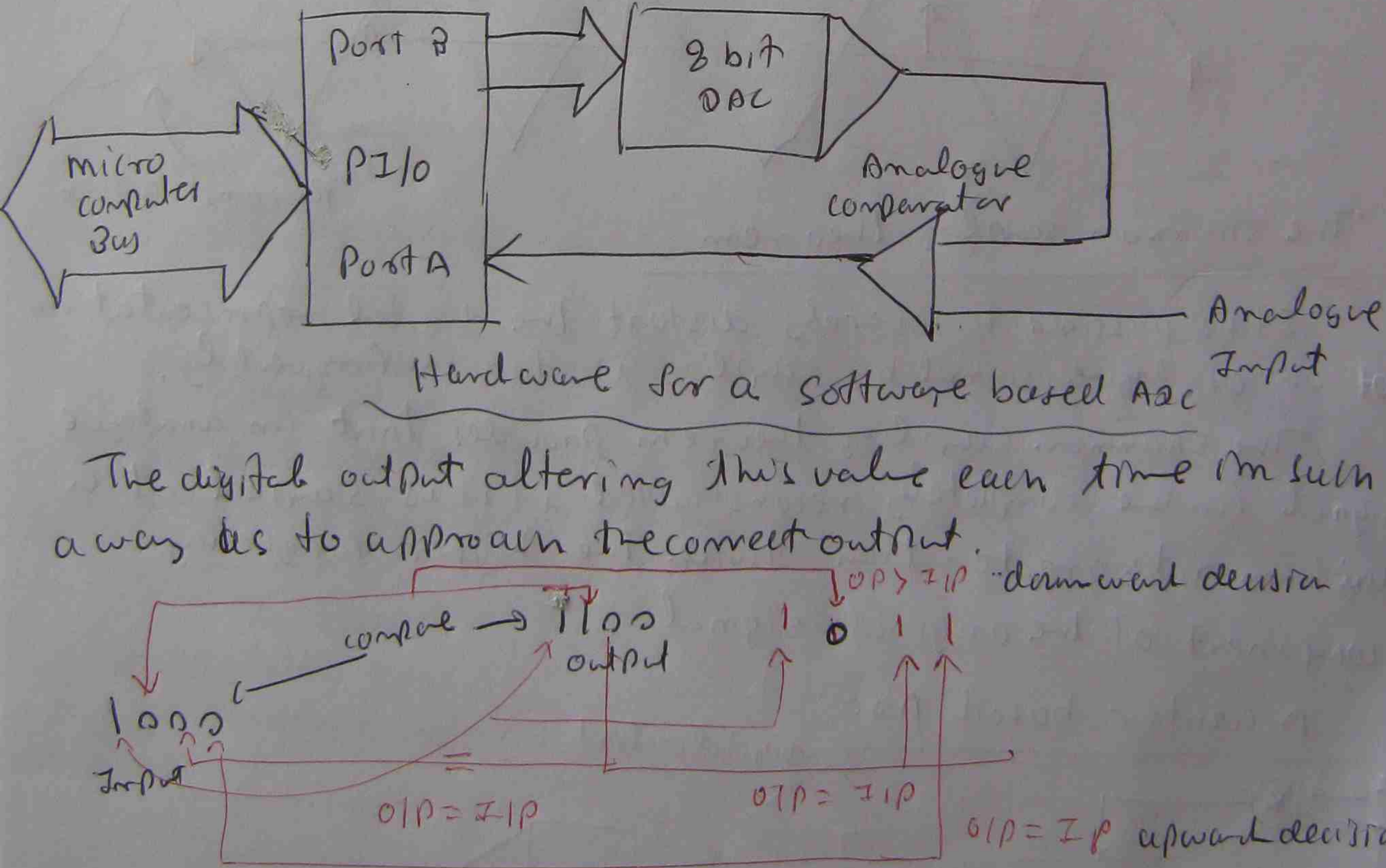
The shannon sampling theorem provides that an analogue signal can be completely reconstructed if it is sampled at a uniform rate greater than twice the higher frequency component of the original signal.

A counter based ADC



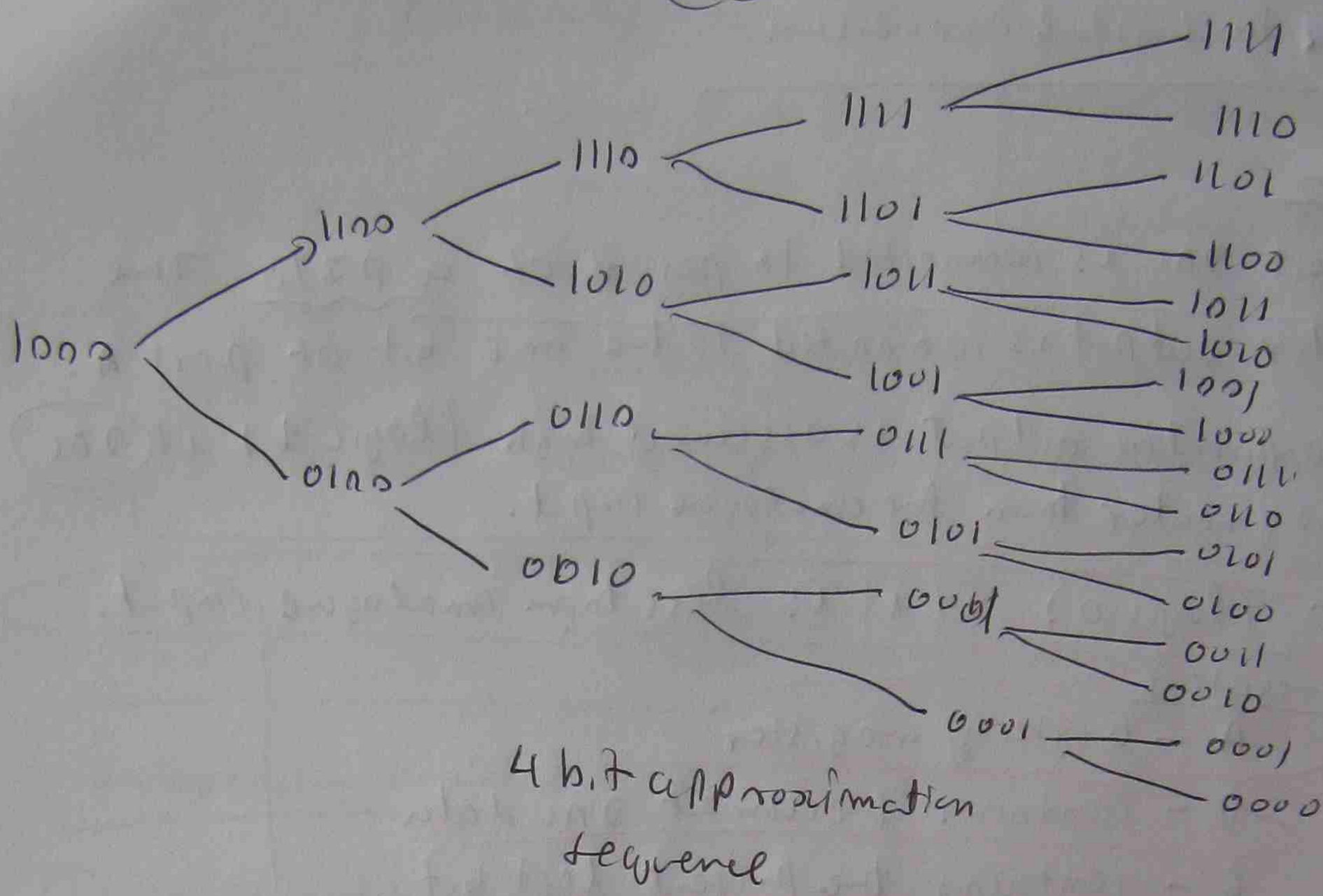
- The counter is initially reset at the start of a conversion.
- until DAC output just exceeds the analogue input, counter clock pulses are then enabled, causing the comparator to further counter clock pulses.
- Digital representation of the analogue input is then binary output of the counter
- DAC out put follows a stepped ramp until it reaches a value approximately equal to the analogue input of the circuit.
- Noise rejection properties.

Successive Approximation ADC

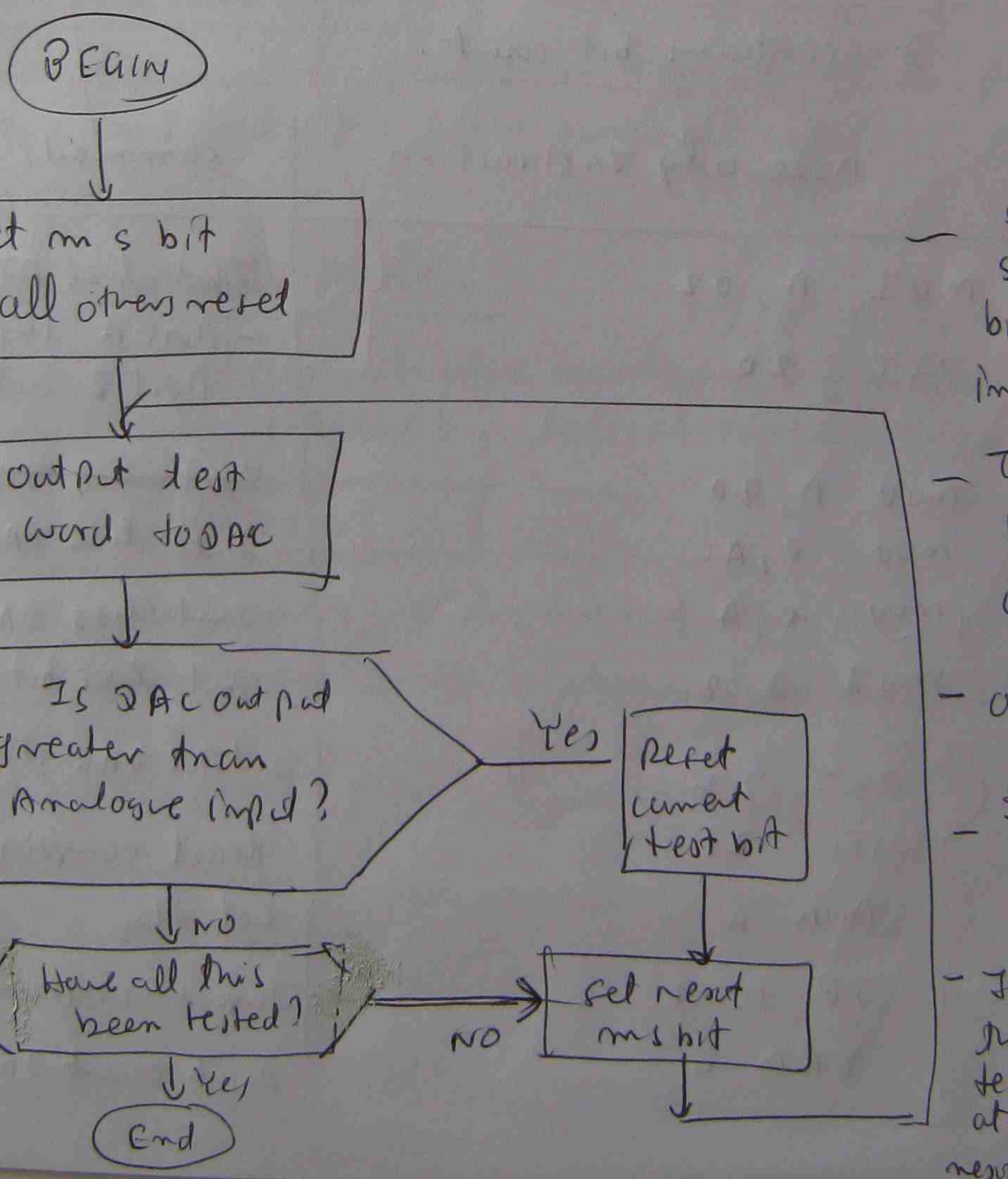


Hardware for a software based ADC

The digital output altering this value each time in such a way as to approach the correct output.



4 bit approximation sequence



Analogue to digital conversion

Ex 7-2

8 bit DAC is connected to port B of a PIO. The comparator output is connected to the ms bit of port A.

The comparator output is assumed high (logic 1) if DAC output is greater than the analogue input.

Low (logic 0) if it is less than analogue input.

Processor registers

A - working register

B - contains the current DAC data

C - contains the present test bit

D - contains bit count.

Assembly Instruction		comments
MVI	A, 02	Initialise PIO
OUT	20	- Port A Input
		- Port B output
MOV	A, B0	~~Initialise DAC data~~
MOV	B, A	Initialize DAC ~~test data~~ med
MOV	C, A	Initialise DAC test bit
MVI	D, 08	Initialise bit count
REPEAT: OUT	22	output DAC data
IN	21	Read comparator output
ANA	A	Set flag
JP	CONT	Jump if comparator o/p = 0
XRA	C	Reset count DAC test bit

Assembly Instruction		comments
COMZ	MOV B, A	save DAC data
	MOV A, C	update DAC test bit
	RAR	
	MOV C, A	
	ORA B	set next ms bit of DAC data
	DCR D	Decrement bit count
	JNZ REPEAT	Jump if not zero

Interfacing analogue devices

(1) Sample and hold circuit

There are used to sample a signal at a precise time and hold the value constant during the conversion process

(2) Analogue multiplexers

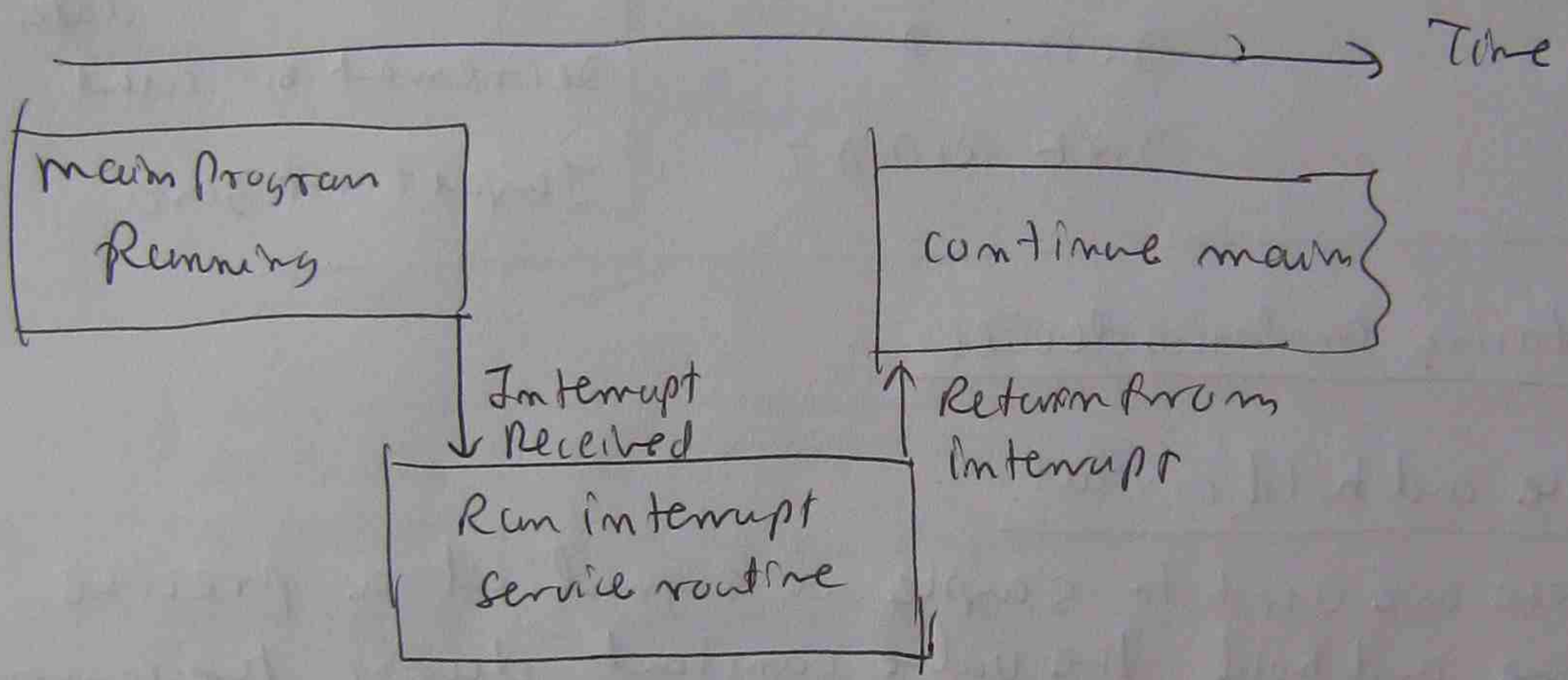
These devices permit one analogue signal out of several to be selected by logical control signals.

(3) Real time clock

Signal sampling and construction is often performed in conjunction with an interrupt driven real time clock.

Interrupts

To enable a peripheral device to inform the microprocessor when it wishes to transfer data. On receipt of the interrupt, the microprocessor temporily suspends its current activity, performs the required input or output operation, and then returns to its previous task.



Interrupt

(1) Save the contents of the Program counter on the System stack

(2) Load the program counter with the start address of the interrupt service routine

(3) Run the Interrupt Service routine

(4) Finally, return control to the interrupted Program by restoring the saved contents of the Program counter from the system stack.

circular buffer

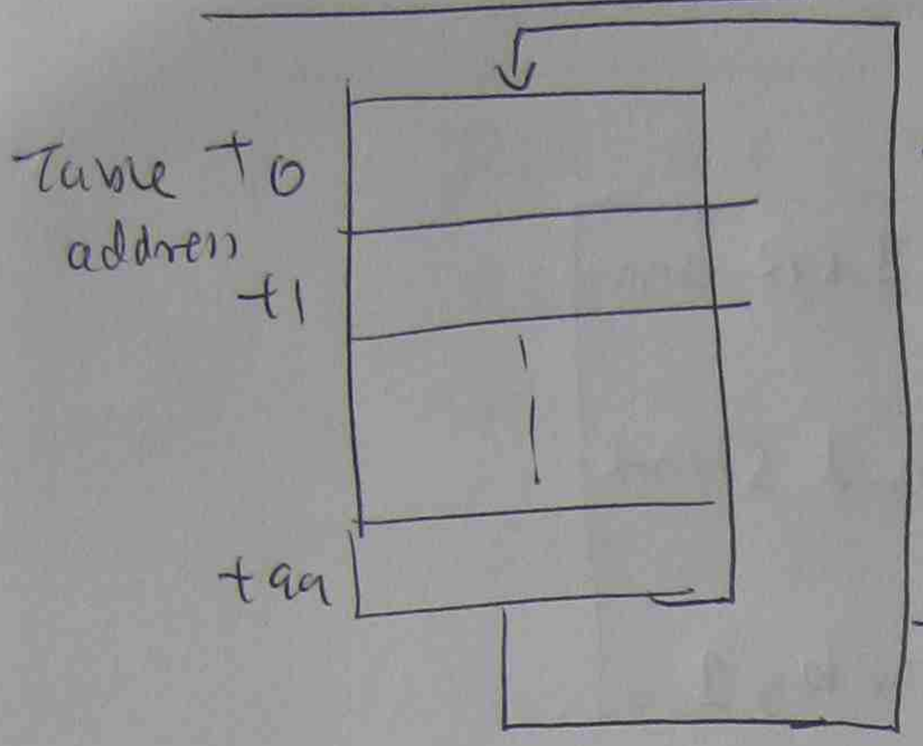
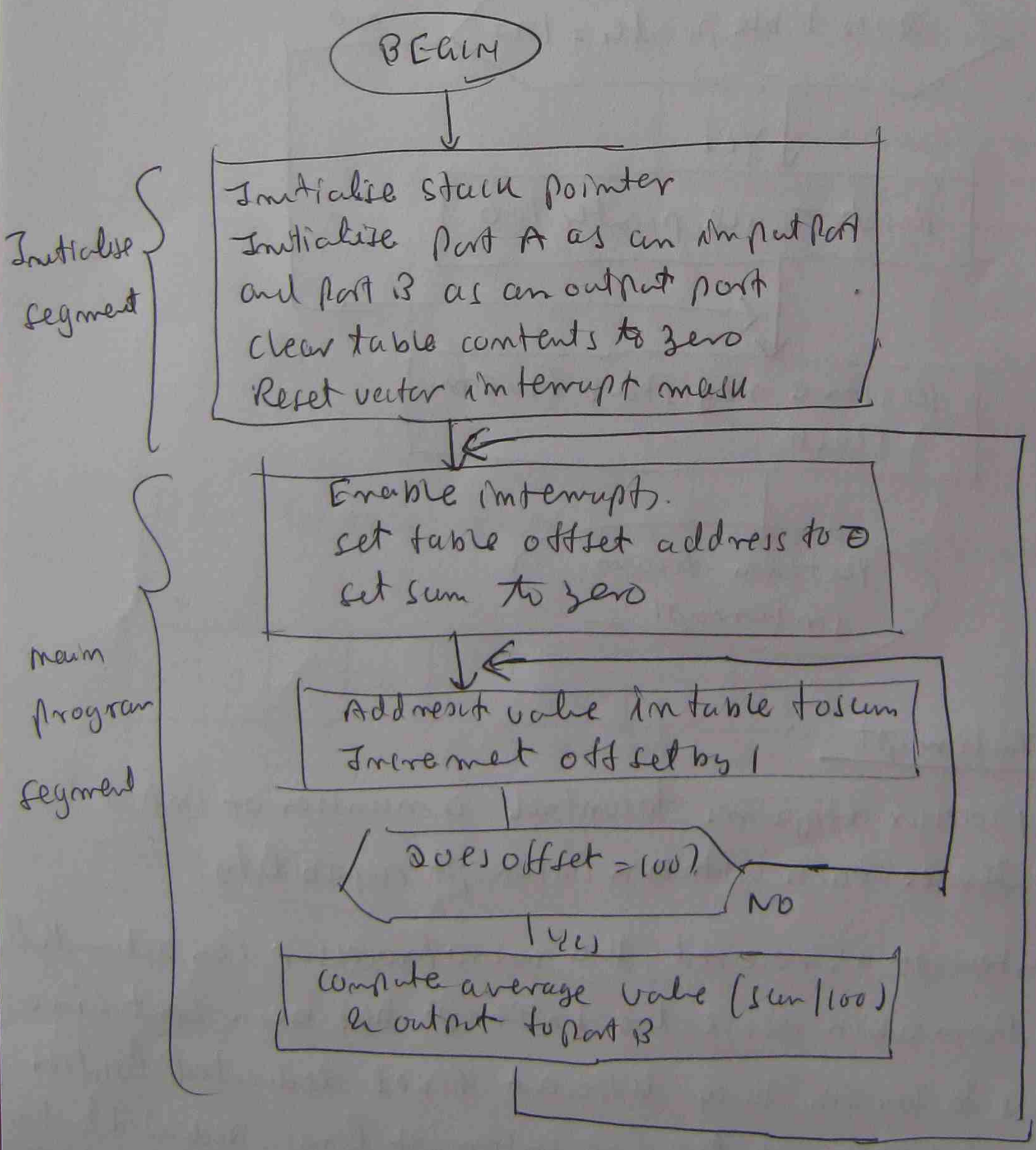


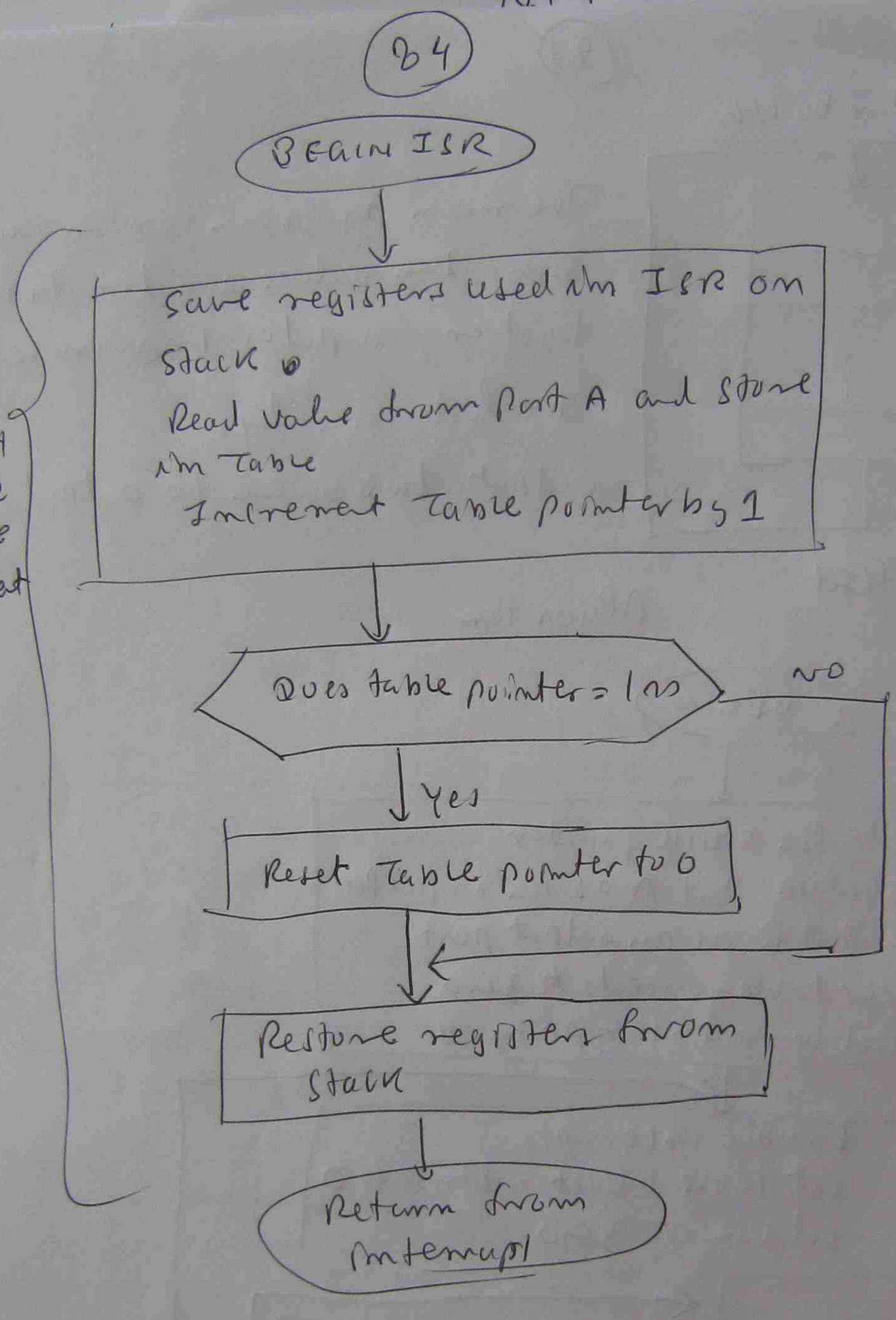
Table offset

- The main program continuously sums the values in the table together, computes the average of the last 100 values.
- output this value to output port.

Algorithm.



(b) 4



Multiple Interrupt

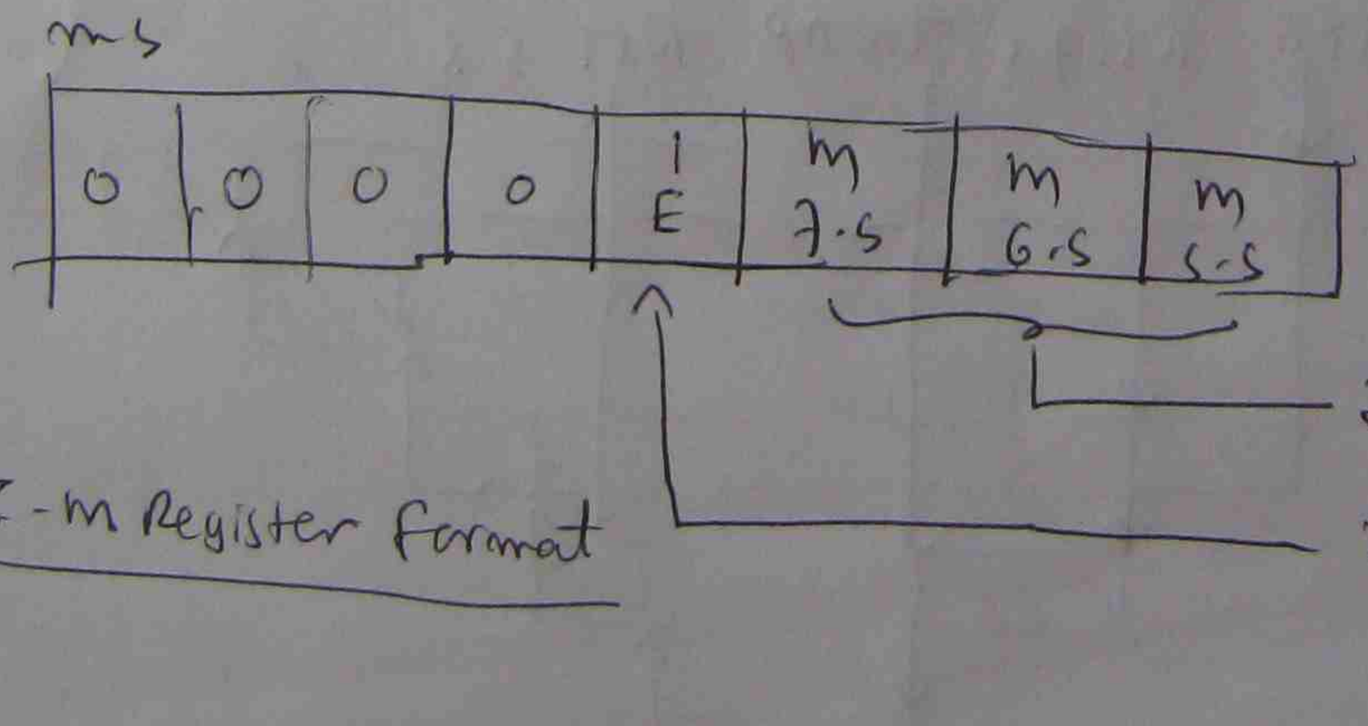
The microprocessor requires to control a number of input (or) output devices each with own interrupt capability.

When an interrupt is received, the microprocessor can automatically determine from which device the interrupt has been sent since it is caused to branch to a different fixed dedicated locations in memory for each of the five interrupt lines. Dedicated locations — vector address

Interrupt Input	Vector address
RST 4.5 (TRAP)	0024 (hex)
RST 5.5	002C
RST 6.5	0034
RST 7.5	003C
INTR	The vector address for this input is not fixed and is part of the instruction placed on the data bus by the interrupting device when the interrupt request is acknowledged

RST 4.5 (TRAP) Highest Priority
RST 7.5
RST 6.5
RST 5.5
↓
Lowest Priority

EI - Enable Interrupt, DI - Disable Interrupt



I-m Register Format

SIm - set interrupt mask

The A register is first loaded with the required mask bits in the three least significant bit positions together with a logical 1 in the 4th bit and the SIm instruction is then given

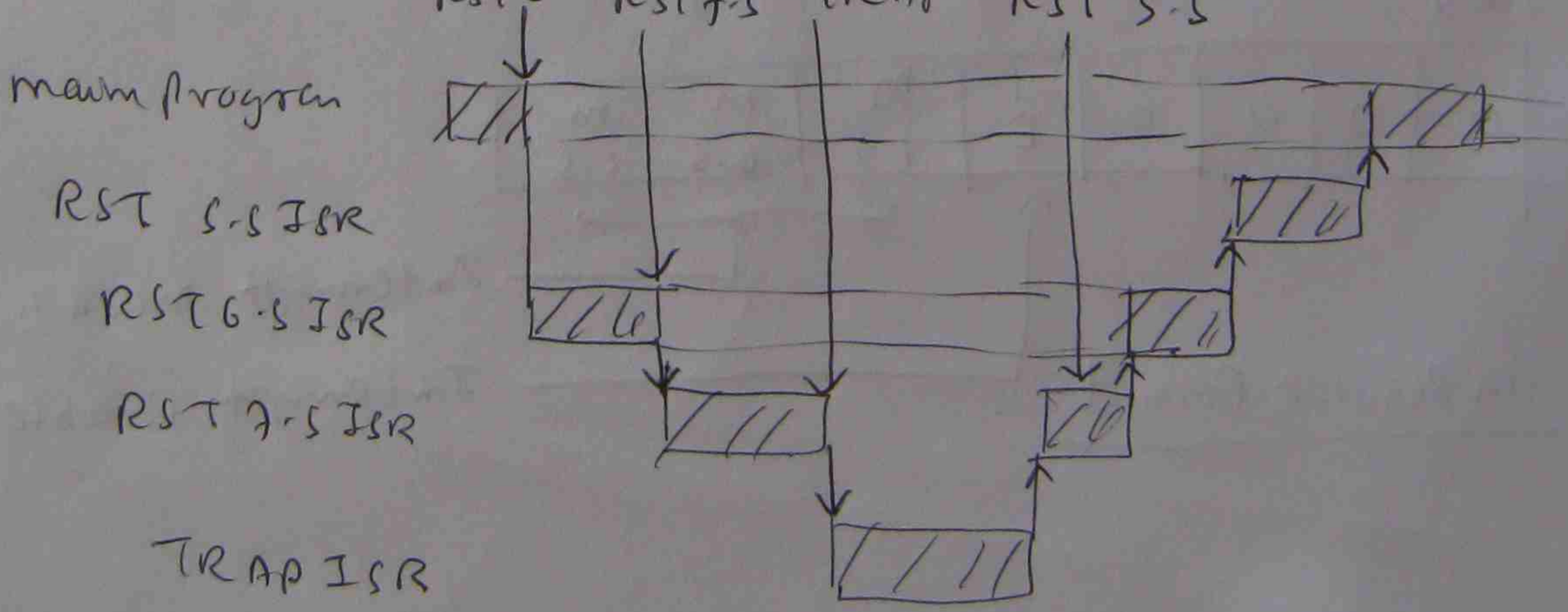
```
MVI A,08
SIM
H
```

RIm - Read Interrupt mask } After a RIm instruction the least significant 3 bits of A register indicate the state of the corresponding mask bit

Interrupt Priority Levels

Priority assignment of different interrupt inputs.

		Priority
TRAP	Power failure	HI
RST 7.5	Over temperature alarm	
RST 6.5	Real time clock	
RST 5.5	Read rear temperature set value	LO



Running

Suspended

	Assembly Instruction		comments
main program	BEGIN	MVI A, 02 SIM EI	Reset all mask bits and enable Interrupts
S.S Interrupt Service Routine	ISR S.S	PUSH PSW MVI A, 00 SIM EI RET	Disable S.S Interrupt & enable 6.S & 7.S Interrupts.
6.S Interrupt Service Routine	ISR 6.S	PUSH PSW MVI A, 0B SIM EI RET	Disable S.S and 6.S Interrupt and enable 7.S Interrupts.
7.S Interrupt Service Routine	ISR 7.S	PUSH PSW MVI A, 0f SIM EI RET	Disable S.S, 6.S and 7.S Interrupts.

Use of micro processor with interrupt

Ex 7.3 1 The program first initialize an area of memory as a stack

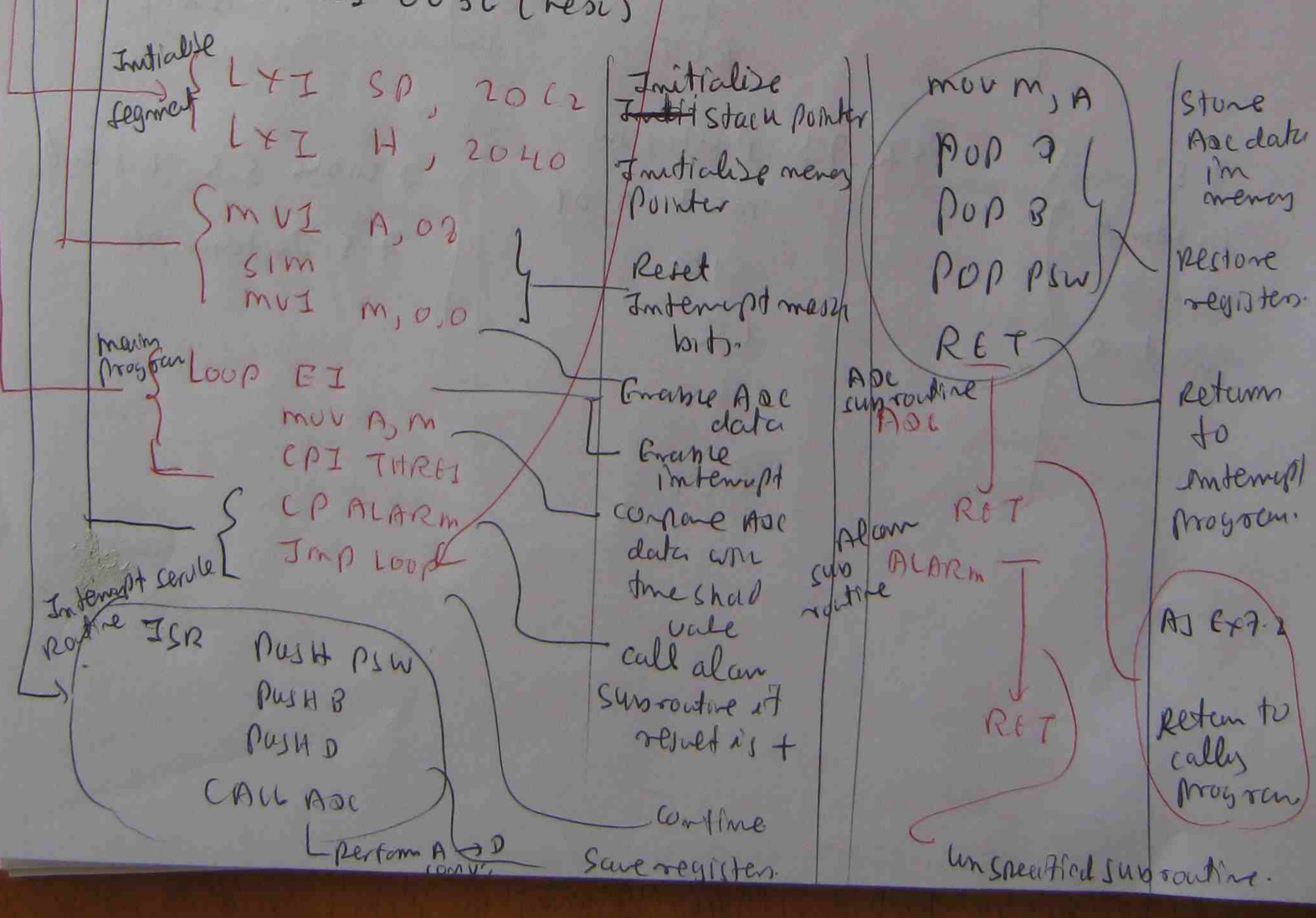
2 Reset the interrupt mask bits.

3 Then continuously compare the data returned by subroutine ADC. with a preset threshold value. (THRES)

4 If this value is equalled or exceeded, subroutine alarm is called.

5 Analogue to digital conversion is performed on receipt of interrupt on RST.

6 The instruction JMP ISR must therefore be stored in memory starting at the interrupt vector address 003C (hex)



Application Examples

Micro processer applications →
- sequence controller
- digital clock, timing sequence
- digital to analogue converter
- Analogue to digital converter

Basic Sequencing

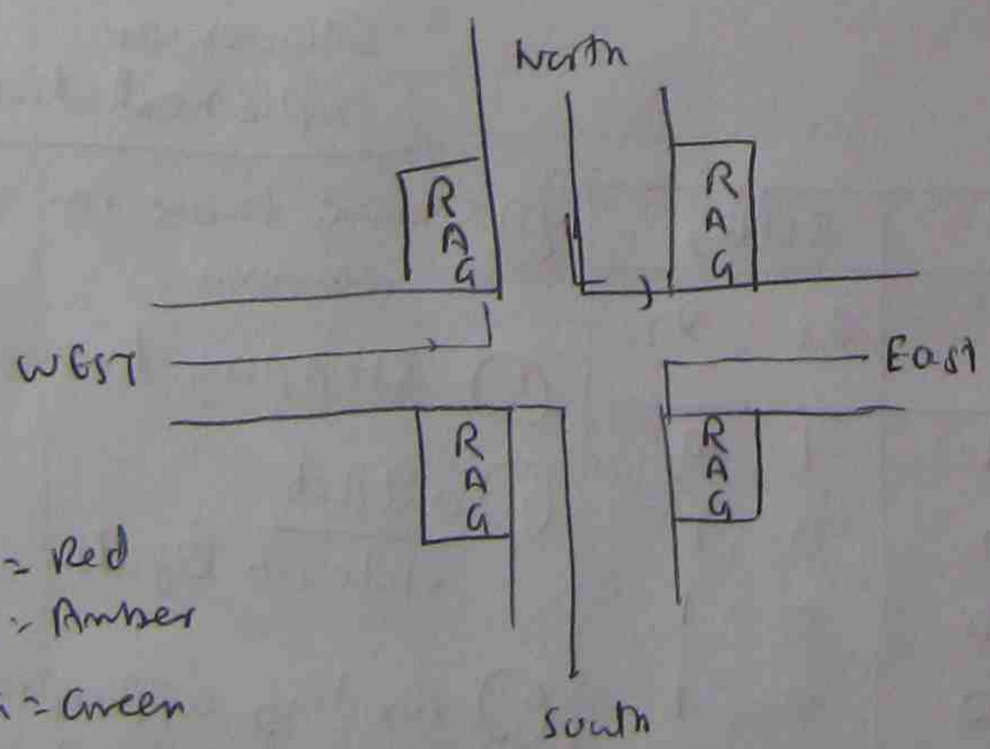
- The instructions provided by a microprocessor to input & output logical data to and from an external device.
- Programmable input/output (PIO) port may be used to provide the necessary latching (&) isolation functions.

Thermostat → Logical Input

Logical output → Relay (ON/OFF control)

A sequencer activates a number of devices in a preset sequence with a preset time delay between each new device state.

Ex(1) Traffic light sequencer



I SEQUENCING

- Sequence

Red, Red + Amber, Green, Amber

Red again

North & South - change in the same sequence

East & West - change in the same sequence.

Traffic light sequence table

	North / South			East / west		
	Red	Amber	Green	Red	Amber	Green
7 state 0 delay 1	ON	OFF	OFF	OFF	OFF	ON
1 delay 2	ON	OFF	OFF	OFF	ON	OFF
2 delay 2	ON	OFF	OFF	ON	OFF	OFF
3 delay 2	ON	ON	OFF	ON	OFF	OFF
4 delay 1	OFF	OFF	ON	ON	OFF	OFF
5 delay 2	OFF	ON	OFF	ON	OFF	OFF
6 delay 2	ON	OFF	OFF	ON	OFF	OFF
	ON	OFF	OFF	ON	ON	OFF

Light ON = 1 Light OFF = 0

delay 1 presents → $P_1 = 1$ delay 1 is not present $P_1 = 0$

delay 2 presents → $P_2 = 1$ delay 2 is not present $P_2 = 0$

delay 1 = Long delay between overall direction changes (~2 mins)

delay 2 = Short delay between transitional light settings (~3 sec)

Traffic light state table

State	N/S	E/W			delay	
	R	A	G	R'	A'	G'	D_1	D_2
0	1	0	0	0	0	1	1	0
1	1	0	0	0	1	0	0	1
2	1	0	0	1	0	0	0	1
3	1	1	0	1	0	0	0	1
4	0	0	1	1	0	0	1	0
5	0	1	0	1	0	0	0	1
6	1	0	0	1	0	0	0	1
7	1	0	0	1	1	0	0	1

Sequencer Implementation

① Store table in memory

② stepping through it

③ output / state of lights.

④ writing appropriate delay time between steps

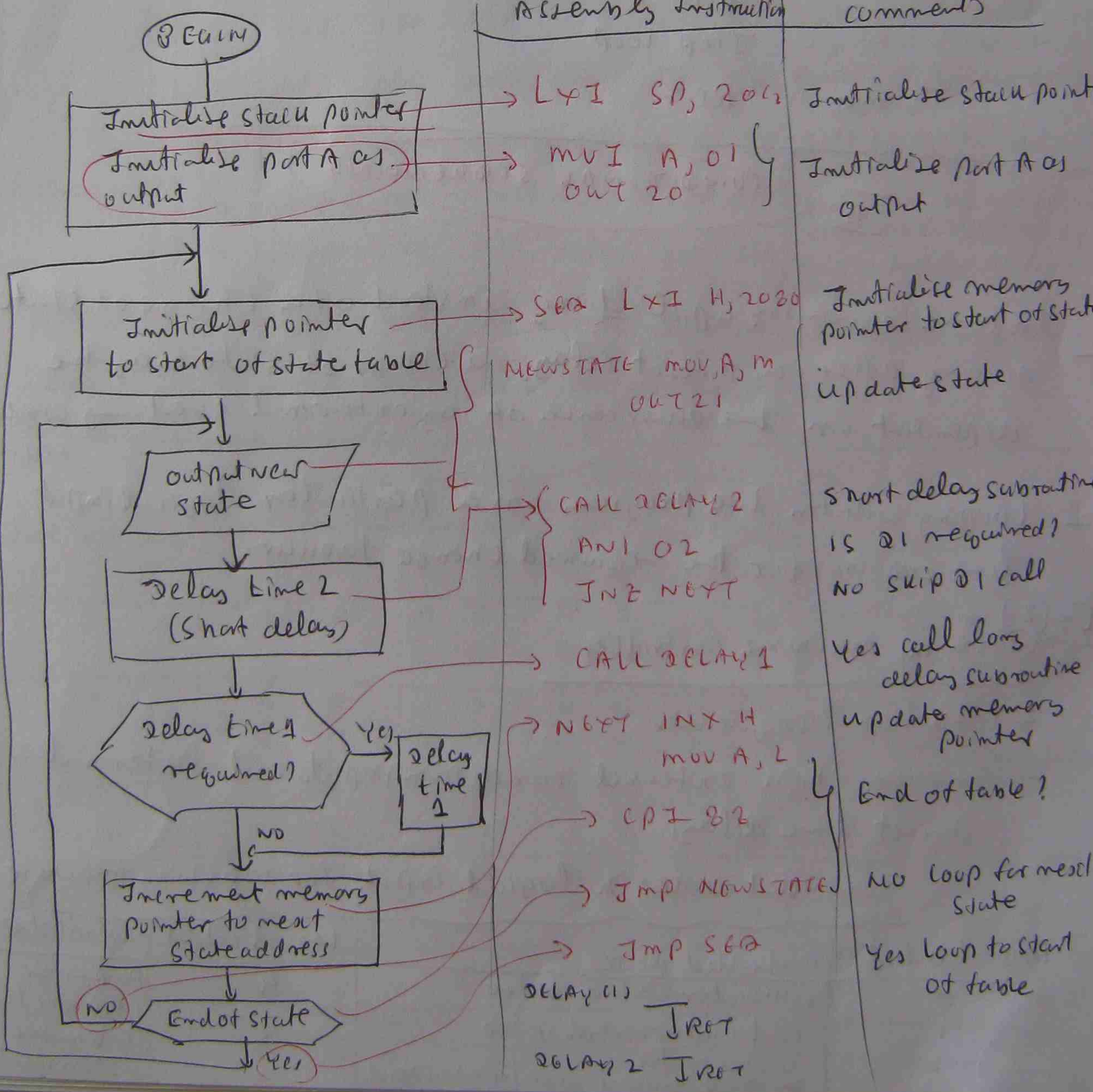
8155 PIO

- most significant 3 bits of port A device arrive North/south lights (RAG)
- The next most significant 3 bits arrive the East / west lights (R', A', G')

Relay time;

Stage 1 – New state, a delay of D_2 is executed

Stage 2 – if required a further delay of $D_1 - D_2$ is executed.



DELAY (1)/(2)

```
TIM DLY     MVI A,00
LOOP        CMP C
            JZ TIME
            NOP
            NOP
            NOP
            NOP
            NOP
            DCR C
            JMP LOOP
TIME        RET
```

II CONDITIONAL SEQUENCING

In some sequencing applications, instead of a change of state occuring after a preset delay, a change of state may be dependent on the occurance of an external event. → conditional sequencing

- Looping within the program on a particular logical input line waiting for the required change to occur.

EX(2)
washing machine controller

- conditional sequencer
- Involves both external condition inputs and internal Preset time delays.

controlled devices & logical inputs for washing machine

Logical output	controlled device	Logical output	controlled device
0	Hot water control valve	4	Pump motor (emptying tub)
1	cold water control valve	5	Tub motor spin speed
2	water heater		
3	Tub motor/ wash/rinse speed		

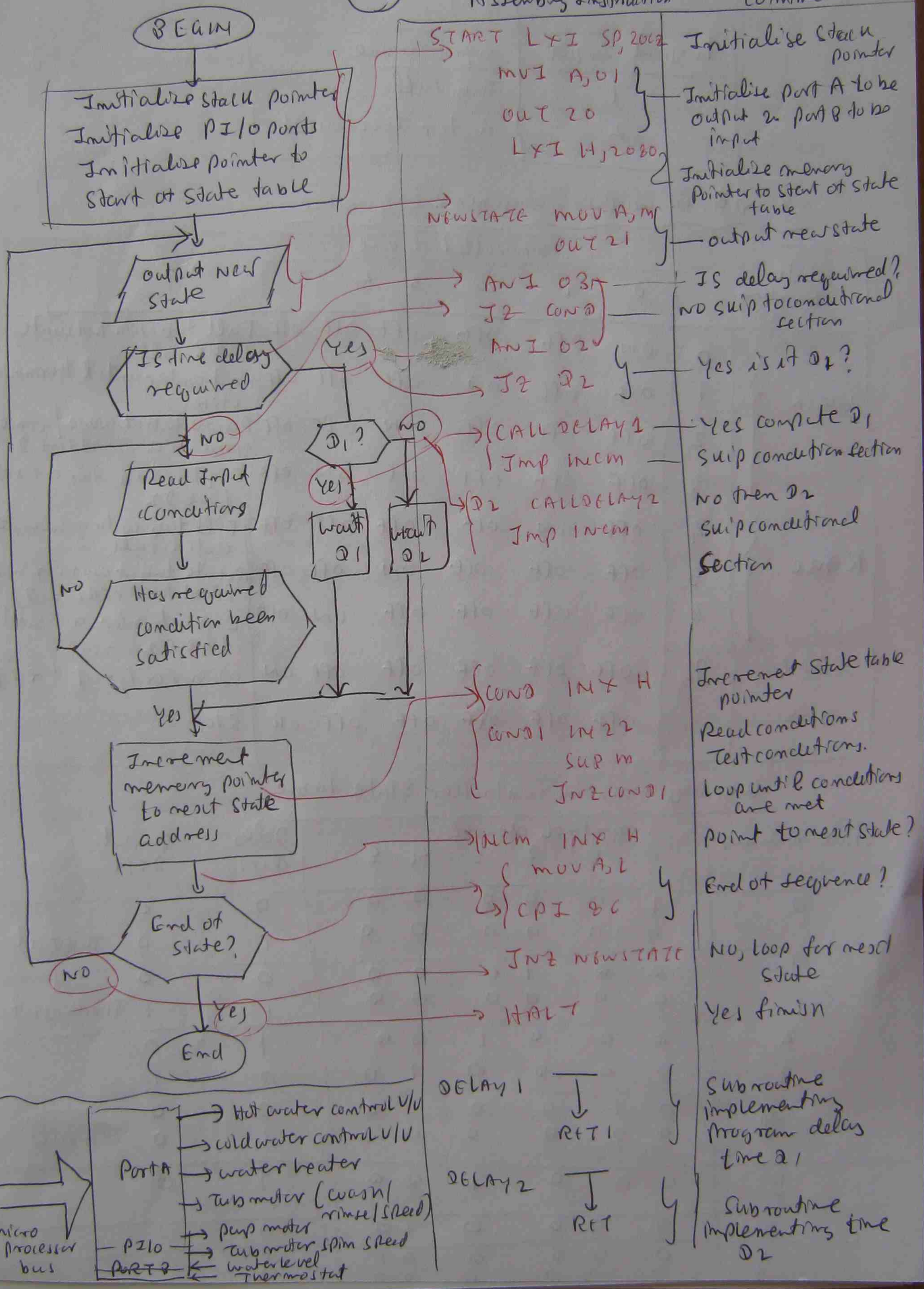
Logical Input	significance
1	Tub full
2	water thermostat

simplified washing machine sequence

State Number		controlled device 0	1	2	3	4	5	Action
Wash	0	ON	Off	Off	Off	Off	Off	Fill tub with hot water until full
	1	Off	Off	ON	Off	Off	Off	Heat water until thermostat closes.
	2	Off	Off	Off	ON	Off	Off	Rotate tub at wash/rinse speed for a fixed time D1
	3	Off	Off	Off	Off	ON	Off	Empty tub for a fixed time D2
Rinse	4	Off	ON	Off	Off	Off	Off	Fill tub with cold water until full
	5	Off	Off	Off	ON	Off	Off	Rotate tub at wash/rinse speed for fixed time D1
	6	Off	Off	Off	Off	ON	Off	Empty tub for a fixed time D2
SPIN	7	Off	Off	Off	Off	Off	ON	spin for fixed time D3
OFF		Off	Off	Off	Off	Off	Off	Stop

Washing machine controller state table

State number	control device 0	1	2	3	4	5	Delay D1/1	Input D2/2	
0	1	0	0	0	0	0	0	0	
	0	0	0	0	0	0	1	0	Tub full
1	0	0	1	0	0	0	0	0	
	0	0	0	0	0	0	1	1	Thermostat
2	0	0	0	1	0	0	1	0	
3	0	0	0	0	1	0	0	1	
4	0	1	0	0	0	0	0	0	
	0	0	0	0	0	0	1	0	Tub full
5	0	0	0	1	0	0	1	0	
6	0	0	0	0	1	0	0	1	
7	0	0	0	0	0	1	1	0	
8	0	0	0	0	0	0	0	1	


BEGIN
Initialize stack pointer
Initialize PI/O ports
Initialize pointer to start of state table
output NEW state
Is time delay required
Yes
NO
Read Input conditions
NO
Has required condition been satisfied
Yes
Increment memory pointer to next state address
End of state?
NO
Yes
End
D1?
Yes
NO
wait D1
wait D2
Hot water control V/V
cold water control V/V
water heater
Tub motor (wash/ rinse/ SPeed)
pump motor
Tub motor Spin SPeed
water level
Thermostat
Port A
PZIO
PORT B
micro Processor bus
START LXI SP, 2002
MVI A, 01
OUT 20
LXI H, 2030
NEWSTATE MOV A, M
OUT 21
ANI 03H
JZ COND
ANI 02
JZ D2
CALL DELAY 1
JMP INCM
D2 CALLDELAY2
JMP INCM
COND INX H
COND1 INX H
SUB M
JNZ COND1
INCM INX H
MOV A, L
CPI 8C
JNZ NEWSTATE
HALT
DELAY 1
RET 1
DELAY 2
RET
Assembly Instruction
comment
Initialise stack pointer
Initialise Port A to be output & Port B to be input
Initialize memory Pointer to start of state table
output new state
IS delay required? NO skip to conditional section
Yes is it D1?
Yes compute D1 skip conditional section
NO then D2 skip conditional section
Increment State table pointer
Read conditions
Test conditions.
Loop until conditions are met
Point to next state?
End of sequence?
NO, loop for next state
Yes finish
Subroutine implementing Program delay time D1
Subroutine implementing time D2

11) Programmable Timer

Intell 8155 - Single integrated circuit which incorperates a programmable timer with a 256 x 8 bit state RAM and 3 programmable input/output ports

- 14 bit counter — Programmed to generate either a square wave of a selectable period (or) terminal count pulse.

Tmax ≤ 8.1915 ms

Timer may be programmed either to stop after the terminal count pulse is generated (OR) to continue counting and hence generate a new count pulse every, say, 8.1915 ms.

Digital clock Ex(3)

use the programmable timer to continuously generate a count pulse every 5 ms and to connect the timer output to one of the interrupt lines (RST 7.5) of the microprocessor.

COUNT - contains the current number of interrupts since the last change in SECS

SECS - contains the two BCD digits corresponding to seconds. The contents are incremented by 1 each time time COUNT reaches 200.

MINS - contains the two BCD digits corresponding to minutes. The contents are incremented by 1 each time SECS reaches 60.

HOURS - contains the two BCD digits corresponding to hours. The contents are incremented by 1 each time MINS reaches 60.

LXI SP, 20C2 ← Initialise stack pointer
LXI H, 2030
MVI M, 00 } clear count
INX H
MVI M, 00 } Sec min to 0
INX H
MVI M, 00
INX H
MVI M, 12 } Preset Hr=12

MVI A, 10 } Initialise count for 5ms
OUT 24
MVI A, 27
OUT 25

MVI A, C3 } Initialise Port A, B as o/p & timer
OUT 20
MVI A, 0B } Reset interrupt mask unit
SIM

Loop EI
LXI H, 2032 } o/p to Port B
OUT 22
INX H
MOV A, M } Load Hr digit to A o/p to port A
OUT 21
JMP LOOP

ISR

Initialise stack pointer

Add 1 to count

IS COUNT 200?

NO

Yes

Reset count to 0
Add 1 to SECS & adjust for decimal

Is SECS = 60?

NO

Yes

Load min: digit to A

Reset SECS to 0
Add 1 to MINS and adjust for decimal

IS MINS = 60?

NO

Yes

Reset MINS to 0
Add 1 to HOURS & adjust for decimal

IS HOURS = 24

NO

Yes

MVI M, 00
Yes reset Hr to 0

MVI A, 24
CMP M
JNZ END
IS HR = 24 NO end

END: POP H
POP PSW
RET
Restore register content

RET
Return from interrupt

ISR PUSH PSW } save regist:
PUSH H
LXI H, 2030 } Add 1 to count
INR M
MVI A, C8 } IS count = 200?
CMP M
JNZ End

BCD digit from BISS → BCD to 7 segment decoder driver → 7 seg LED Display

MVI M, 00
Yes reset count to 0

INX H
MOV A, M
INR A
DAA
MOV M, A
Add 1 to SECS for decimal

MVI A, 60 } IS SECS = 60?
CMP M
JNZ END no end

MVI M, 00

INX H
MOV A, M
INR A
DAA
MOV M, A
Add 1 to MIN & adjust for decimal

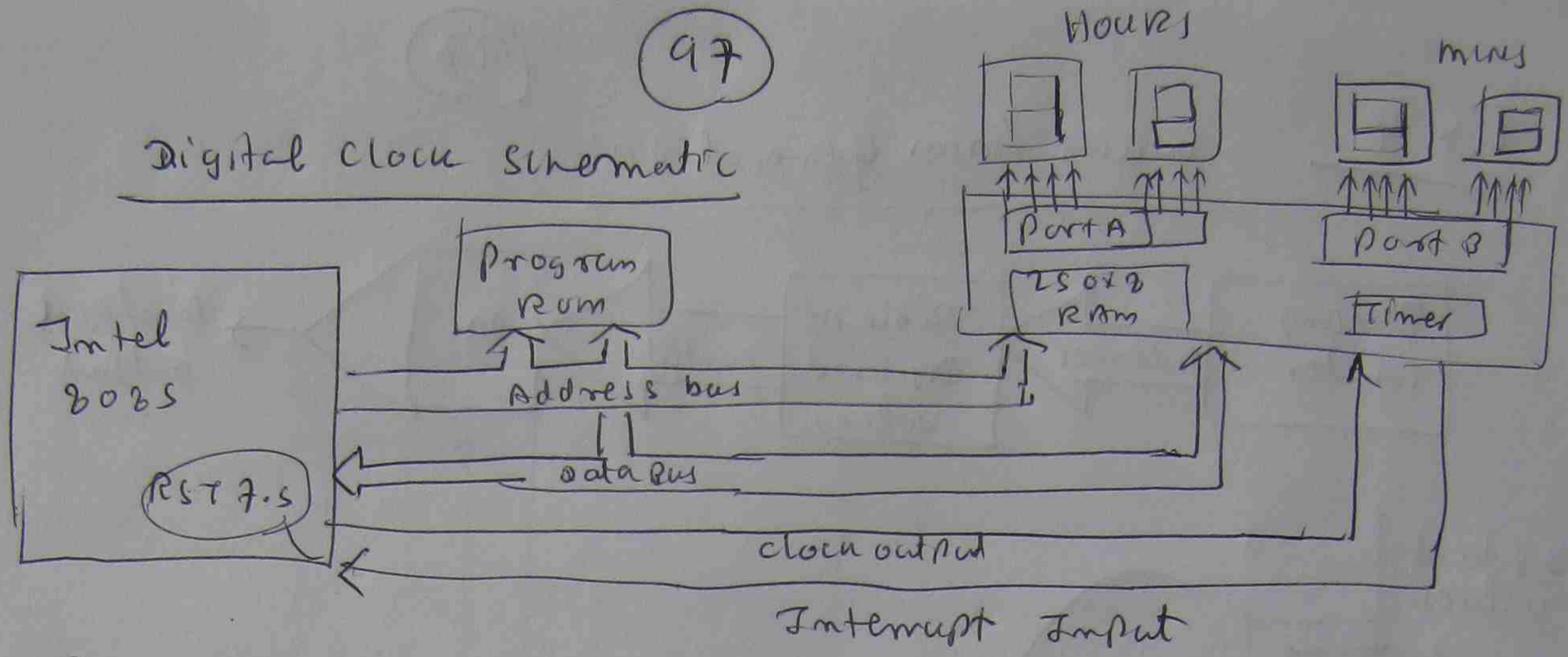
MVI A, 60
CMP M
JNZ END no end

Interrupt Service routine

IS min = 60?
no end

MVI M, 00
INX H
MOV A, M
INR A
DAA
MOV M, A
Add 1 to HOURS & adjust for decimal

Digital clock schematic



RAm addresses

2000 - Stack pointer
2080 - count
2081 - secs
2082 mins
2083 Hours

8155 addresses

command Register 20 (hex)
Port A data 21
Port B data 22
low order byte of count length 24
High order byte of count length 25

IV WAVE FORM GENERATION

Direct wave form

- Use of DAC to convert the digital output from a microprocessor into an analogue form
- use of look up table.
- Micro processor was used as a counter
- Its contents were incremented by unity with in a program loop
- The contents were output to DAC after each count increment.

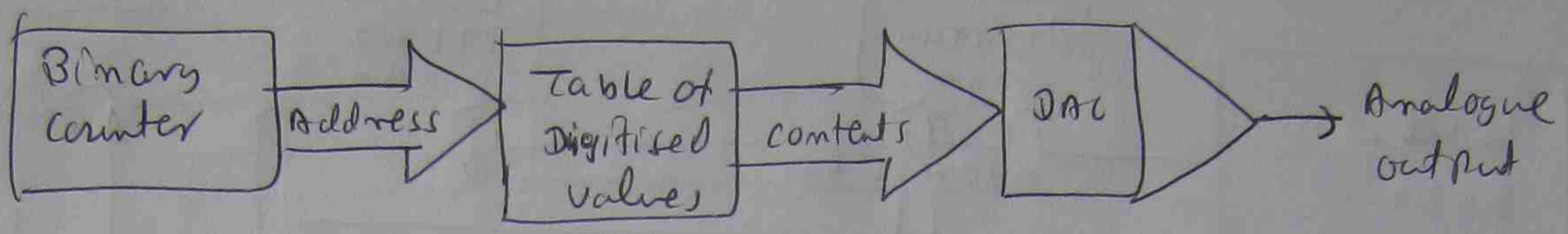
Alternative wave form

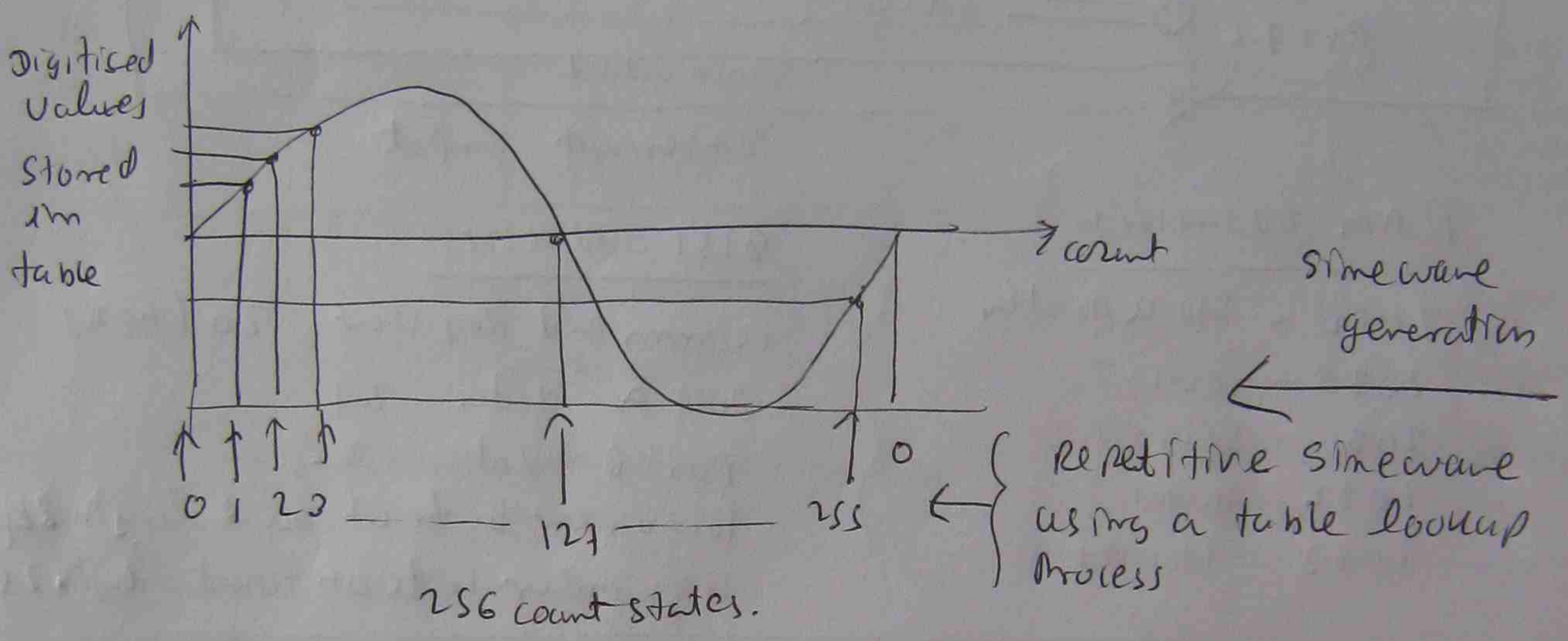
- The contents of the counter not to drive the DAC directly
- To provide addresses to sucessive memory locations, the contents of which store the required digital value which is to be output to the DAC.

Table — The contents of the block of sucessive memory locations
Table look up process.

DAC - 8 bits, 256 count states.

Ex 4 — Wave form generation

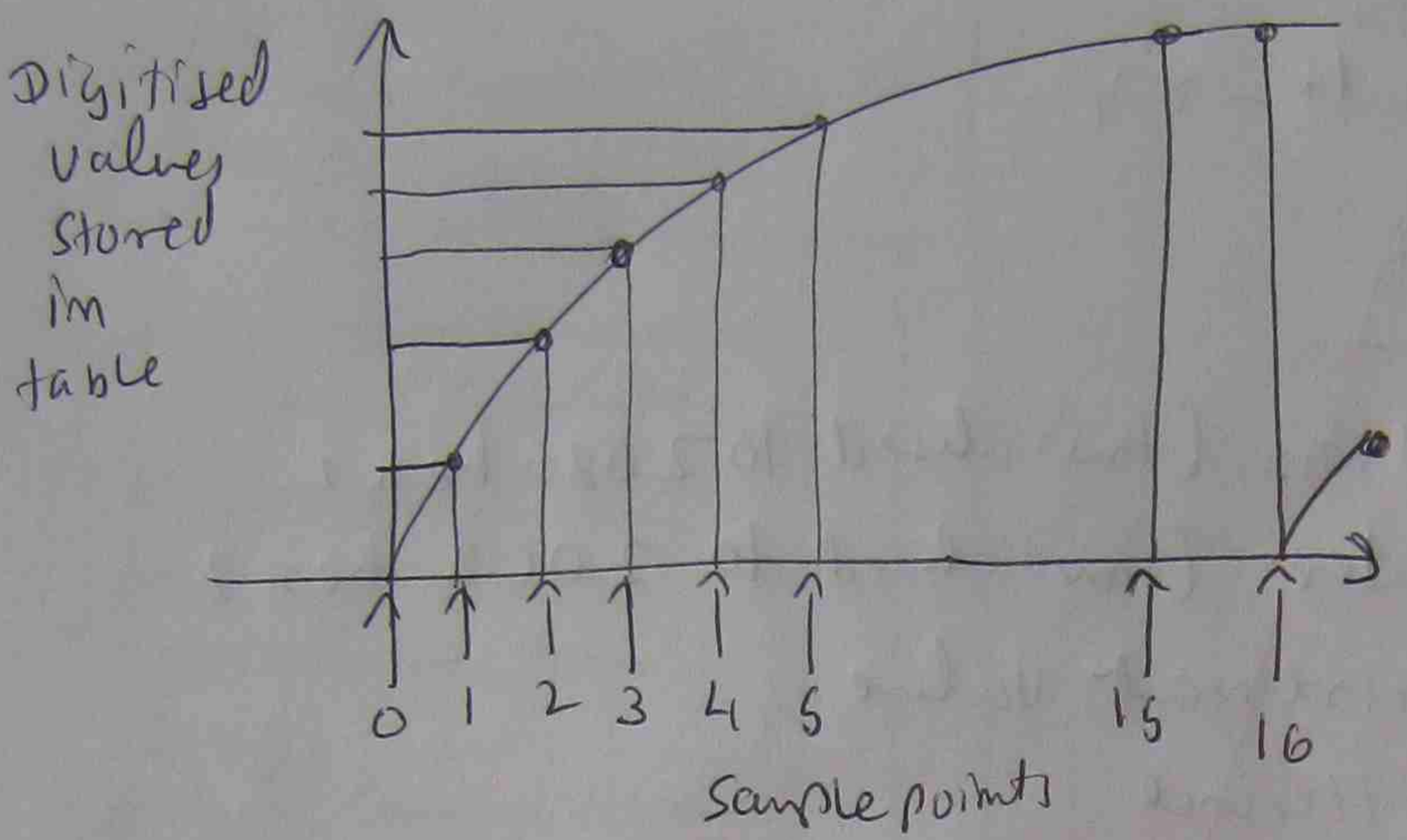


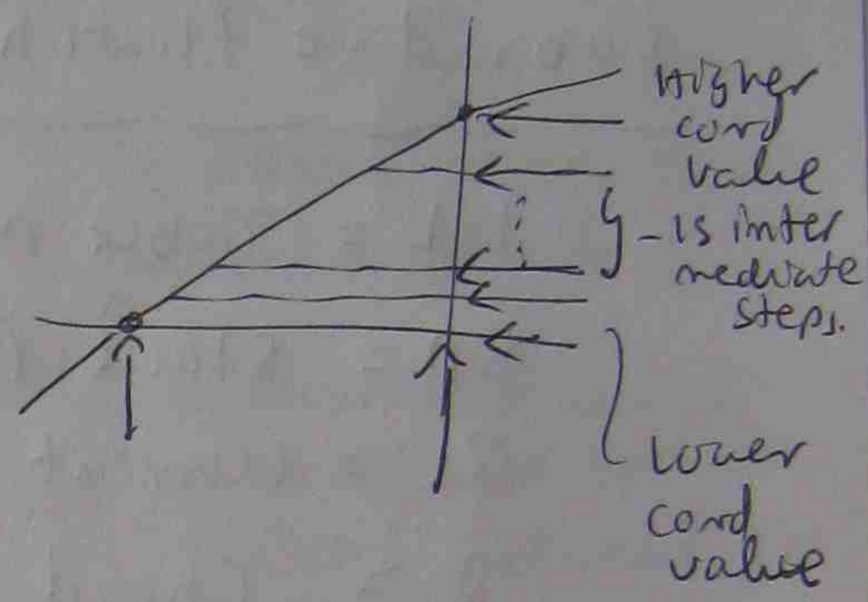


- The table is stored in memory starting at address 2000 and
  - Register pair HL is initialized to this value.
  - Register L is 8 bit counter & the combined contents of register pair HL.
  - Provide complete 16 bit memory address automatically.
- On reaching FF, the contents of register L will return to 00
- the combined contents of HL return to 2000 & the process will repeat.

Assembly Instructions		Comments
	MVI A, 01	Initialize port A as an output port
	OUT 20	
	LXI H, 2000	Initialize register pair HL to point to start of table
LOOP	MOV A, M	Lookup value from table
	OUT 21	output value to DAC
	INR L	Increment table offset
	JMP LOOP	

V Piece wise Linear interpolation





↑ Piecewise linear interpolation ↑

↑ Linear interpolation process ↑

17 stored values → Chords

15 values in between → Steps
adjacent chords

Intermediate values between adjacent chord values are obtained by first evaluating the chord difference.

Ex 5 The program generates an exponential wave form using a table look up process and piece wise linear interpolation.

- To find intermediate value, it maintains a current increment value (Chord x n) and simply adds the cord difference to this following each iteration.
- The current increment value is held as a 16 bit number in register pair BC.
- The combined contents are divided by 16 to form each intermediate value.
- (Shifting the current increment value four places to the right. The process is performed in separate subroutine)

Subroutine

IF (B) = 0x (hex) (ms 4 bits of B are always zero)

(C) = YZ (hex)

Then (A) ← result - XY

Subroutine flowchart

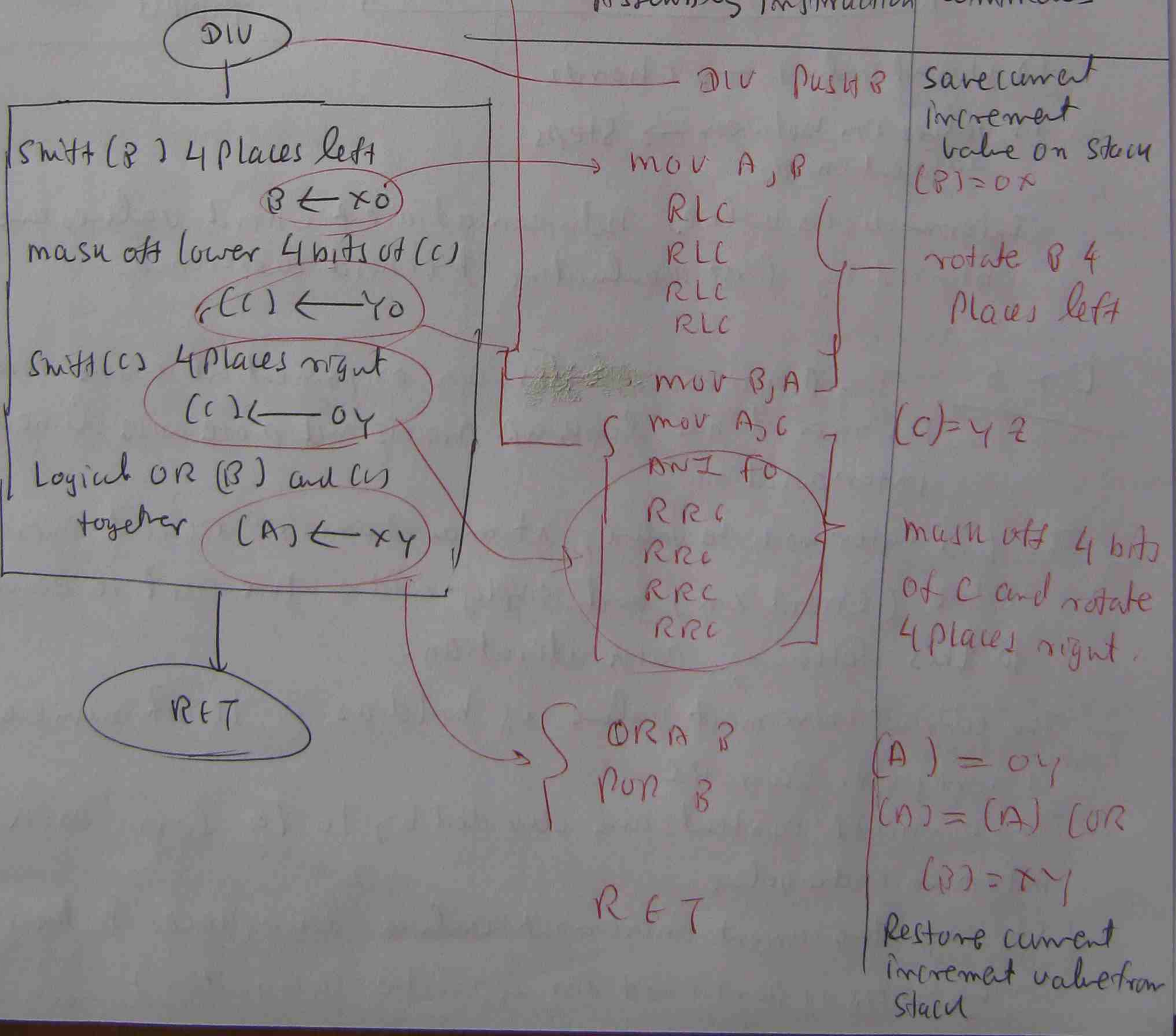
HL = Table Address (initialised to 2080 hex)

SP = Stack Pointer (initialised to 20C2 hex)

BC = Current Increment value

D = Chord Difference

E = Step count



101

BEGIN

Initialise Table Offset to zero

Set current Increment to zero

Read lower & higher Chord values from table

Compute Chord difference

Initialise stepcount to 15

Add Chord Difference To temporary Increment value

Divide result by 16 (subroutine DIV) and add to lower Chord value

Output result to DAC

Decrement stepcount

NO — Is stepcount = 0?

Yes

Increment Table Offset

Is offset = 16? — NO

Yes

```
LXI SP, 20C2
MVI A, 01
OUT 20
```

Initialise Stack Pointer

Initialise Port A as an output Port

```
START LXI H, 2020
      LXI B, 0000
```

Initialise HL to start of Table

clear BC

```
CONT  INR L
      MOV A, M
      DCR L
      SUB M
      MOV D, A
```

Obtain Higher Chord value from table

Substract lower Chord value to from Chord Differenc & save D

```
      MVI E, 0F
```

Set setupcount to 15

```
STEP  MOV A, C
      ADD D
      MOV C, A
      JNC NXT
      INR B
```

Add Chord Difference to 16 bit current Increment value

```
NXT   CALL DIV
      ADD M
```

Divide current Increment value by 16 & add to lower Chord value

```
      OUT 21
      DCR E
      JNZ STEP
      INR L
      MOV A, L
      CPI 10
      JNZ CONT
      JMP START
```

output value to DAC

Test if step count = 0

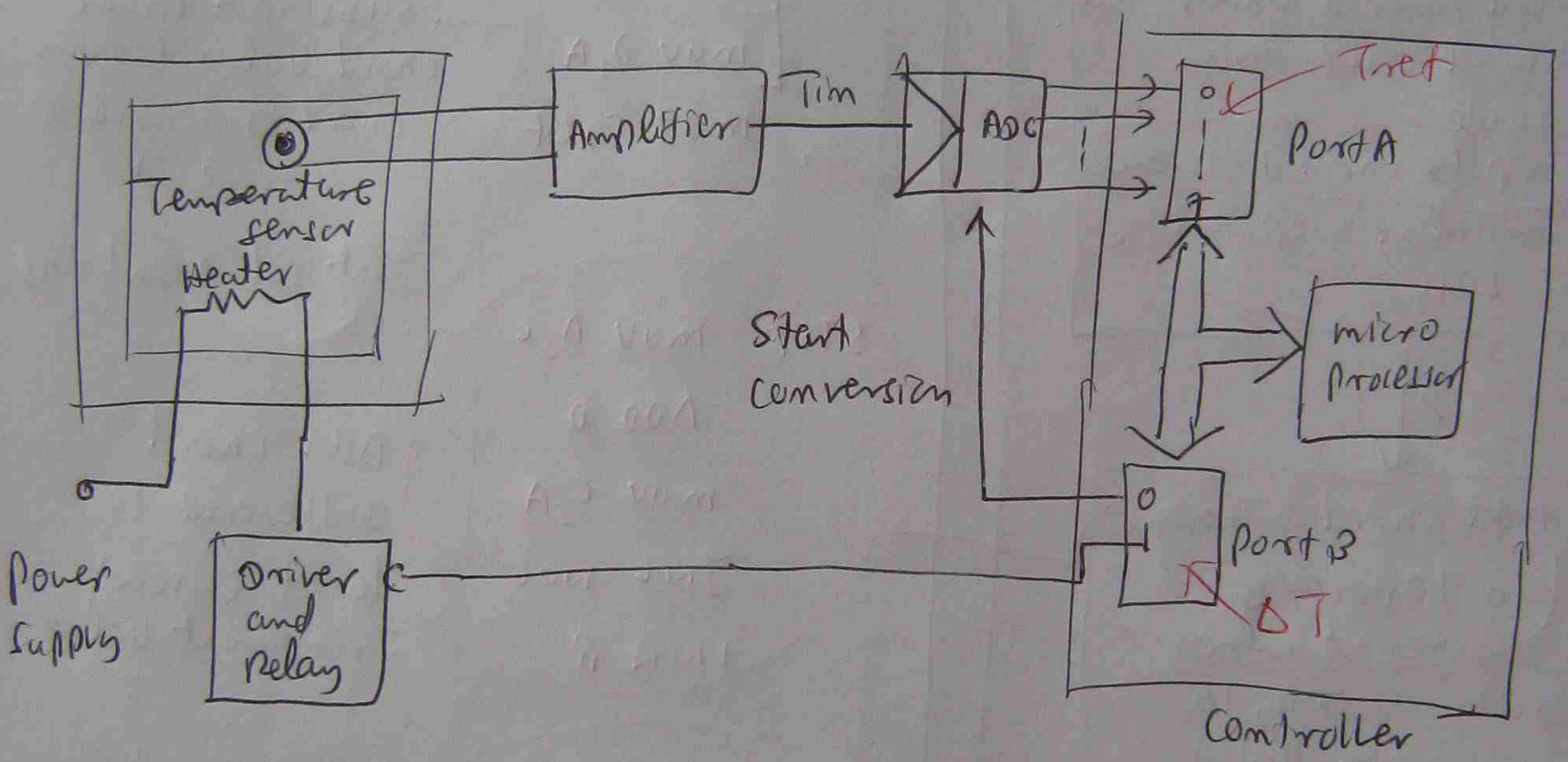
Increment Table Offset & test if = 16

NO continue

Yes - Reinitialise Table offset & repeat

Temperature controller

- The required temperature Tref is variable and is stored in a location in RAM.
- The tolerance limits on the required temperature $\pm \Delta T$ are also assumed variable in stored in a second RAM Location.



- The controlled temperature is controlled by first deriving an analogue voltage Proportional to the temperature - from a thermistor & associated amplifier.
- converting this voltage into an 8 bit digital value by means of an ADC
- supply to heater is controlled by a single digital output from the microprocessor via a suitable driver & relay

1 ⟶ heater on 0 ⟶ heater off

If Input $T_{im} > T_{ref} + \Delta T \Longrightarrow$ Heater is ~~off~~ turned off

If Input $T_{im} < T_{ref} - \Delta T \Rightarrow$ Heater is turned on

BEGIN

Start conversion and wait convert time

Read digitised ~~output~~ input temperature from ADC

Tim < (Tref + ΔT) ?

NO

Yes

Switch heater ON

T im > (Tref + ΔT)

NO

Yes

Switch heater Off

Temperature controller flow chart

Assembly Instruction	comments
LXI H, 2000	
MOV A, M	Read Tref from A2
MOV E, A	Save in E
INX H	
SUB M	Form (Tref − ΔT) and
MOV B, A	Store in B
MOV A, E	
ADD M	Form (Tref + ΔT)
MOV C, A	and Store in C
MVI A, 02	Initialise Port A
OUT 20	as input
MVI A, 00	Port B as output
OUT 22	Switch heater off
START MOV A, D	Recall state of Port B
ORI 01	Switch bit 0 on
OUT 22	
ANI 02	Switch bit 0 off
OUT 22	Start ADC
CALL DELAY	wait conversion time
IN 21	Read Tim from ADC
CMP B	
JC HTON	If Tim < (Tref − ΔT)
JZ START	Jump to switch on (HTON)
CMP C	
JC START	If Tim < (Tref + ΔT)
JZ START	Jump to start
MVI A, 00	Switch heater off
MOV D, A	
OUT 22	
JMP START	
HTON MVI A, 02	
MOV D, A	Switch heater on
OUT 22	
DELAY	
JMP START	
RET	

Development Aids

- Single board system - Intel 8085 microprocessor microprocessor, system clock source, Random access memory for holding a system monitor program, a number of input/output ports, key pad, associated numeric display, bus control logic. application
- Program stored in ROM
- Development phase for program stored in RAM.

monitor command

Digit displayed are in hexadecimal code.

xxxx. xx
address field — data for content field

* Reset → Enable the user to force the monitor to restart from the beginning of the program

Substitute memory - Allow the user to examine the contents of successive memory location
If required, to modify its contents

Run - Execute a program which is already stored in RAM.
Run key is first pressed, followed by 4 digit start address.

Examine Register allows the users to display and if required, modify the contents of each of the microprocessor registers.

Single step - Enable the user to examine the state of the complete system.

To overcome the errors, additional Software / hardware are provided.

Assembly Language → [Assembler] → machine code

High level language → [compiler] → machine code.

Assembler

```
LDA  : SECS    Increment contents of memory
INR    A       Location with symbolic name SECS
STA    SECS    by unity

SECS DS 1      Define one storage location for SECS

LXI H SECS     This load the absolute value
                 in to register pair HL

INR    M       contents of memory location
                 SECS (2800 (hex)) are incremented
                 by 1

SECS EQU 2800H Define SECS to be
                 absolute value 2800 (hex)
```

macro

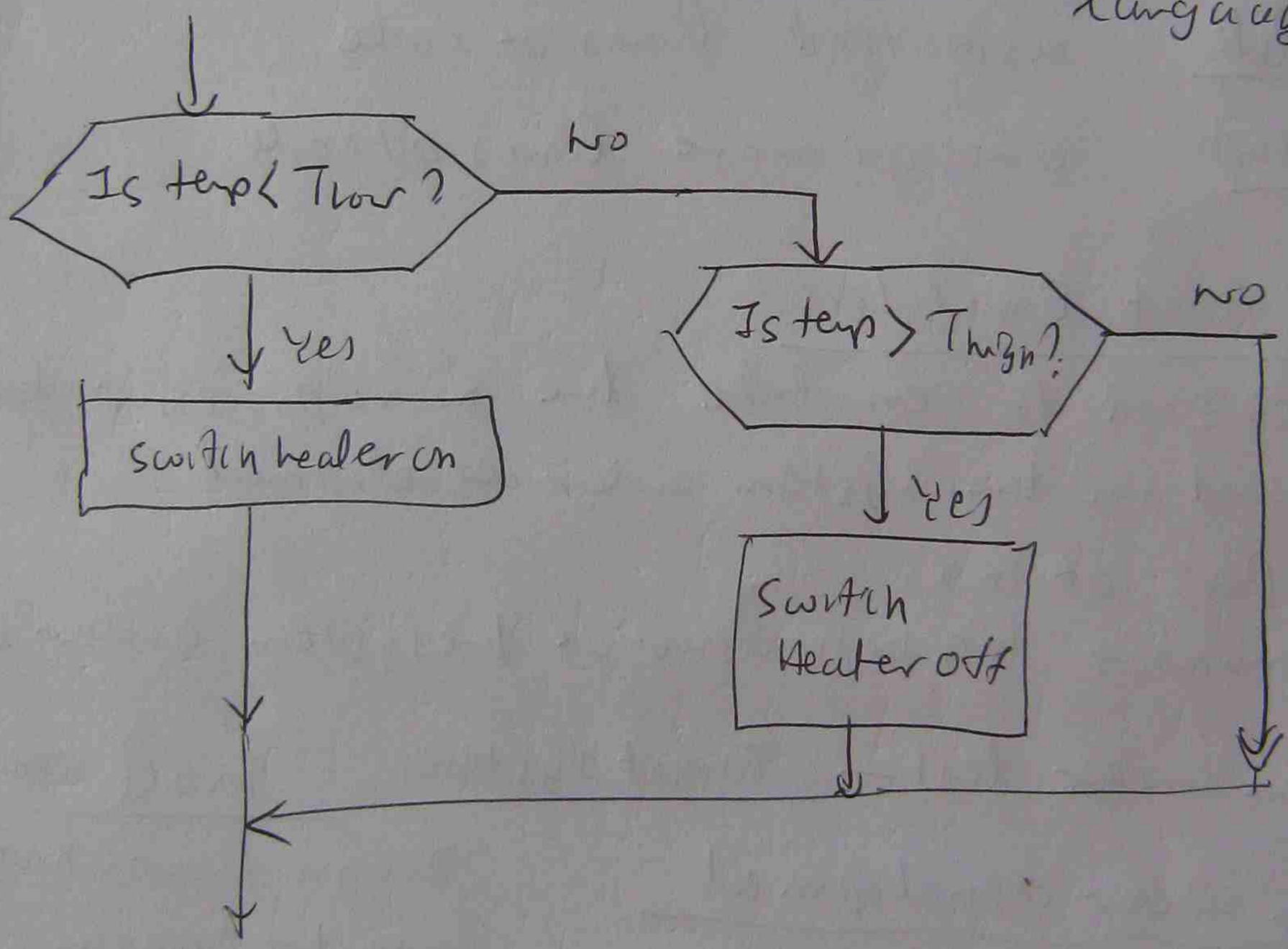
```
SHIFT : MACRO      start of macro SHIFT
        RRC
        RRC
        RRC
        RRC
        MEND       end of macro
```

To define the start address of compiler

```
ORG   2000H
List of Program
Instructions
  END
```

High level language → compiler → machine code

machine code → Interpreter → High level language.



IF TEMP < TLOW THEN HEATER : -1

ELSE IF TEMP > THIGH THEN HEATER :-0

Subroutine CALL DELAY (COUNT)

Editors

Enable Perser to readily modify Pe source Program code

Run interactively.

list a specified number of lines of source code on Pe visual display screen

move to a specified line in Pe source Program

delete a specified lines of code

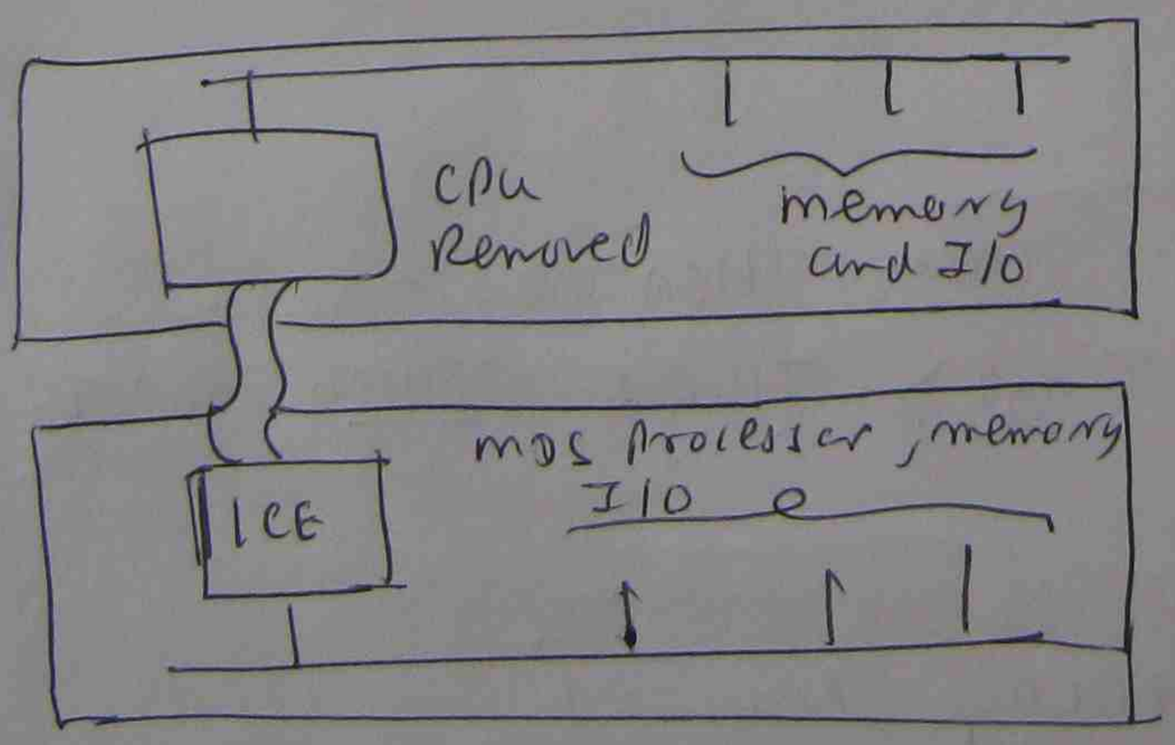
insert one or more lines of code.

In circuit Emulators

- designed to emulate Pe microprocessor being used in Pe system under development
- Part of mos.
- monitor Pe behaviour of Pe system under development

System under test — Target system

System under development



In circuit emulation

mos (Development System)

Design flowchart

code Pe program

Enter Pe program in to mos memory

Assemble / compile

Run Pe object program

Debug using ICE

Edit Pe source program

Proportional-Integral-Derivative Control

Dr M.J. Willis

Dept. of Chemical and Process Engineering
University of Newcastle

e-mail: mark.willis@ncl.ac.uk

Written: 17th November, 1998
Updated: 6th October, 1999

Aims and Objectives

The PID algorithm is the most popular feedback controller used within the process industries. It has been successfully used for over 50 years. It is a robust easily understood algorithm that can provide excellent control performance despite the varied dynamic characteristics of process plant. These lecture notes,

- introduce the Proportional- Integral- Derivative (PID) control algorithm.
- discuss the role of the three modes of the algorithm.
- highlight different algorithm structures.
- Discuss methods that have evolved over the last 50 years as aids in control loop tuning.

After completion of this section of the course a student should be capable of approaching a loop tuning problem in a competent and efficient manner and have sufficient knowledge to effectively tune a PID control algorithm.

The Proportional-Integral-Derivative (PID) algorithm

As the name suggests, the PID algorithm consists of three basic modes, the Proportional mode, the Integral and the Derivative modes. When utilising this algorithm it is necessary to decide which modes are to be used (P, I or D ?) and then specify the parameters (or settings) for each mode used. Generally, three basic algorithms are used P, PI or PID.

A Proportional algorithm

The mathematical representation is,

$\frac{mv(s)}{e(s)} = k_c$ (Laplace domain) or $mv(t) = mv_{ss} + k_c e(t)$ (time domain) (3)

The proportional mode adjusts the output signal in direct proportion to the controller input (which is the error signal, e). The adjustable parameter to be specified is the controller gain, k_c. *This is not to be confused with the process gain, k_p.* The larger k_c the more the controller output will change for a given error. For instance, with a gain of 1 an error of 10% of scale will change the controller output by 10% of scale. Many instrument manufacturers use Proportional Band (PB) instead of k_c.[1]

The time domain expression also indicates that the controller requires calibration around the steady-state operating point. This is indicated by the constant term mv_{ss}. This represents the 'steady-state' signal for the mv and is used to ensure that at zero error the cv is at setpoint. In the Laplace domain this term disappears, because of the ‘deviation variable’ representation.

A proportional controller reduces error but does not eliminate it (unless the process has naturally integrating properties), i.e. an offset between the actual and desired value will normally exist.

A proportional integral algorithm

The mathematical representation is,

$$\frac{mv(s)}{e(s)} = k_c\left[1+\frac{1}{T_i s}\right] \text{ or } mv(t) = mv_{ss} + k_c\left[e(t)+\frac{1}{T_i}\int e(t)dt\right] \quad (4)$$

The additional integral mode (often referred to as reset) corrects for any offset (error) that may occur between the desired value (setpoint) and the process

[1] This is defined as the range over which the error must change in order to drive the controller output over full range. The PB also tells you how large the error has to be before the manipulated variable reaches 0 or 100%. The PB is generally centered around the setpoint causing the output to be at 50% when the setpoint and the process output are equal.

output automatically over time[2]. The adjustable parameter to be specified is the integral time (Ti) of the controller.

Where does the term reset come from?

Reset is often used to describe the integral mode. Reset is the time it takes for the integral action to produce the same change in mv as the P modes initial (static) change. Consider the following figure,

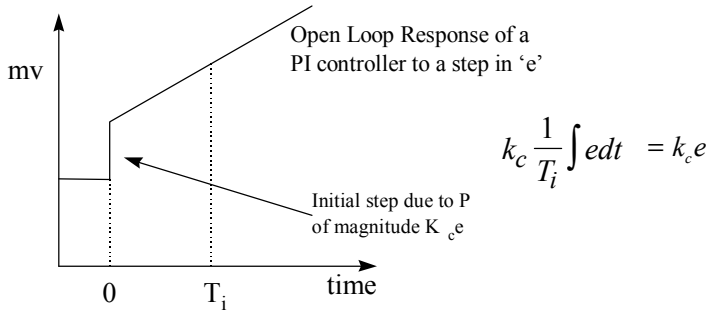


Figure (1) The response of a PI algorithm to a step in error

Figure (1) shows the output that would be obtained from a PI controller given a step change in error. The output immediately steps due to the P mode. The magnitude of the step up is $K_c e$. The integral mode then causes the mv to 'ramp'. Over the period 'time 0 to time T_I' the mv again increases by $K_c e$.

Integral wind-up

When a controller that possesses integral action receives an error signal for significant periods of time the integral term of the controller will increase at a rate governed by the integral time of the controller. This will eventually cause the manipulated variable to reach 100 % (or 0 %) of its scale, i.e. its maximum or minimum limits. This is known as integral wind-up. A sustained error can occur due to a number of scenarios, one of the more common being control system 'override'. Override occurs when another controller takes over control of a particular loop, e.g. because of safety reasons. The original controller is not switched off, so it still receives an error signal, which through time, 'winds-up' the integral component unless something is done to stop this occurring. There are many techniques that may be used to stop this

[2] Different control manufacturers use different definitions for the integral mode of a controller. It can be defined as minutes, minutes/repeat or repeats per minute. The difference is very important to note so as to ensure problems do not occur during a tuning exercise. *Remember the 'name game'*. T_i is the integral time (minutes), if specified as repeats / minute then it is $1/T_i$ that must be entered into the controller, while minutes / repeat is again T_i. This is confusing and is compounded by the fact that manufacturers are not consistent !

happening. One method is known as ‘external reset feedback’ (Luyben, 1990). Here, the signal of the control valve is also sent to the controller. The controller possess logic that enables it to integrate the error when its signal is going to the control value, but breaks the loop if the override controller is manipulating the valve.

A Proportional Integral Derivative algorithm

The mathematical representation is,

$$\frac{mv(s)}{e(s)} = k_c\left[1+\frac{1}{T_i s}+T_D s\right] \text{ or } mv(t) = mv_{ss} + k_c\left[e(t)+\frac{1}{T_i}\int e(t)dt + T_D\frac{de(t)}{dt}\right] \qquad (5)$$

Derivative action (also called rate or pre-act) *anticipates* where the process is heading by looking at the time rate of change of the controlled variable (its derivative). T_D is the ‘rate time’ and this characterises the derivative action (with units of minutes). In theory derivative action should always improve dynamic response and it does in many loops. In others, however, the problem of noisy signals makes the use of derivative action undesirable (differentiating noisy signals can translate into excessive mv movement).

Derivative action depends on the slope of the error, unlike P and I. If the error is constant derivative action has no effect.

Revision Exercise

Use Matlab / Simulink to explore the effect a step change in error has on the various modes of an ideal PID control algorithm. Assume that k_c = 1, T_i = 10 mins and T_D = 5mins.

PID algorithms can be different

Not all manufactures produce PID’s that conform to the ideal 'textbook' structure. So before commencing tuning it is important to know the configuration of the PID algorithm! The majority of ‘text-book’ tuning rules are only valid for the ideal architecture. If the algorithm is different then the controller parameters suggested by a particular tuning methodology will have to be altered.

Ideal PID

The mathematical representation of this algorithm is:

$$\frac{mv(s)}{e(s)} = k_c\left[1 + \frac{1}{T_i s} + T_D s\right]$$

One disadvantage of this ideal 'textbook' configuration is that a sudden change in setpoint (and hence e) will cause the derivative term to become very large and thus provide a “derivative kick” to the final control element - this is undesirable. An alternative implementation is

$$mv(s) = k_c\left[1 + \frac{1}{T_i s}\right]e(s) + T_D s cv(s)$$

The derivative mode acts on the measurement and not the error. After a change in setpoint the output will move slowly avoiding "derivative kick" after setpoint changes. This is therefore a standard feature of most commercial controllers.

Series (interacting) PID

The mathematical representation of this algorithm is:

$$\frac{mv(s)}{e(s)} = k_c\left[1 + \frac{1}{T_i s}\right]T_D s$$

As with the ideal implementation the series mode can include either derivative on the error or derivative on the measurement. In which case, the mathematical representation is,

$$\frac{mv(s)}{e(s)} = k_c\left[1 + \frac{1}{T_i s}\right] \quad where\ e(s) = SP - T_D s cv(s)$$

Parallel PID

The mathematical description is,

$$mv(s) = k_c e(s) + \frac{1}{T_i s}e(s) + T_D s e(s)$$

The proportional gain only acts on the error, whereas with the ideal algorithm it acts on the integral and derivative modes as well.

Revision Exercises

1. Draw the block diagram representation of the ideal, series (interacting) and parallel PID control laws.
2. Write down the 'time-domain' mathematical representation of the ideal (without derivative kick) , series (interacting) and parallel PID control laws.

3. Suppose that the controller settings for an ideal PID algorithm are given by, k_c, T_i, T_D. Work out the conversion factors required to ensure that a parallel implementation of the PID algorithm will provide the same mv signal given the same error signal.

Controller tuning

Controller tuning involves the selection of the best values of k_c, Ti and T_D (if a PID algorithm is being used). This is often a subjective procedure and is certainly process dependent. A number of methods have been proposed in the literature over the last 50 years. However, recent surveys indicate,

- 30 % of installed controllers operate in manual.
- 30 % of loops increase variability.
- 25 % of loops use default settings.
- 30 % of loops have equipment problems.

A possible explanation for this is lack of understanding of process dynamics, lack of understanding of the PID algorithm or lack of knowledge regarding effective tuning procedures. This section of the notes concentrates on PID tuning procedures. The suggestion being that if a PID can be properly tuned there is much scope to improve the operational performance of chemical process plant.

When tuning a PID algorithm, generally the aim is to match some preconceived 'ideal' response profile for the closed loop system. The following response profiles are typical.

Servo Control

For a unit step change in setpoint (0 - 1) the two response profiles shown in figure 2 could be obtained (depending upon the process dynamics and controller settings),

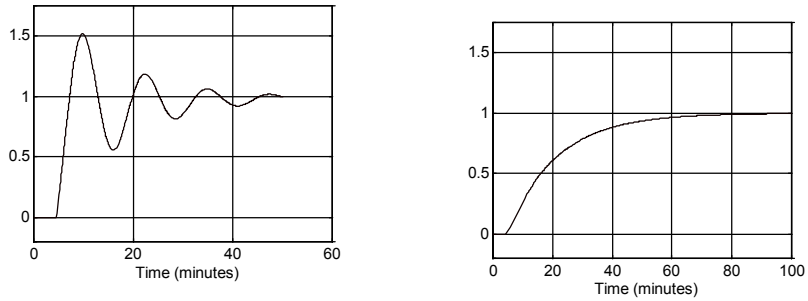


Figure (2) Underdamped (LHS) and overdamped (RHS) system response to a unit change in setpoint (PI control).

Terms used to describe underdamped response characteristics are,

- **Overshoot:** this is the magnitude by which the controlled variable 'swings' past the setpoint. 5/10% overshoot is normally acceptable for most loops.
- **Rise time:** the time it takes for the process output to achieve the new desired value. One-third the dominant process time constant would be typical.
- **Decay ratio:** this is the ratio of the maximum amplitude of successive oscillations.
- **Settling time:** the time it takes for the process output to die to between, say +/- 5% of setpoint.

These characteristics are often used as objectives during a tuning exercise.

Regulatory Control

For a unit step change in the dv, the following type of response profile may be desired,

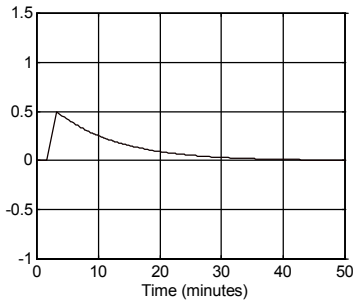


Figure (3) Disturbance rejection (a typical response profile)

i.e. the disturbance initially causes the process to move away from the desired value (which is set to zero in this figure). The controller then adjusts the mv so that the cv slowly moves back to setpoint. In other words the impact that the disturbance has on the closed loop system is eliminated and the system returns to the desired value. A transfer function that could be used to model this behaviour is,

$$\frac{cv(s)}{dv(s)} = \frac{\lambda s}{\lambda s + 1} \qquad (6)$$

where the constant λ models the 'peak' effect of the disturbance as well as the speed at which the system returns to steady-state.

Tuning Rules

Rules of thumb

The following rules of thumb are intended to give “ball-park” figure controller settings. The settings[1] assume a series algorithm, the others are for ideal PID

Loop Type	PB(%)	I (mins)	D (mins)
Liquid level	**< 100**	**10**	**-**
Temperature	**20 - 60**	**2 - 15**	**I/4**
Flow	**150**	**0.1**	**-**
Liquid Pressure[1]	**50 - 500**	**0.005 - 0.5**	**-**
Gas Pressure[1]	**1- 50**	**0.1 - 50**	**0.02 - 0.1**
Chromatograph[1]	**100 - 2000**	**10 - 120**	**0.1 - 20**

Often, with level systems exact setpoint following is not essential, hence proportional control is often used. Temperature loop dynamics can be slow because of process heat transfer lags. Deadtime is possible, especially in heat exchangers and temperature is not normally noisy. Consequently PID control is normally preferred. Flow loop dynamics are generally fast (of the order of seconds). Control valve dynamics are normally the slowest in the loop. Flow systems are noisy. However, noise can often be dealt with simply by reducing the gain.

Ziegler Nichols closed loop method

The method is straightforward. First, set the controller to P mode only. Next, set the gain of the controller (k_c) to a small value. Make a small setpoint (or load) change and observe the response of the controlled variable. If k_C is low the response should be sluggish. Increase k_C by a factor of two and make another small change in the setpoint or the load. Keep increasing k_C (by a factor of two) until the response becomes oscillatory. Finally, adjust k_C until a response is obtained that produces continuous oscillations. This is known as the ultimate gain (ku). Note the period of the oscillations (Pu). The control law settings are then obtained from the following table,

	k_C	T_i	T_D
P	ku/2		
PI	Ku/2.2	Pu/1.2	
PID	Ku/1.7	Pu/2	Pu/8

Practical use of the technique

It is unwise to force the system into a situation where there are continuous oscillations as this represents the limit at which the feedback system is stable. Generally, it is a good idea to stop at the point where some oscillation has been obtained. It is then possible to approximate the period (Pu) and if the gain at this point is taken as the ultimate gain (ku) , then this will provide a more conservative tuning regime.

Cohen - Coon

This method depends upon the identification of a suitable process model (plant identification has been covered in previous lectures). Cohen-Coon recommended the following settings to give responses having ¼ decay ratios, minimum offset and other favourable properties,

	k_c	T_i	T_D
P	$\frac{1}{k_p}\frac{\tau}{\theta}(1+\frac{\theta}{3\tau})$		
PI	$\frac{1}{k_p}\frac{\tau}{\theta}(\frac{9}{10}+\frac{\theta}{12\tau})$	$\theta\frac{30+3(\theta/\tau)}{9+20(\theta/\tau)}$	
PID	$\frac{1}{k_p}\frac{\tau}{\theta}(\frac{4}{3}+\frac{\theta}{4\tau})$	$\theta\frac{32+6(\theta/\tau)}{13+8(\theta/\tau)}$	$\theta\frac{4}{11+2(\theta/\tau)}$

In the table k_p is the process gain, τ the process time constant and θ the process time delay.

Practical use of the technique

If the process delay is small (in the limit as it approaches zero) increasingly large controller gains will be predicted. The method is therefore not suitable for systems where there is zero or virtually no time delay.

Direct synthesis

This is a model based tuning technique. It uses an identified process model in conjunction with a user specified closed loop response characteristic. An advantage of this approach is that it provides insight into the role of the 'model' in control system design. A disadvantage of the approach is that a PID controller may not be realised unless an appropriate model form is used to synthesise the control law.

Tuning for servo control

Let the symbol Gp represent the process dynamics and Gc the controller dynamics. If all other dynamic elements within the loop are ignored then the following closed loop transfer function can be derived,

$$\frac{cv}{SP} = \frac{G_c G_p}{1 + G_c G_p} \tag{6}$$

this can be re-arranged to give an expression for the feedback control law as,

$$G_c = \frac{1}{G_p}\left(\frac{\frac{cv}{SP}}{1-\frac{cv}{SP}}\right) \tag{7}$$

In other words, the controller comprises the inverse of the process model (common to model based design techniques) as well as a specification for the closed loop response characteristic, cv/SP.

A process model can be obtained through plant identification. The closed loop response characteristic, cv/SP must be specified. A simple specification is,

$$\frac{cv}{SP} = \frac{1}{\lambda s + 1} \tag{8}$$

λ is a user specified closed loop time constant.

Substituting this into equation (7) and re-arranging gives,

$$G_c = \frac{1}{G_p}\left(\frac{1}{\lambda s}\right) = \frac{\tau_p s + 1}{k_p \lambda s} = \frac{\tau_p}{k_p \lambda}\left(1 + \frac{1}{\tau_p s}\right) \tag{9}$$

where it has been assumed that the process transfer function is,

$$G_p(s) = \frac{k_p}{\tau_p s + 1} \tag{10}$$

ie. first order, no dead-time.

Based on this process description, the ideal form of a PI controller results, where,

$$k_c = \frac{\tau_p}{k_p \lambda} \text{ and } T_i = \tau_p \tag{11}$$

What do you do if you want derivative action? The first order model results in a control law that is of the PI type. If you wish to synthesis a PID controller, there are two options

- choose T_D = Ti/4

- model the process using a 2nd order transfer function.

Revision Exercise

Starting with a second order process transfer function show that a PID control structure can be developed using the direct synthesis derivation technique. What are the settings of the PID controller (in terms of the coefficients of the second order process transfer function) ?

Systems with time delay

Throughout this course, our basic assumption has been that we can model systems using the following transfer function,

$$G_p(s) = \frac{k_p e^{-s\theta}}{\tau_p s + 1} \qquad (12)$$

i.e. a first order plus dead-time transfer function. If this were the case, what type of control law would result using the direct synthesis procedure?

Following the derivation presented, the following control law results**,**

$$G_c = \frac{1}{G_p}\left(\frac{e^{-s\theta}}{\lambda s + 1 - e^{-s\vartheta}}\right) \qquad (13)$$

Note that the following response specification was used (as the time delay cannot be removed from the process),

$$\frac{cv}{SP} = \frac{e^{-s\theta}}{\lambda s + 1} \qquad (14)$$

The control law, equation (13) is of non-standard form because of the time-delay terms. Suppose that $e^{-s\theta}$ is approximated by a 1st order Taylor series expansion, i.e.

$$e^{-s\theta} \approx 1 - \theta s$$

Substituting into the denominator of equation (13) and re-arranging gives,

$$G_c = \frac{1}{G_p}\left(\frac{e^{-s\theta}}{(\lambda + \theta)s}\right) \qquad (15)$$

It is not necessary to approximate the time delay in the numerator of equation (13) as this is cancelled by an identical term in the process transfer function, $G_p(s)$ giving,

$$G_c = \frac{\tau_p s + 1}{k_p(\lambda + \theta)s} = \frac{\tau_p}{k_p(\lambda + \theta)}\left(1 + \frac{1}{\tau_p s}\right) \qquad (16)$$

which is the form of an ideal PI controller where,

$$k_c = \frac{\tau_p}{k_p(\theta + \lambda)} \text{ and } T_i = \tau_p \quad (17)$$

Note the intuitive nature of the controller gain calculation: as the process time delay increases the controller gain will decrease.

Tuning for regulatory control

With reference to the closed loop block diagram, for regulatory control the following closed loop transfer function may be derived,

$$\frac{cv}{dv} = \frac{1}{1 + G_c G_p} \quad (18)$$

This closed loop expression can be re-arranged to give an expression for the feedback control law as,

$$G_c = \frac{1}{G_p}\left(\frac{1 - \frac{Y}{d}}{\frac{Y}{d}}\right) \quad (19)$$

Again, the controller consists of the inverse of the process model as well as a specification for the closed loop response characteristic, cv/dv.

The process model is obtained through plant identification however, the closed loop response characteristic, cv/dv, must be specified by the designer. Using the simple specification described earlier,

$$\frac{cv(s)}{dv(s)} = \frac{\lambda s}{\lambda s + 1} \quad (20)$$

where λ is user specified. Substituting this into equation (19) and re-arranging gives,

$$G_c = \frac{1}{G_p}\left(\frac{1}{\lambda s}\right) \quad (21)$$

This is exactly the same form as equation (9) for servo control. Hence the controller gain and integral term for a PI controller is given by,

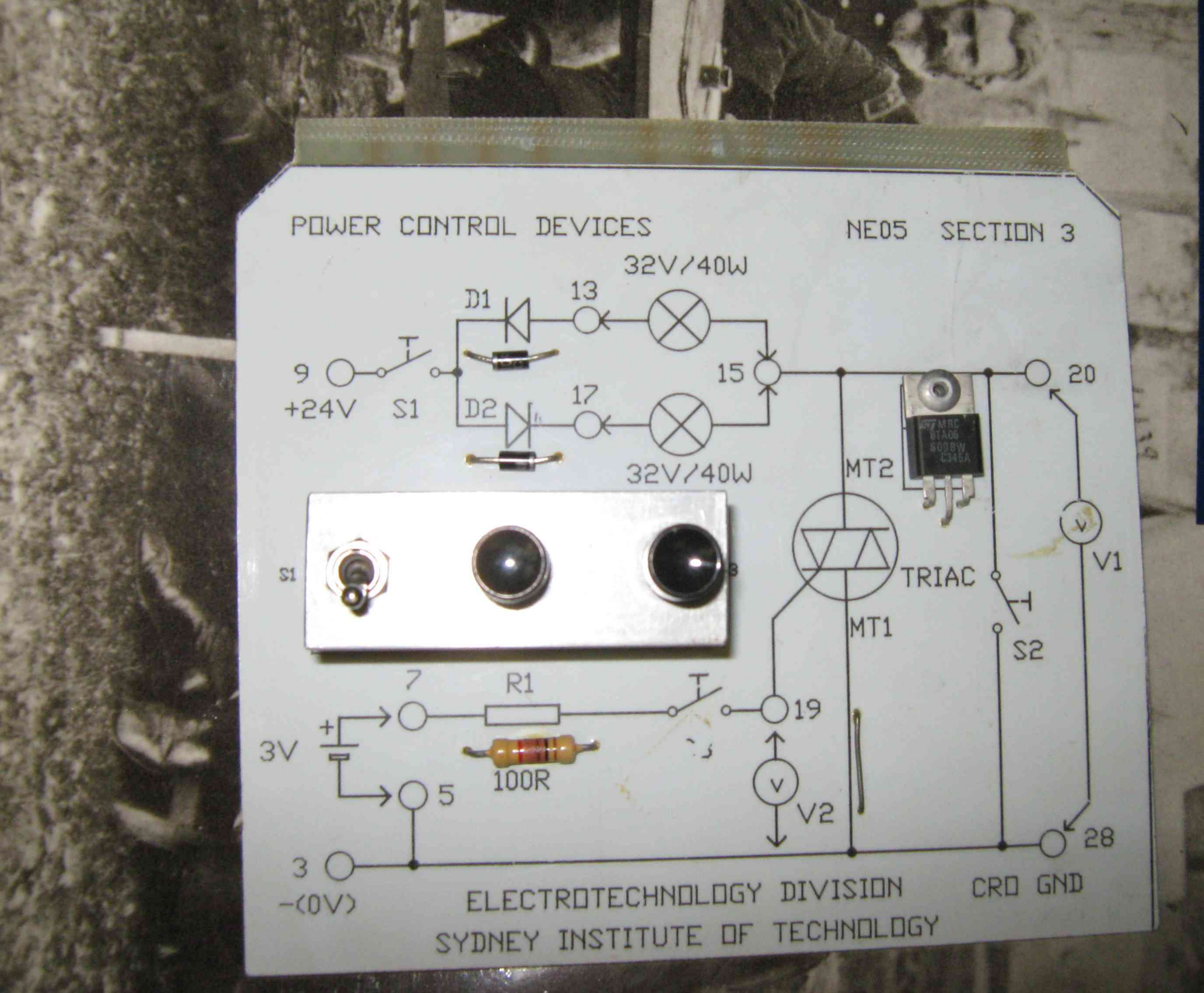
$$k_c = \frac{\tau_p}{k_p \lambda} \text{ and } T_i = \tau_p \quad (22)$$

Final Remarks

The notes have reviewed PID control, discussed the modes of the various control algorithms, the different structures of algorithms that exist and standard tuning rules. The tuning rules reviewed include, Ziegler-Nichols, Cohen- Coon, and direct synthesis. Remember:

- the tuning rules are only valid for the 'ideal' PID control structure and any prediction of control law settings should be adjusted if an alternative PID implementation is used.
- the tuning rules are only valid for self-regulating processes (i.e open loop stable processes such as those that may be described by the 1^{st} order plus dead-time description).

Luckily most process systems are self-regulating the exception to the rule being level systems. Tuning of level controllers will be the subject of the next section of the notes.

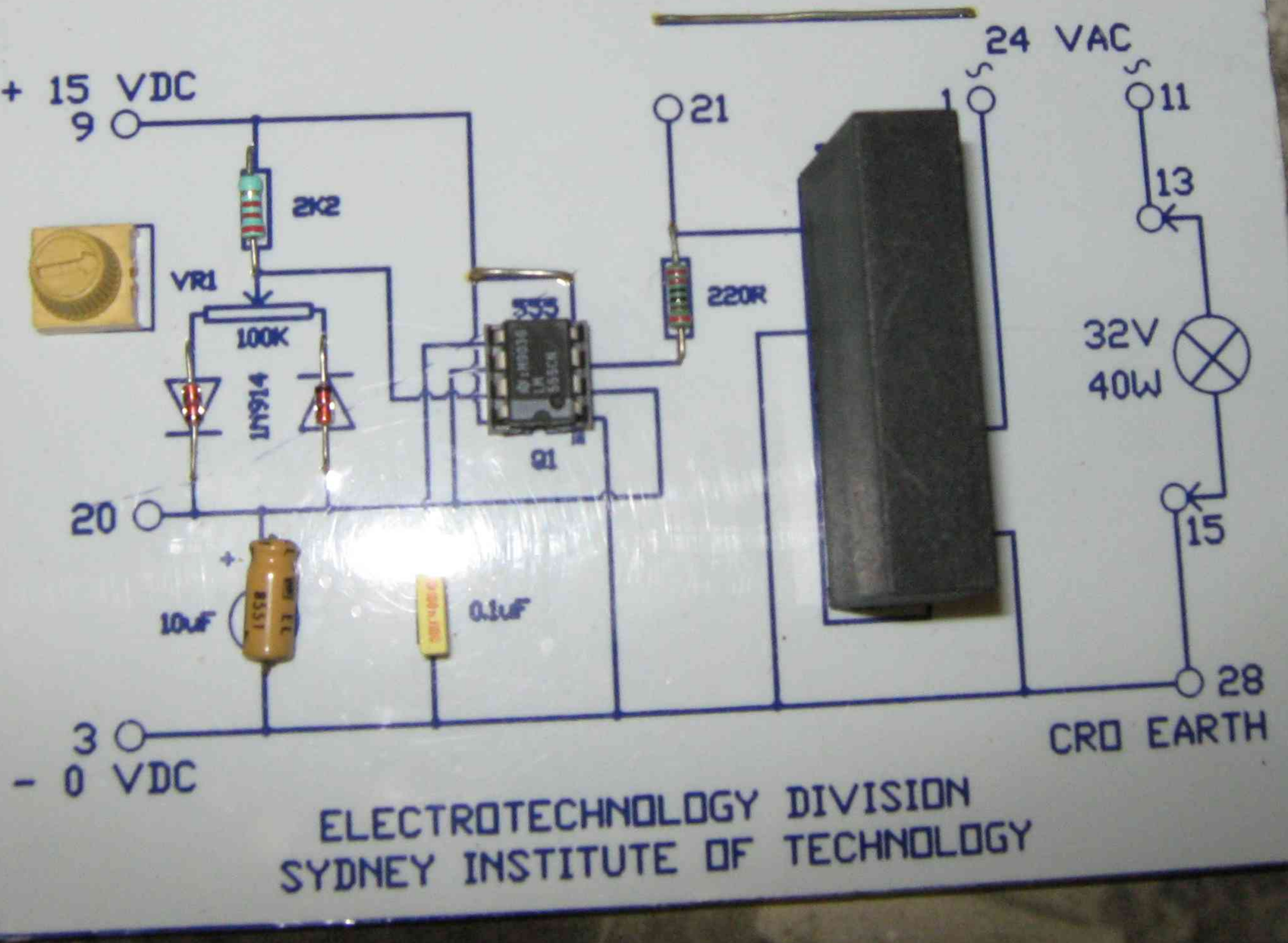


WAR

Library of Congress

POWER CONTROL DEVICES NE05 SECTION 7

ZERO VOLTAGE SWITCHING



ELECTROTECHNOLOGY DIVISION

SYDNEY INSTITUTE OF TECHNOLOGY

WAR

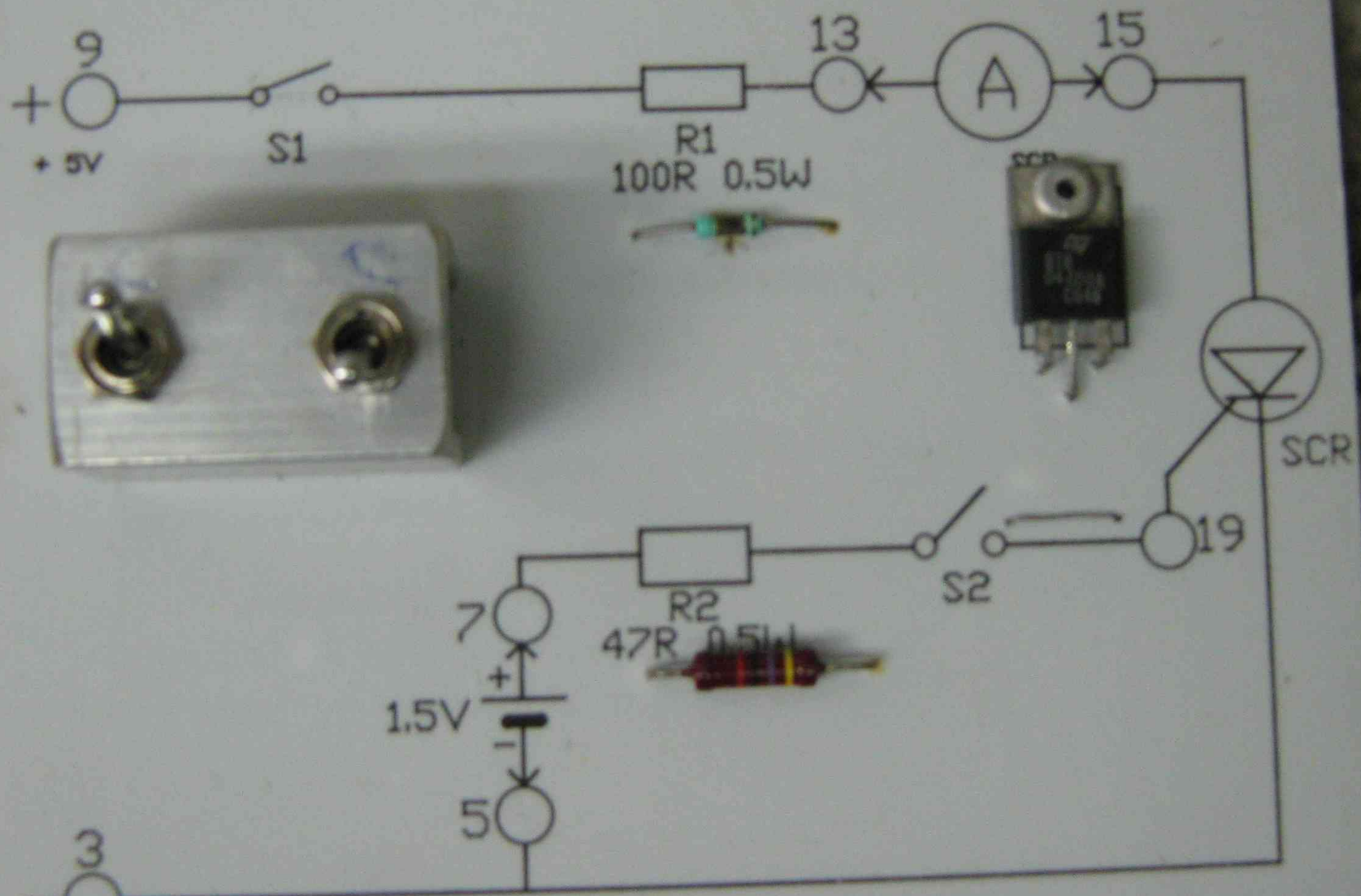
ry of Congress

07 Calendar

NE05 - 7.

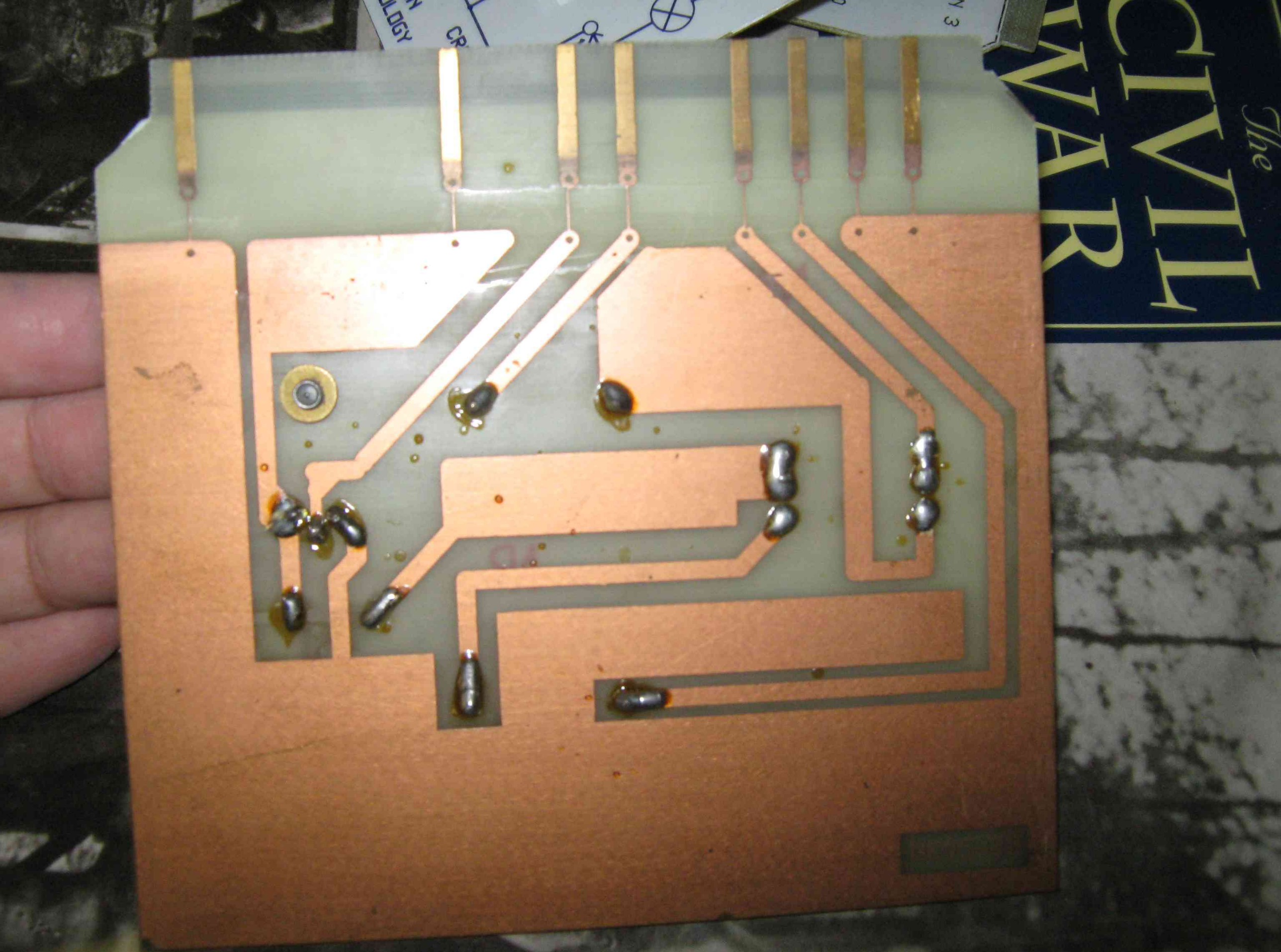
POWER CONTROL DEVICES

NE05 SECTION 2B

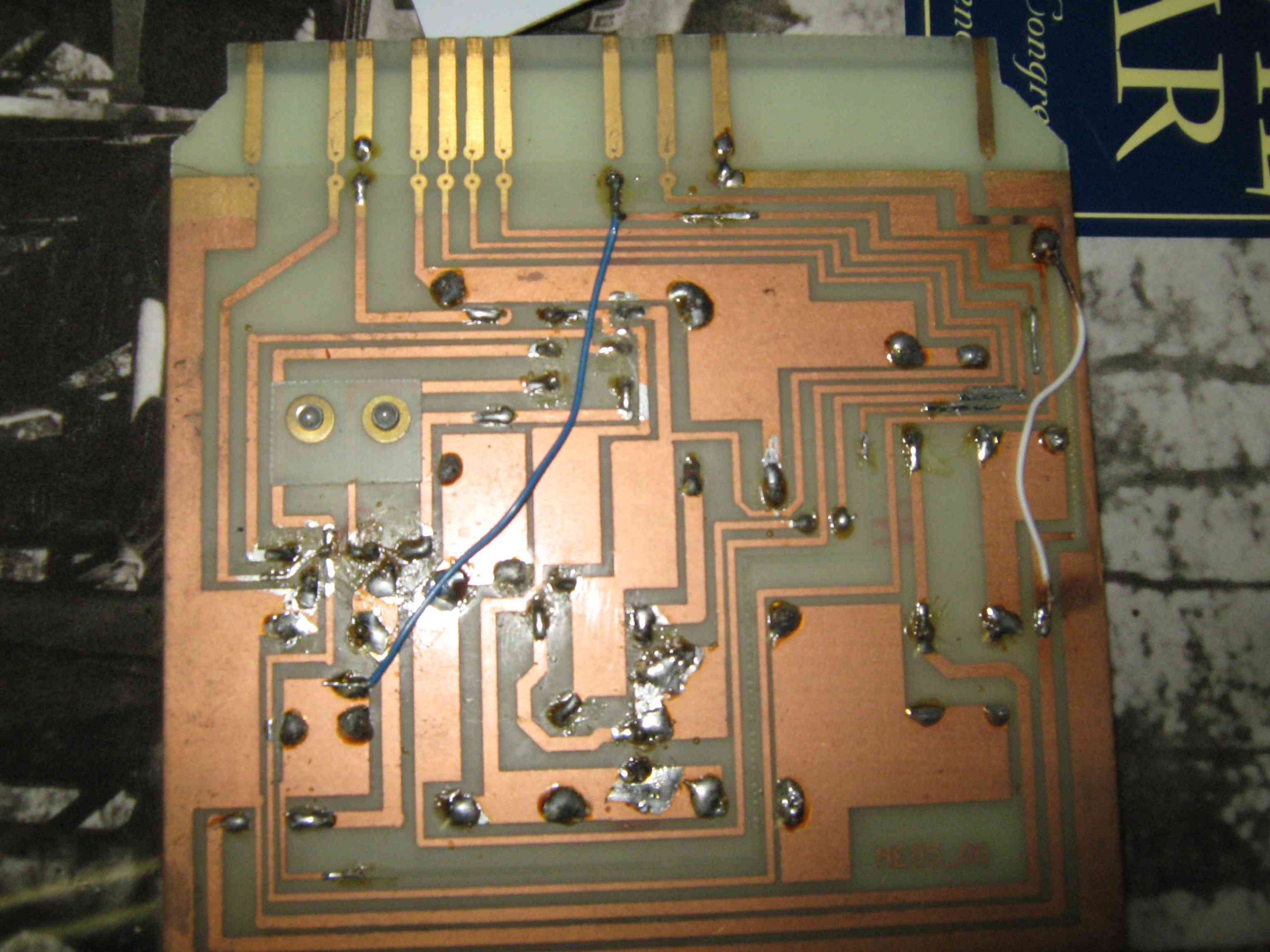


ELECTROTECHNOLOGY DIVISION

SYDNEY INSTITUTE OF TECHNOLOGY

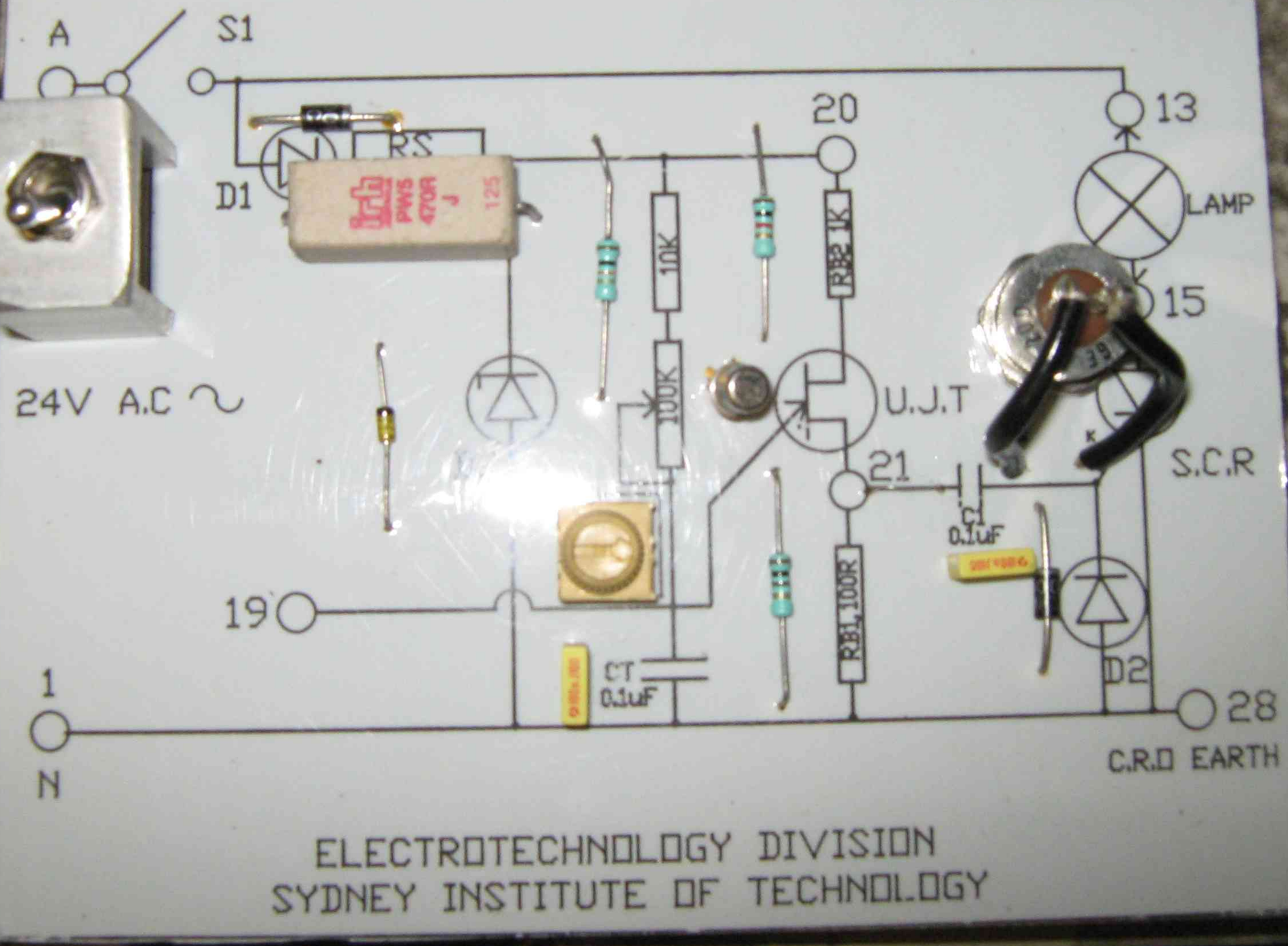

The
CIVIL
WAR


POWER CONTROL DEVICES
NE05 SECTION 6
ELECTROTECHNOLOGY DIVISION
SYDNEY INSTITUTE OF TECHNOLOGY
11
22
24
25
LINK
SCR1
SCR2
13
15
1
28
19
20
21
D1
D2
D3
D4
D5
D6
ZD1
10K
12K
100K
18K
0.1uF
PW5
470R
J
C106D

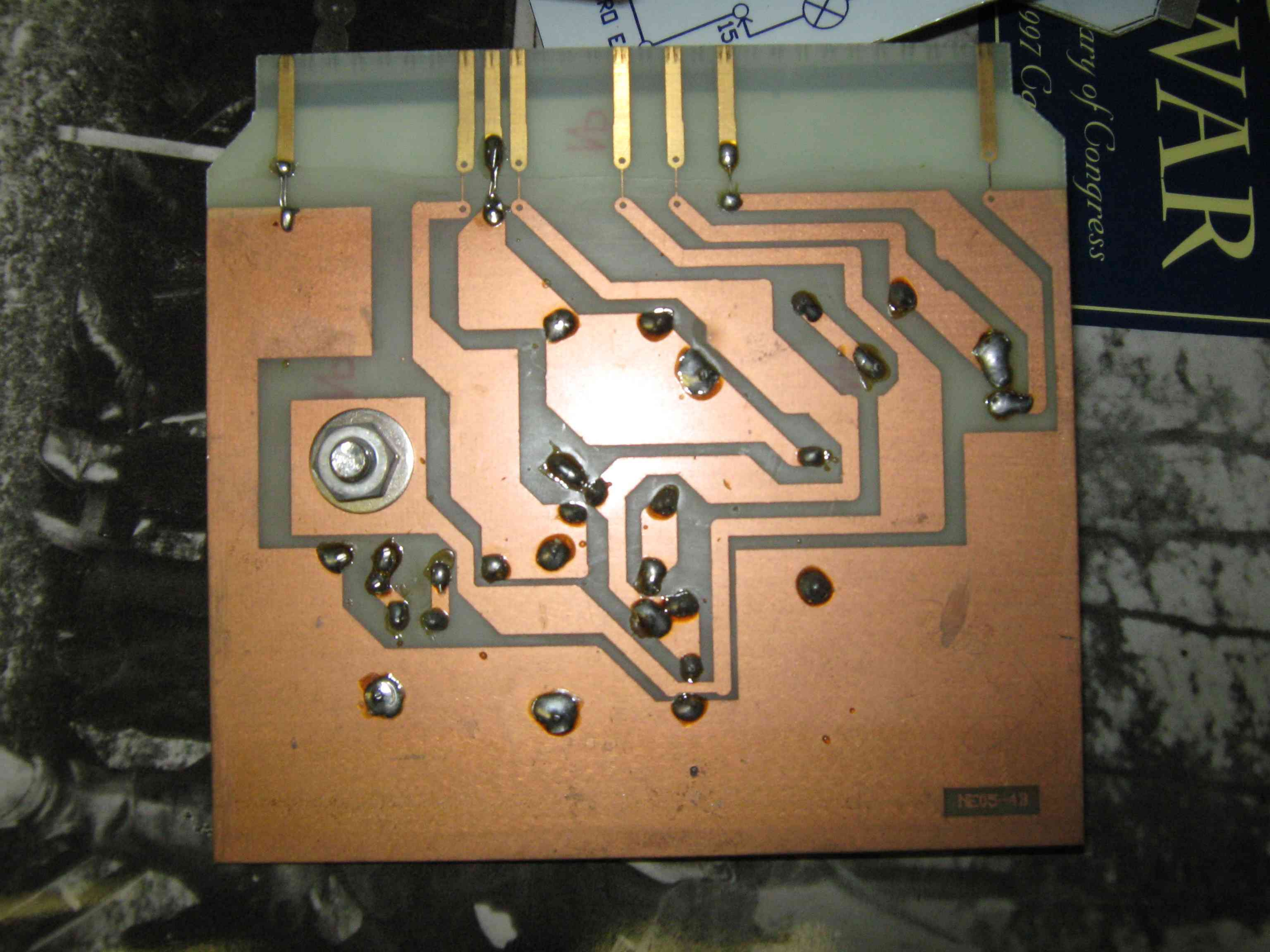


POWER CONTROL DEVICES NE05_4B

U.J.T TRIGGER PULSE GENERATOR - MAINS SYNCHRONISATION

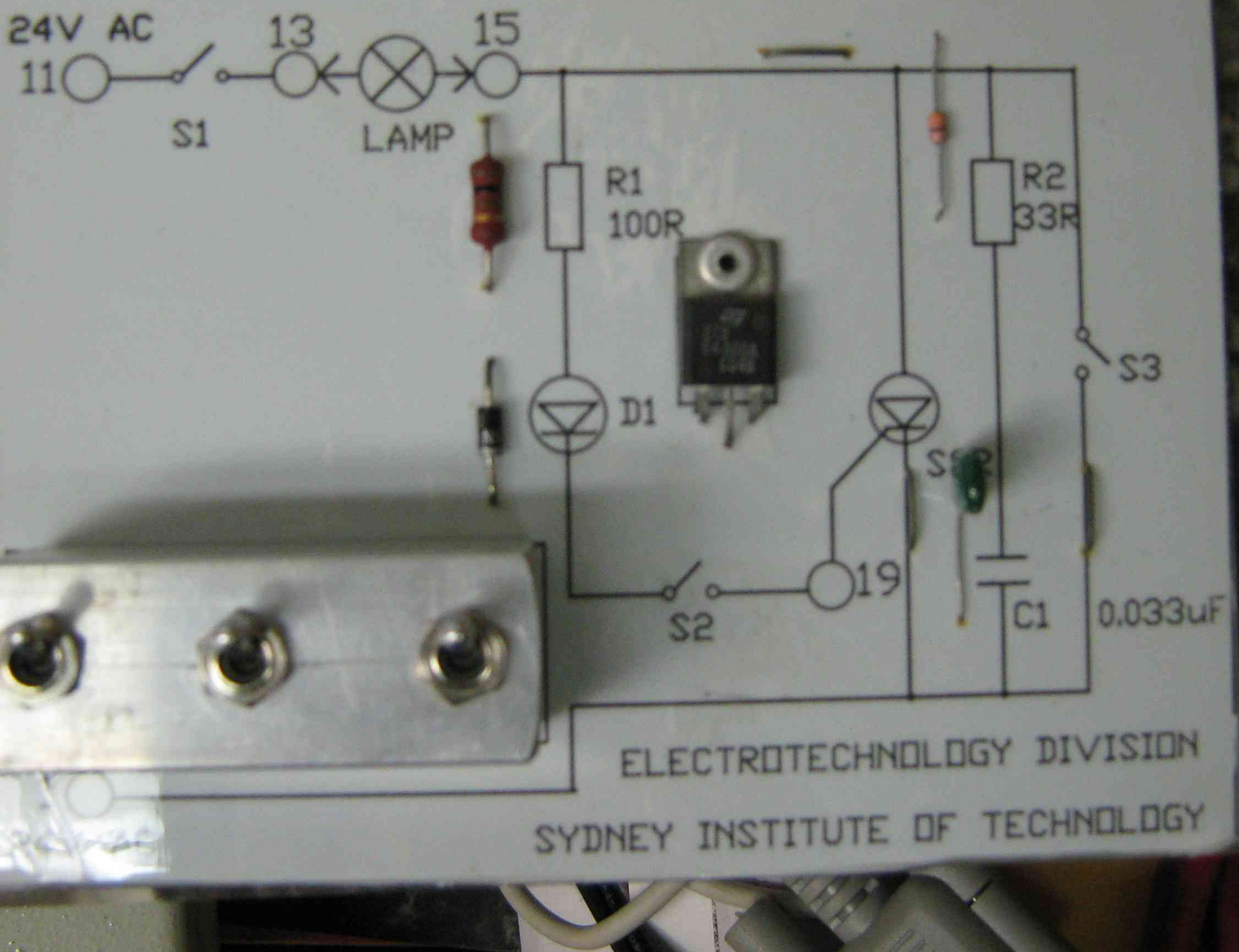


ELECTROTECHNOLOGY DIVISION
SYDNEY INSTITUTE OF TECHNOLOGY



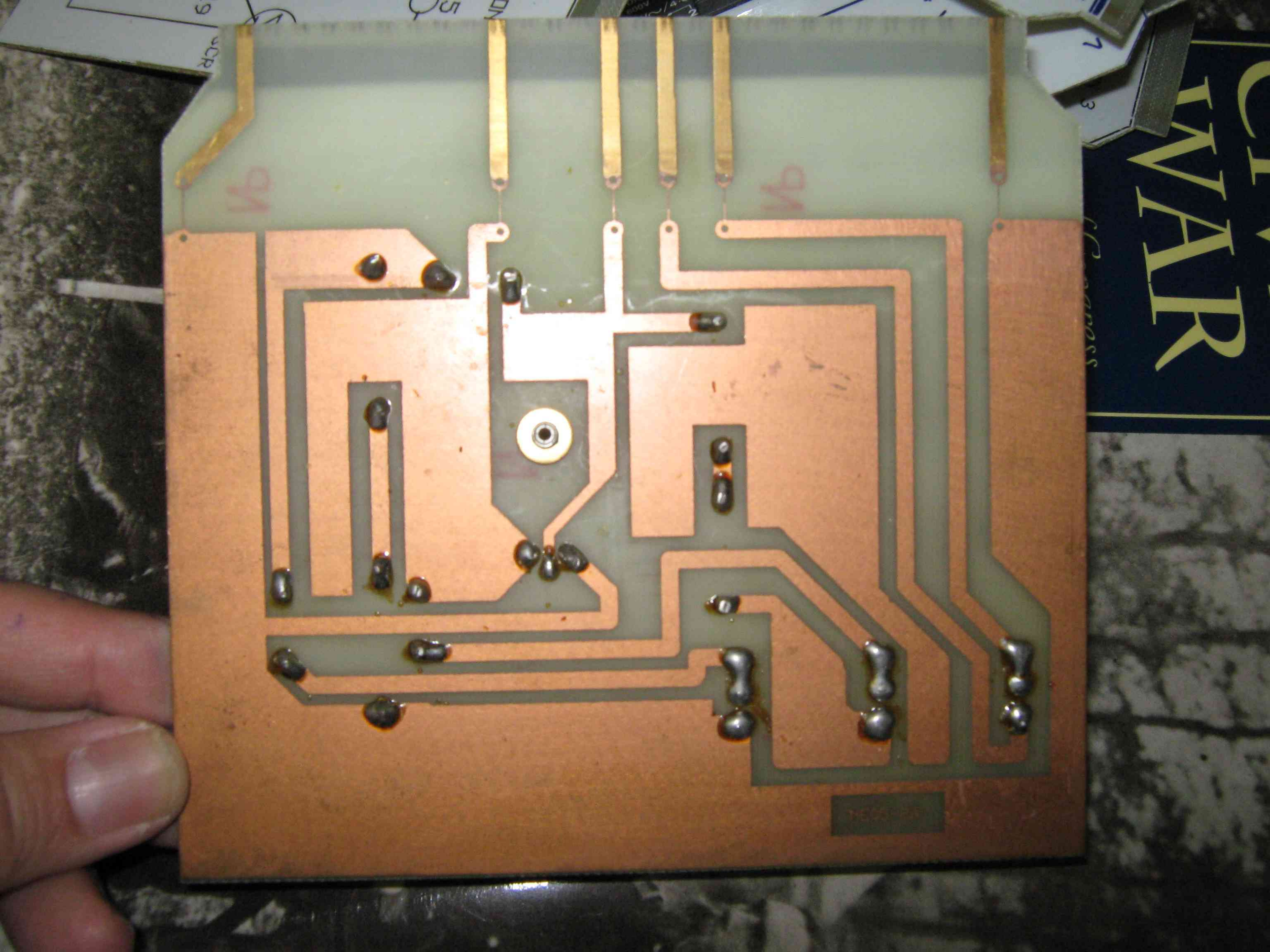
POWER CONTROL DEVICES

NE05 SECTION 2A



ELECTROTECHNOLOGY DIVISION

SYDNEY INSTITUTE OF TECHNOLOGY



MPLAB® IDE User's Guide with MPLAB Editor and MPLAB SIM Simulator

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Table of Contents

Chapter 13. Windows

Chapter 14. Dialogs

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXA", where "XXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE on-line help. Select the Help menu, and then Topics to open a list of available on-line help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using MPLAB IDE. Items discussed include:

- Document Layout
- Conventions Used in this Guide
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support

DOCUMENT LAYOUT

This document describes how to use the MPLAB IDE as a development tool to emulate and debug firmware on a target board. The manual layout is as follows:

Part 1 – MPLAB IDE Overview

- **Chapter 1: What is MPLAB IDE?** – Describes MPLAB IDE and how it can help develop an application.
- **Chapter 2: Integrated Language Tools** – Describes the language tools (assemblers, compilers), software tools and hardware tools that may be used with MPLAB IDE.
- **Chapter 3: Integrated Software/Hardware Tools** – Describes the integrated software and hardware tools (editors, simulators, debuggers, etc.) of MPLAB IDE.

Part 2 – MPLAB IDE Tutorials

- **Chapter 4: A Basic Tutorial for MPLAB IDE** – How to install MPLAB IDE software and how to use the software to develop an example application.
- **Chapter 5: Walk-Through and Detailed Tutorial** – Walks through the necessary steps to develop an application using MPLAB IDE. An example is given with each step.

Part 3 – MPLAB IDE Features

- **Chapter 6: Projects and Workspaces** – Describes the use of MPLAB IDE Projects and Workspaces when developing an application. Includes information on the Project Wizard, version control systems and single and multiple file projects.
- **Chapter 7: Programming Language Features –** Describes how to set up language tools (assemblers, compilers, linkers, etc.) for use with MPLAB IDE.
- **Chapter 8: Debug Features –** Describes the built-in debug features of MPLAB IDE. Includes execution control, trace, and Watch window operation.
- **Chapter 9: Device-Related Features –** Describes features that mirror device components. Includes windows and/or dialogs for configuration bit, program and data memory, external memory, hardware stack, ID memory, and peripheral-specific support.
- **Chapter 10: MPLAB Macros –** Describes the support MPLAB IDE gives to the creation of macros of macros, using macros, the macros menu and toolbar and the macros dialog.

Part 4 – MPLAB IDE Reference

- **Chapter 11: Troubleshooting –** Describes common problems and solutions with MPLAB IDE operation.
- **Chapter 12: Desktop –** Describes the MPLAB IDE desktop, including menu bar, toolbars and status bar.
- **Chapter 13: Windows –** Describes all MPLAB IDE windows. Includes window symbols definitions.
- **Chapter 14: Dialogs –** Describes all MPLAB IDE dialogs.
- **Chapter 15: Operational Reference –** Miscellaneous information on command-line options, shortcut (hot) keys, files used by MPLAB IDE and portability information.

Part 5 – MPLAB Editor

- **Chapter 16: Using the Editor –** Describes how to use MPLAB Editor. Includes text handling, configuring the editor, working with files and working with text. Additional information includes keyboard features, editor context (right mouse) menu and troubleshooting.

Part 6 – MPLAB SIM Simulator

- **Chapter 17: Simulator Overview** – An overview of MPLAB SIM simulator. Topics discussed are simulator features, models and execution.
- **Chapter 18: Getting Started with MPLAB SIM** – Describes getting started using MPLAB SIM. Tutorials are suggested, and simulator features are discussed.
- **Chapter 19: Using Stimulus** – Describes the use of simulator stimulus for most PICmicro microcontroller (MCU) and dsPIC digital signal controller (DSC) devices. Discusses stimulus creation with the SCL generator and stimulus control.
- **Chapter 20: Using Stimulus - PIC17 Devices**– Details simulator stimulus use for PIC17CXXX MCU devices. Discusses pin and file stimulus.
- **Chapter 21: Simulator Troubleshooting** – Describes common problems and solutions with MPLAB SIM operation.
- **Chapter 22: Simulator Reference** – Details functions available for use in debugging an application with the simulator.

CONVENTIONS USED IN THIS GUIDE

The following conventions may appear in this documentation:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF,'A'`
Italic	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`mpasmwin [options] file [options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

RECOMMENDED READING

This user's guide describes how to use MPLAB IDE. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

Readme for MPLAB IDE

For the latest information on using MPLAB IDE, read the "`Readme for MPLAB IDE.htm`" file (an HTML file) in the Readmes subdirectory of the MPLAB IDE installation directory. The Readme file contains update information and known issues that may not be included in this user's guide.

Readme Files

For the latest information on using other tools, read the tool-specific Readme files in the Readmes subdirectory of the MPLAB IDE installation directory. The Readme files contain update information and known issues that may not be included in this user's guide.

MPLAB IDE User's Guide (DS51519)

The user's guide is a comprehensive guide for MPLAB IDE, including MPLAB Editor and MPLAB SIM simulator. Tutorials, functional descriptions and reference material are included.

On-line Help Files

Comprehensive help files are available for MPLAB IDE, MPLAB Editor and MPLAB SIM simulator. Tutorials, functional descriptions and reference material are included.

Device Data Sheets and Family Reference Manuals

See the Microchip web site for complete and updated versions of device (PIC® MCU and dsPIC® DSC) data sheets and related device family reference manuals.

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers, assemblers, linkers and other language tools. These include all MPLAB C compilers; all MPLAB assemblers (including MPASM™ assembler); all MPLAB linkers (including MPLINK™ object linker); and all MPLAB librarians (including MPLIB™ object librarian).
- **Emulators** – The latest information on Microchip in-circuit emulators.These include the MPLAB REAL ICE™, MPLAB ICE 2000 and MPLAB ICE 4000 in-circuit emulators
- **In-Circuit Debuggers** – The latest information on Microchip in-circuit debuggers. These include the MPLAB ICD 2 and 3 in-circuit debuggers and PICkit™ 2 and 3 debug express.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB IDE Project Manager, MPLAB Editor and MPLAB SIM simulator, as well as general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 and PRO MATE II device programmers and the PICSTART® Plus and PICkit 1, 2 and 3 development programmers.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document. See our web site for a complete, up-to-date listing of sales offices.

Technical support is available through the web site at: http://support.microchip.com.

NOTES:

Part 1 – MPLAB IDE Overview

NOTES:

Chapter 1. What is MPLAB® IDE?

1.1 AN OVERVIEW OF EMBEDDED SYSTEMS

MPLAB IDE is a Windows® Operating System (OS) software program that runs on a PC to develop applications for Microchip microcontrollers and digital signal controllers. It is called an Integrated Development Environment, or IDE, because it provides a single integrated "environment" to develop code for embedded microcontrollers. Experienced embedded systems designers may want to skip ahead to **Section 1.7 "Components of MPLAB IDE"**. It is also recommended that **Section 1.9 "MPLAB IDE On-line Help"** and **Section 1.11 "MPLAB IDE Updates"** be reviewed. The rest of this chapter briefly explains embedded systems development and how MPLAB IDE is used.

1.1.1 Description of an "Embedded System"

An embedded system is typically a design making use of the power of a small microcontroller, like the Microchip PIC® MCU or dsPIC® Digital Signal Controller (DSCs). These microcontrollers combine a microprocessor unit (like the CPU in a desktop PC) with some additional circuits called "peripherals", plus some additional circuits on the same chip to make a small control module requiring few other external devices. This single device can then be embedded into other electronic and mechanical devices for low-cost digital control.

1.1.2 Differences Between an Embedded Controller and a PC

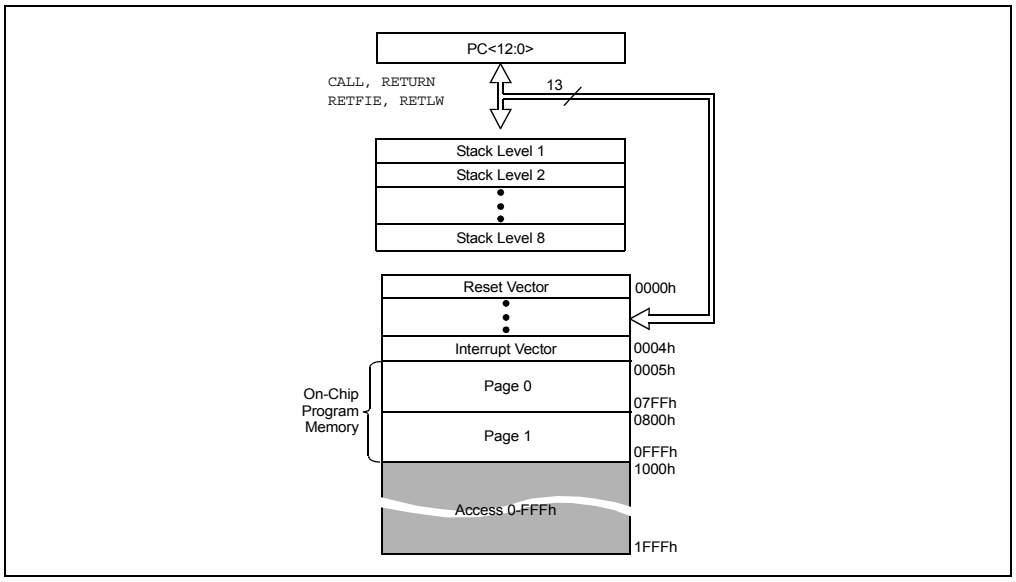
The main difference between an embedded controller and a PC is that the embedded controller is dedicated to one specific task or set of tasks. A PC is designed to run many different types of programs and to connect to many different external devices. An embedded controller has a single program and, as a result, can be made cheaply to include just enough computing power and hardware to perform that dedicated task. A PC has a relatively expensive generalized central processing unit (CPU) at its heart with many other external devices (memory, disk drives, video controllers, network interface circuits, etc.). An embedded system has a low-cost microcontroller unit (MCU) for its intelligence, with many peripheral circuits on the same chip, and with relatively few external devices. Often, an embedded system is an invisible part, or sub-module of another product, such as a cordless drill, refrigerator or garage door opener. The controller in these products does a tiny portion of the function of the whole device. The controller adds low-cost intelligence to some of the critical sub-systems in these devices.

An example of an embedded system is a smoke detector. Its function is to evaluate signals from a sensor and sound an alarm if the signals indicate the presence of smoke. A small program in the smoke detector either runs in an infinite loop, sampling the signal from the smoke sensor, or lies dormant in a low-power "sleep" mode, being awakened by a signal from the sensor. The program then sounds the alarm. The program would possibly have a few other functions, such as a user test function, and a low battery alert. While a PC with a sensor and audio output could be programmed to do the same function, it would not be a cost-effective solution (nor would it run on a nine-volt battery, unattended for years!). Embedded designs use inexpensive microcontrollers to put intelligence into the everyday things in our environment, such as smoke detectors, cameras, cell phones, appliances, automobiles, smart cards and security systems.

1.1.3 Components of a Microcontroller

The PIC MCU has on-chip program memory for the firmware, or coded instructions, to run a program (Figure 1-1). A Program Counter (PC) is used to address program memory, including reset and interrupt addresses. A hardware stack is used with call and return instructions in code, so it works with, but is not part of, program memory. Device data sheets describe the details of program memory operation, vectors and the stack.

FIGURE 1-1: PIC® MCU DATA SHEET – PROGRAM MEMORY AND STACK



The microcontroller also has data or "file register" memory. This memory consists of Special Function Registers (SFRs) and General Purpose Registers (GPRs) as shown in Figure 1-2. SFRs are registers used by the CPU and peripheral functions for controlling the desired operation of the device. GPRs are for storage of variables that the program will need for computation or temporary storage. Some microcontrollers have additional data EEPROM memory. As with program memory, device data sheets describe the details of data memory use and operation.

FIGURE 1-1: PIC® MCU DATA SHEET – FILE REGISTERS

	File Address		File Address		File Address		File Address
Indirect addr. (1)	00h	Indirect addr. (1)	80h	Indirect addr. (1)	100h	Indirect addr. (1)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	PORTA	105h	TRISA	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	PORTC	107h	TRISC	187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2(1)	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh		18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh		18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h		91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD(2)	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUA	95h	WPUB	115h		195h
CCPR1H	16h	IOCA	96h	IOCB	116h		196h
CCP1CON	17h	WDTCON	97h		117h		197h
RCSTA	18h	TXSTA	98h	VRCON	118h		198h
TXREG	19h	SPBRG	99h	CM1CON0	119h		199h
RCREG	1Ah	SPBRGH	9Ah	CM2CON0	11Ah		19Ah
	1Bh	BAUDCTL	9Bh	CM2CON1	11Bh		19Bh
PWM1CON	1Ch		9Ch		11Ch		19Ch
ECCPAS	1Dh		9Dh		11Dh	PSTRCON	19Dh
ADRESH	1Eh	ADRESL	9Eh	ANSEL	11Eh	SRCON	19Eh
ADCON0	1Fh	ADCON1	9Fh	ANSELH	11Fh		19Fh
General Purpose Register 96 Bytes	20h – 7Fh	General Purpose Register 80 Bytes	A0h – EFh	General Purpose Register 80 Bytes	120h – 16Fh		1A0h
accesses 70h-7Fh		accesses 70h-7Fh	F0h – FFh	accesses 70h-7Fh	170h – 17Fh	accesses 70h-7Fh	1F0h – 1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

Unimplemented data memory locations, read as '0'.

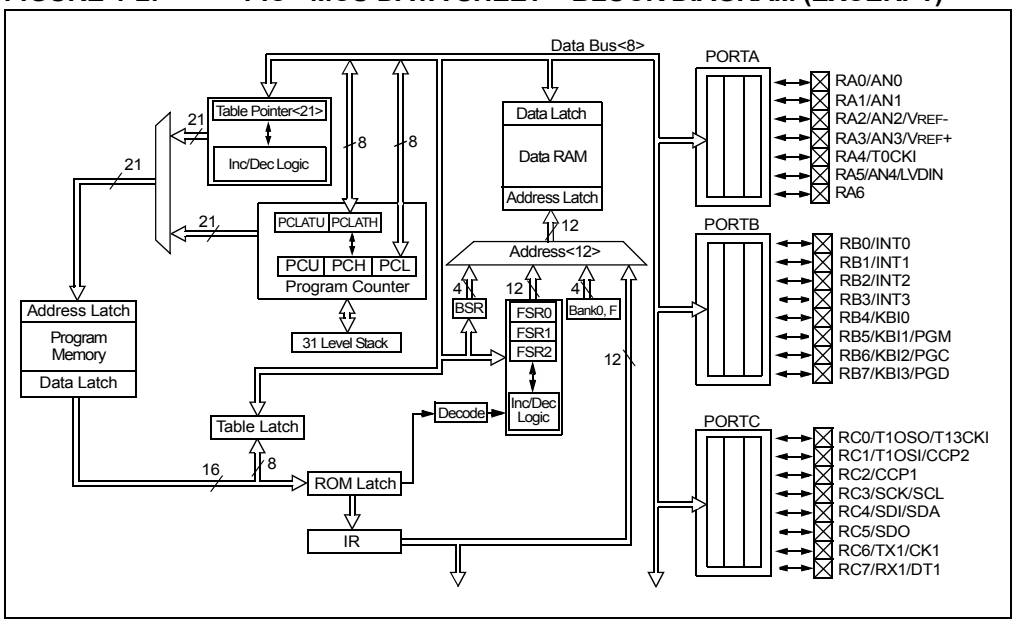
Note 1: Not a physical register.

2: Address 93h also accesses the SSP Mask (SSPMSK) register under certain conditions.

For more details, see the *PIC16F690 Data Sheet* (DS41262).

In addition to memory, the microcontroller has a number of peripheral device circuits on the same chip. Some peripheral devices are called input/output (I/O) ports. I/O ports are pins on the microcontroller that can be used as outputs and driven high or low to send signals, blink lights, drive speakers – just about anything that can be sent through a wire. Often these pins are bidirectional and can also be configured as inputs allowing the program to respond to an external switch, sensor or to communicate with some external device.

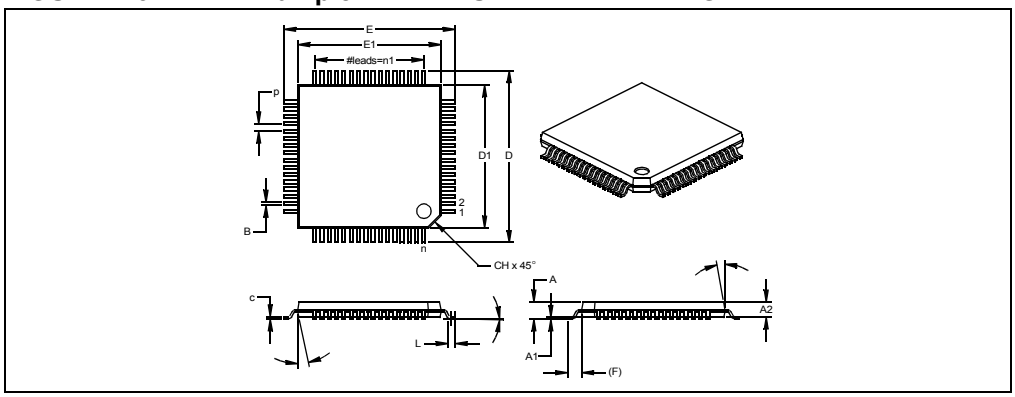
FIGURE 1-2: PIC® MCU DATA SHEET – BLOCK DIAGRAM (EXCERPT)



In order to design such a system, it must be decided which peripherals are needed for an application. Analog-to-Digital Converters (ADCs) allow microcontrollers to connect to sensors and receive changing voltage levels. Serial communication peripherals allow you to stream communications over a few wires to another microcontroller, to a local network or to the internet. Peripherals on the PIC MCU called "timers" accurately measure signal events and generate and capture communications signals, produce precise waveforms, even automatically reset the microcontroller if it gets "hung" or lost due to a power glitch or hardware malfunction. Other peripherals detect if the external power is dipping below dangerous levels so the microcontroller can store critical information and safely shut down before power is completely lost.

The peripherals and the amount of memory an application needs to run a program largely determines which PIC MCU to use. Other factors might include the power consumed by the microcontroller and its “form factor,” i.e., the size and characteristics of the physical package that must reside on the target design.

FIGURE 1-3: Example PIC® MCU DEVICE PACKAGE



1.1.4 Implementing an Embedded System Design with MPLAB IDE

A development system for embedded controllers is a system of programs running on a desktop PC to help write, edit, debug and program code – the intelligence of embedded systems applications – into a microcontroller. MPLAB IDE runs on a PC and contains all the components needed to design and deploy embedded systems applications.

The typical tasks for developing an embedded controller application are:

1. Create the high level design. From the features and performance desired, decide which PIC MCU or dsPIC DSC device is best suited to the application, then design the associated hardware circuitry. After determining which peripherals and pins control the hardware, write the firmware – the software that will control the hardware aspects of the embedded application. A language tool such as an assembler, which is directly translatable into machine code, or a compiler that allows a more natural language for creating programs, should be used to write and edit code. Assemblers and compilers help make the code understandable, allowing function labels to identify code routines with variables that have names associated with their use, and with constructs that help organize the code in a maintainable structure.

FIGURE 1-4: PIC® MCU DATA SHEET – TIMING (EXCERPT)

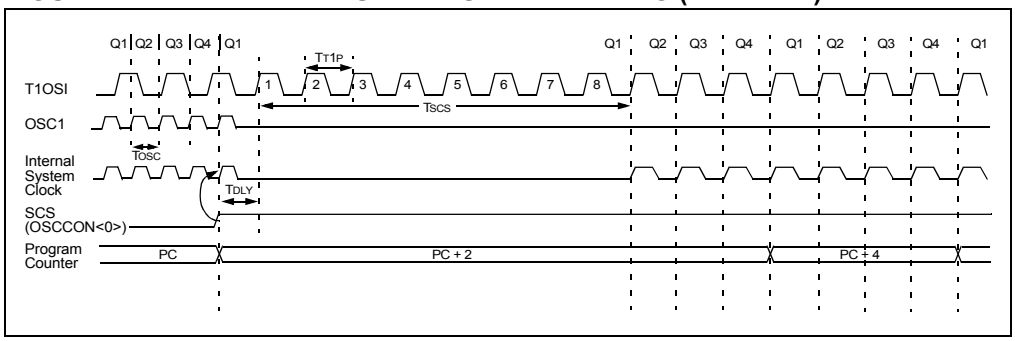
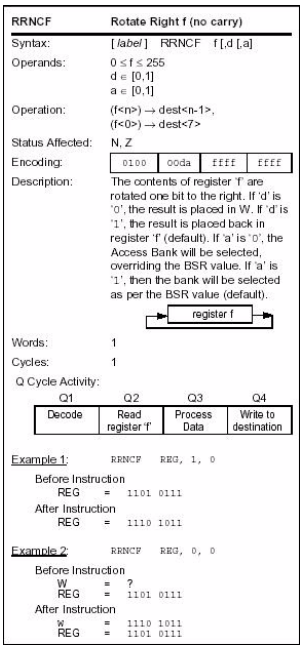


FIGURE 1-5: PIC® MCU DATA SHEET – INSTRUCTIONS (EXCERPT)



RRNCF — Rotate Right f (no carry)

Syntax: [*label*] RRNCF f [,d [,a]

Operands: $0 \le f \le 255$, $d \in [0,1]$, $a \in [0,1]$

Operation: (f<n>) → dest<n-1>, (f<0>) → dest<7>

Status Affected: N, Z

Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

2. Compile, assemble and link the software using the assembler and/or compiler and linker to convert your code into "ones and zeroes" – machine code for the PIC MCUs. This machine code will eventually become the firmware (the code programmed into the microcontroller).
3. Test your code. Usually a complex program does not work exactly the way imagined, and "bugs" need to be removed from the design to get proper results. The debugger allows you to see the "ones and zeroes" execute, related to the source code you wrote, with the symbols and function names from your program. Debugging allows you to experiment with your code to see the value of variables at various points in the program, and to do "what if" checks, changing variable values and stepping through routines.
4. "Burn" the code into a microcontroller and verify that it executes correctly in the finished application.

Of course, each of these steps can be quite complex. The important thing is to concentrate on the details of your own design, while relying upon MPLAB IDE and its components to get through each step without continuously encountering new learning curves.

Step 1 is driven by the designer, although MPLAB IDE can help in modeling circuits and code so that crucial design decisions can be made.

MPLAB IDE really helps with steps 2 through 4. Its Programmer's Editor helps write correct code with the language tools of choice. The editor is aware of the assembler and compiler programming constructs and automatically "color-keys" the source code to help ensure it is syntactically correct. The Project Manager enables you to organize the various files used in your application: source files, processor description header files and library files. When the code is built, you can control how rigorously code will be optimized for size or speed by the compiler and where individual variables and program data will be programmed into the device. You can also specify a "memory model" in order to make the best use of the microcontroller's memory for your application. If the language tools run into errors when building the application, the offending line is shown and can be "double clicked" to go to the corresponding source file for immediate editing. After editing, press the "build" button to try again. Often this write-compile-fix loop is done many times for complex code as the sub-sections are written and tested. MPLAB IDE goes through this loop with maximum speed, allowing you to get on to the next step.

Once the code builds with no errors, it needs to be tested. MPLAB IDE has components called "debuggers" and free software simulators for all PIC MCU and dsPIC DSC devices to help test the code. Even if the hardware is not yet finished, you can begin testing the code with the simulator, a software program that simulates the execution of the microcontroller. The simulator can accept a simulated input (stimulus), in order to model how the firmware responds to external signals. The simulator can measure code execution time, single step through code to watch variables and peripherals, and trace the code to generate a detailed record of how the program ran.

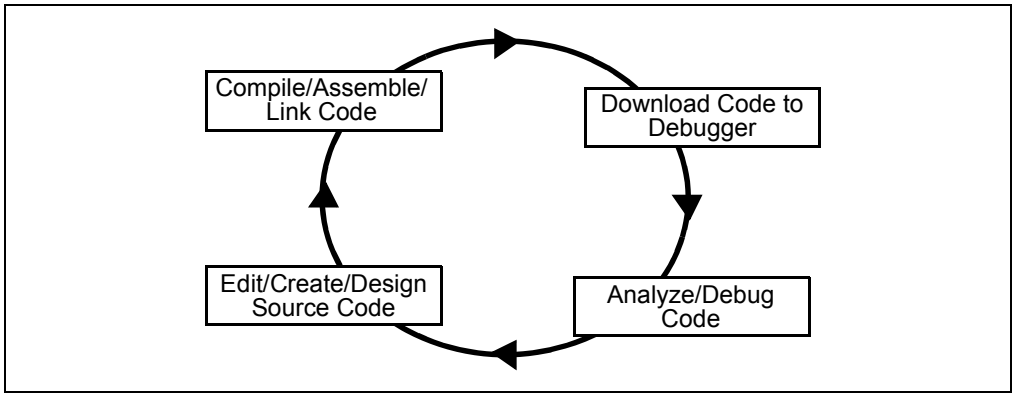
Once the hardware is in a prototype stage, a hardware debugger, such as the MPLAB REAL ICE in-circuit emulator or MPLAB ICD 2 in-circuit debugger, can be used. These debug tools run the code in real time on your actual application. The MPLAB REAL ICE emulator and MPLAB ICD 2 debugger use special circuitry built into many devices with Flash program memory and can "see into" the target microcontrollers program and data memory. These debuggers can stop and start program execution, allowing you to test the code with the microcontroller in place on the application.

After the application is running correctly, you can program a microcontroller with one of Microchip's device programmers, such as PICSTART® Plus or MPLAB PM3. These programmers verify that the finished code will run as designed. MPLAB IDE supports most PIC MCUs and all dsPIC DSCs.

1.2 THE DEVELOPMENT CYCLE

The process for writing an application is often described as a development cycle, since it is rare that all the steps from design to implementation can be done flawlessly the first time. More often code is written, tested and then modified in order to produce an application that performs correctly. The Integrated Development Environment allows the embedded systems design engineer to progress through this cycle without the distraction of switching among an array of tools. By using MPLAB IDE, all the functions are integrated, allowing the engineer to concentrate on completing the application without the interruption of separate tools and different modes of operation.

FIGURE 1-6: THE DESIGN CYCLE

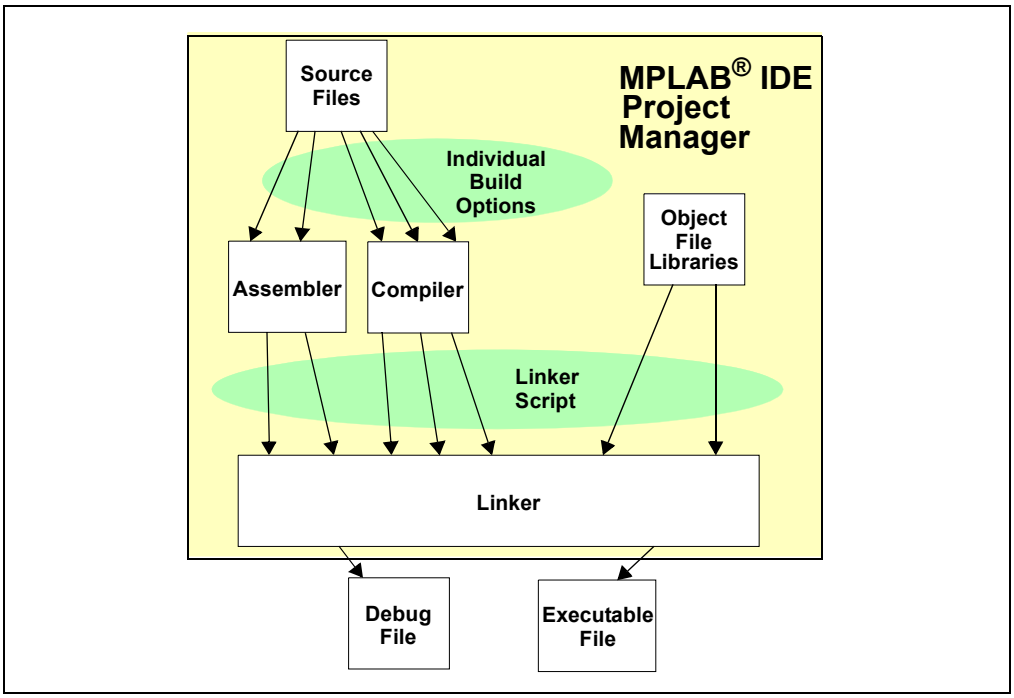


MPLAB IDE is a "wrapper" that coordinates all the tools from a single graphical user interface, usually automatically. For instance, once code is written, it can be converted to executable instructions and downloaded into a microcontroller to see how it works. In this process multiple tools are needed: an editor to write the code, a project manager to organize files and settings, a compiler or assembler to convert the source code to machine code and some sort of hardware or software that either connects to a target microcontroller or simulates the operation of a microcontroller.

1.3 PROJECT MANAGER

The project manager organizes the files to be edited and other associated files so they can be sent to the language tools for assembly or compilation, and ultimately to a linker. The linker has the task of placing the object code fragments from the assembler, compiler and libraries into the proper memory areas of the embedded controller, and ensure that the modules function with each other (or are “linked”). This entire operation from assembly and compilation through the link process is called a project “build”. From the MPLAB IDE project manager, properties of the language tools can be invoked differently for each file, if desired, and a build process integrates all of the language tools operations.

FIGURE 1-7: MPLAB® IDE PROJECT MANAGER



The source files are text files that are written conforming to the rules of the assembler or compiler. The assembler and compiler convert them into intermediate modules of machine code and placeholders for references to functions and data storage. The linker resolves these placeholders and combines all the modules into a file of executable machine code. The linker also produces a debug file which allows MPLAB IDE to relate the executing machine codes back to the source files.

A text editor is used to write the code. It is not a normal text editor, but an editor specifically designed for writing code for Microchip MCUs. It recognizes the constructs in the text and uses color coding to identify various elements, such as instruction mnemonics, C language constructs and comments. The editor supports operations commonly used in writing source code, such as finding matching braces in C, commenting and uncommenting out blocks of code, finding text in multiple files and adding special bookmarks. After the code is written, the editor works with the other tools to display code execution in the debugger. Breakpoints (which stop or “break” the execution of code) can be set in the editor, and the values of variables can be inspected by hovering the mouse pointer over the variable name. Names of variables can be dragged from source text windows and then dropped into a Watch window where their changing values can be watched after each breakpoint or during code execution.

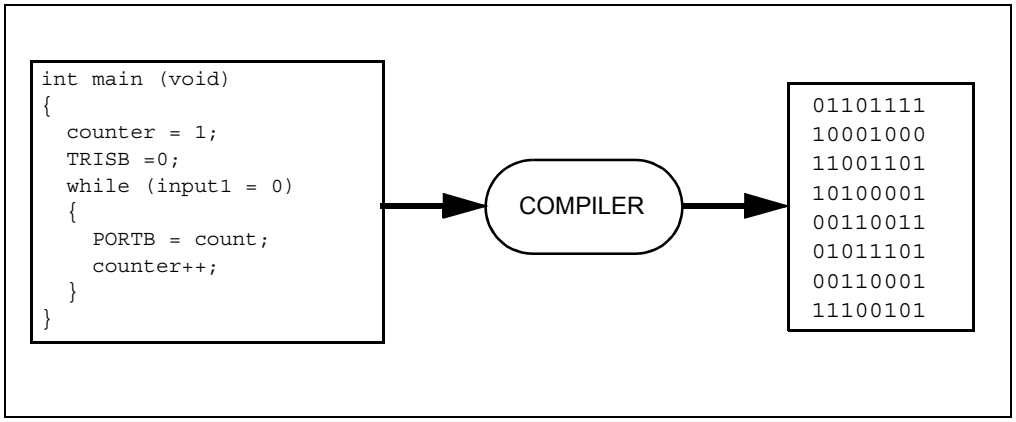
1.4 LANGUAGE TOOLS

Language tools are programs such as cross-assemblers and cross-compilers. Most people are familiar with language tools that run on a PC such as Visual Basic or C compilers. When using language tools for embedded systems, a "cross-assembler" or "cross-compiler" is used. These tools differ from typical compilers in that they run on a PC but produce code to run on another microprocessor, hence they "cross-compile" code for a microcontroller that uses an entirely different set of instructions from the PC.

The language tools also produce a debug file that MPLAB IDE uses to correlate the machine instructions and memory locations with the source code. This bit of integration allows the MPLAB IDE editor to set breakpoints, allows watch windows to view variable contents, and lets you single step through the source code, watching the application execute.

Embedded system language tools also differ somewhat for compilers that run and execute on a PC because they must be very space conscious. The smaller the code produced, the better, because that allows the smallest possible memory for the target, which reduces cost. This means that techniques to optimize and enhance the code using machine specific knowledge are desirable. The size of programs for PCs typically extends into the megabytes for moderately complex programs. The size of simple embedded systems programs may be as small as a thousand bytes or less. A medium size embedded system might need 32K or 64K of code for relatively complex functions. Some embedded systems use megabytes of storage for large tables, user text messages or data logging.

FIGURE 1-8: A COMPILER CONVERTS SOURCE CODE INTO MACHINE INSTRUCTIONS



1.5 TARGET DEBUGGING

In a development environment, the execution of the code is tested on a debugger. The debugger can be a software program that simulates the operation of the microcontroller for testing, or it can be special instrumentation to analyze the program as it executes in the application.

Simulators are built into MPLAB IDE so a program can be tested without any additional hardware. A simulator is a software debugger, and the debugger functions for the simulator are almost identical to the hardware debuggers, allowing a new tool to be learned with ease. Usually a simulator runs somewhat slower than an actual microcontroller, since the CPU in the PC is being used to simulate the operations of the microcontroller. In the case of MPLAB IDE, there are many simulators for each of the PIC MCU and the dsPIC DSC processors.

There are two types of hardware that can be used with MPLAB IDE: programmers and hardware debuggers. A programmer simply burns the machine code from the PC into the internal memory of the target microcontroller. The microcontroller can then be plugged into the application and, hopefully, it will run as designed.

Usually, however, the code does not function exactly as anticipated, and the engineer is tasked with reviewing the code and its operation in the application to determine how to modify the original source code to make it execute as desired. This process is called debugging. As noted previously, the simulator can be used to test how the code will operate, but once a microcontroller is programmed with the firmware, many things outside the scope of the simulator come into play. Using just a programmer, the code could be changed, reprogrammed into the microcontroller and plugged into the target for retest, but this could be a long, laborious cycle if the code is complex, and it is difficult to understand exactly what is going wrong in the hardware.

This is where a hardware debugger is useful. Hardware debuggers can be in-circuit emulators, which use specialized hardware in place of the actual target microcontroller, or they can be in-circuit debuggers, which use microcontrollers that have special built-in debugging features. A hardware debugger, like a simulator, allows the engineer to inspect variables at various points in the code, and single step to follow instructions as the hardware interacts with its specialized circuitry.

Debugging usually becomes urgent near the end of the project design cycle. As deadlines loom, getting the application to function as originally designed is the last step before going into deployment of the product, and often has the most influence on producing delays in getting a product out. That's where an integrated development environment is most important. Doing fine “tweaks” to the code, recompiling, downloading and testing all require time. Using all tools within a single environment will reduce the time around the “cycle.” These last steps, where critical bugs are worked out, are a test for the embedded systems designer. The right tool can save time. With MPLAB IDE many tools can be selected, but they all will have a similar interface, and the learning curve from simulator to low-cost in-circuit debugger to powerful in-circuit emulator is small.

1.6 DEVICE PROGRAMMING

After the application has been debugged and is running in the development environment, it needs to be tested on its own. A device can be programmed with the in-circuit debugger or a device programmer. MPLAB IDE can be set to the programmer function, and the part can be "burned". The target application can now be observed in its nearly final state. Engineering prototype programmers allow quick prototypes to be made and evaluated. Some applications can be programmed after the device is soldered on the target PC board. Using In-Circuit Serial Programming™ (ICSP™) programming capability, the firmware can be programmed into the application at the time of manufacture, allowing updated revisions to be programmed into an embedded application later in its life cycle. Devices that support in-circuit debugging can even be plugged back into the MPLAB ICD 2 after manufacturing for quality tests and development of next generation firmware.

1.7 COMPONENTS OF MPLAB IDE

The MPLAB IDE has both built-in components and plug-in modules to configure the system for a variety of software and hardware tools.

1.7.1 MPLAB IDE Built-In Components

The built-in components consist of:

- **Project Manager**

The project manager provides integration and communication between the IDE and the language tools.

- **Editor**

The editor is a full-featured programmer's text editor that also serves as a window into the debugger.

- **Assembler/Linker and Language Tools**

The assembler can be used stand-alone to assemble a single file, or can be used with the linker to build a project from separate source files, libraries and recompiled objects. The linker is responsible for positioning the compiled code into memory areas of the target microcontroller.

- **Debugger**

The Microchip debugger allows breakpoints, single stepping, watch windows and all the features of a modern debugger for the MPLAB IDE. It works in conjunction with the editor to reference information from the target being debugged back to the source code.

- **Execution Engines**

There are software simulators in MPLAB IDE for all PIC MCU and dsPIC DSC devices. These simulators use the PC to simulate the instructions and some peripheral functions of the PIC MCU and dsPIC DSC devices. Optional in-circuit emulators and in-circuit debuggers are also available to test code as it runs in the applications hardware.

1.7.2 Additional Tools for MPLAB IDE

These components are distributed with MPLAB IDE and may be found under the Tools menu.

- **Graph Output and Control Values Real-time**

The Data Monitor and Control Interface (DMCI) provides a mechanism to view and control variables in code and change their values real-time. It also allows you to view output data in a graphical format.

- **Work with an RTOS**

The RTOS Viewer allows easy viewing of supported real-time operating systems' parameters.

1.7.3 Additional Optional Components for MPLAB IDE

Optional components can be purchased and added to the MPLAB IDE:

- **Compiler Language Tools**

MPLAB C compilers from Microchip provide fully integrated, optimized code for PIC18, PIC24 and PIC32 MCUs and dsPIC DSCs. Along with compilers from HI-TECH, IAR, microEngineering Labs, CCS and Byte Craft, they are invoked by the MPLAB IDE project manager to compile code that is automatically loaded into the target debugger for instant testing and verification.

- **Programmers**

MPLAB PM3, PICSTART® Plus, PICkit™ 1, 2 and 3, MPLAB ICD 2 and 3 in-circuit debuggers, and MPLAB REAL ICE™ in-circuit emulator can program code into target devices. MPLAB IDE offers full control over programming both code and data, as well as the Configuration bits to set the various operating modes of the target microcontrollers or digital signal controllers.

- **In-Circuit Emulators**

The MPLAB REAL ICE and MPLAB ICE 2000 in-circuit emulator systems are for PIC MCU and dsPIC DSC devices. They connect to the PC via I/O ports and allow full control over the operation of microcontroller in the target applications.

- **In-Circuit Debugger**

MPLAB ICD 2 and 3, and PICkit 2 and 3, provide economic alternatives to an emulator. By using some of the on-chip resources, MPLAB ICD 2 and 3 can download code into a target microcontroller inserted in the application, set breakpoints, single step and monitor registers and variables.

1.8 MPLAB IDE DOCUMENTATION

The following documents are available to help you use MPLAB IDE:

- *"MPLAB® IDE User's Guide"* (DS51519)

Other documents exist for various Microchip software and hardware tools that work with MPLAB IDE. Check the Microchip web site for downloadable PDF versions of all these documents.

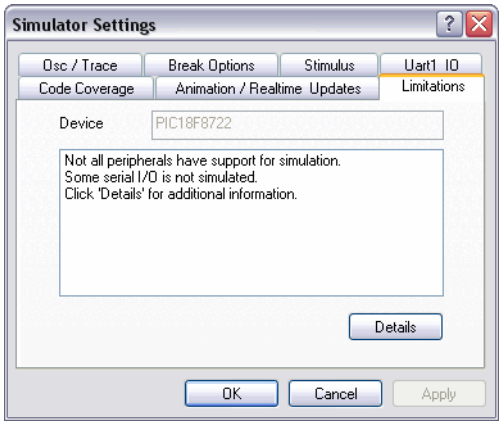
1.9 MPLAB IDE ON-LINE HELP

Since MPLAB IDE is under a constant state of change (see **Section 1.11 "MPLAB IDE Updates"**) some details in this documentation may change. Dialogs might not appear exactly as they do in this manual, menu lists may be in different order, or may have new items. For this reason, the on-line help is the best reference to the version of MPLAB IDE being used.

MPLAB IDE comes with extensive on-line help, which is constantly being updated. If questions arise while using MPLAB IDE, be sure to check the on-line help for answers. Most importantly, the on-line help lists any restrictions that might exist for a particular tool in support of a particular device. Always try to review this section before working with a new device/tool combination.

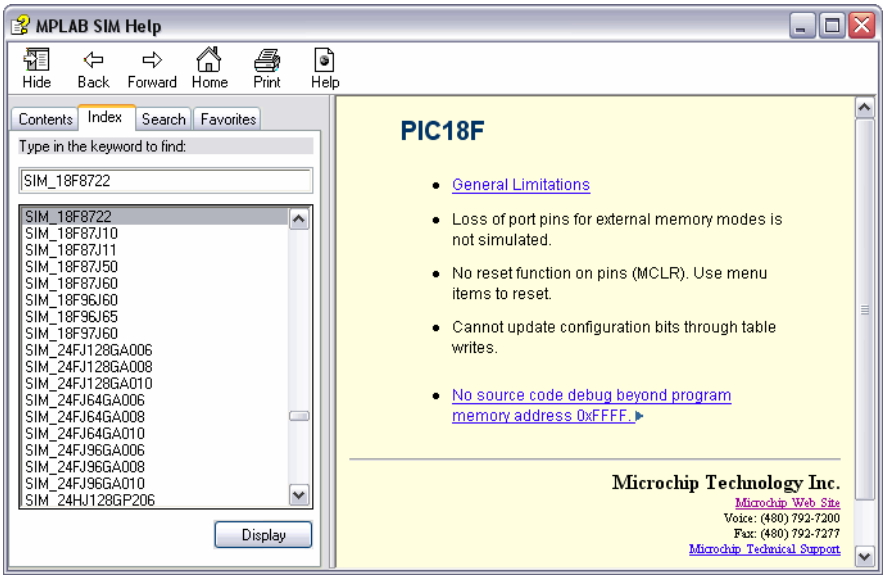
The Limitations tab in the *Debugger>Settings* dialog displays any restrictions the simulator, emulator or in-circuit debugger might have, compared to the actual device being simulated. General limitations are shown in the text area.

FIGURE 1-9: *DEBUGGER>SETTINGS*, LIMITATIONS TAB



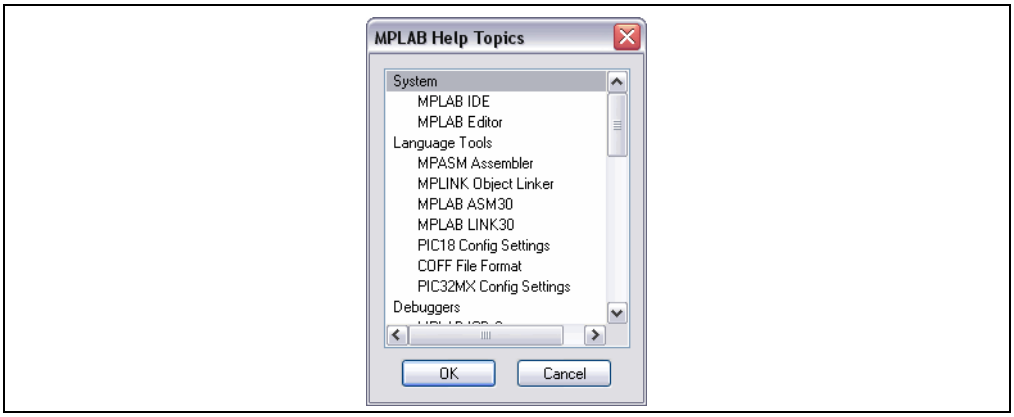
Press the **Details** button to show specific limitations of the device being debugged. From this display, help on general limitations related to the debugger can also be accessed.

FIGURE 1-10: SIMULATOR LIMITATIONS DETAIL



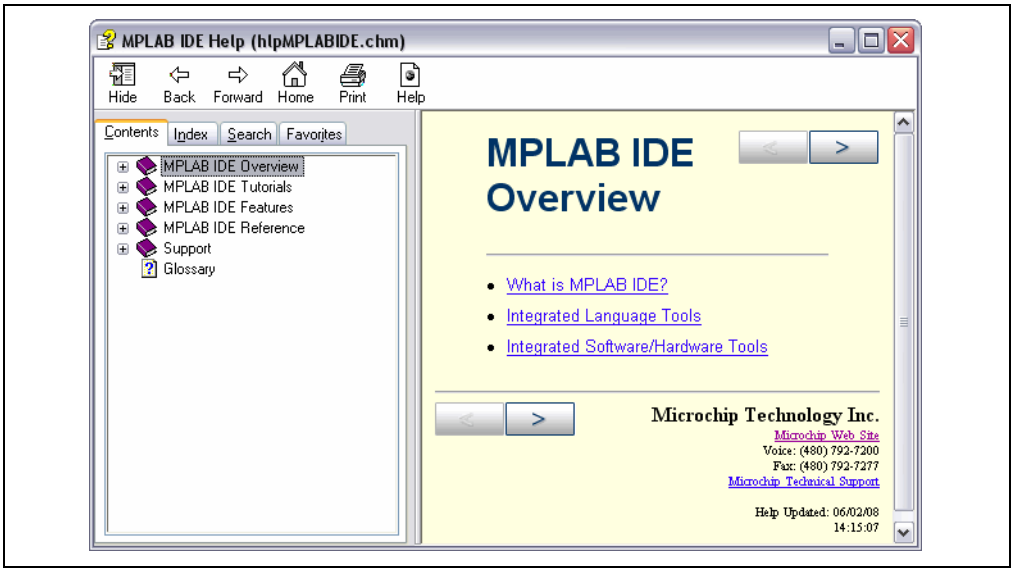
From the main MPLAB IDE Help menu, select *Help>Topics* to get a list of help on MPLAB IDE and all of its components.

FIGURE 1-11: MPLAB® IDE *HELP>TOPICS* MENU



MPLAB IDE Help covers all aspects of MPLAB IDE and all of the Microchip tools. It can be viewed in an outline, as an index and with a search utility for help on any MPLAB IDE topic. It also directs users to other types of assistance, such as the Microchip Update Notification system.

FIGURE 1-12: MPLAB® IDE HELP DIALOG



1.10 WEB SITE

Microchip provides online support via our web site at http://www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- Product Support – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- General Technical Support – Frequently Asked Questions (FAQs), technical support requests, online discussion groups/forums (http://forum.microchip.com), Microchip consultant program member listing
- Business of Microchip – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

In addition, several web seminars are available:

- Introduction to MPLAB IDE
- Introduction to Microchip's Development Tools

1.11 MPLAB IDE UPDATES

MPLAB IDE is an evolving program with thousands of users. Microchip Technology is continually designing new microcontrollers with new features. Many new MPLAB IDE features come from customer requests and from internal usage. Continued new designs and the release of new microcontrollers ensure that MPLAB IDE will continue to evolve.

MPLAB IDE is scheduled for a version update approximately every few months to add new device support and new features.

For projects that are midway through development when a new version of MPLAB IDE is released, it is considered "best practice" to not update to the new release unless there is a compelling reason to do so, such as a bug fix on a bug that inhibits the current efforts. The start of a new project is the best time to update to a new release.

Each new release of the MPLAB IDE software has new features implemented, so the printed documentation will inevitably "lag" the on-line help. The on-line help is the best source for any questions about MPLAB IDE.

To be notified of updates to MPLAB IDE and its components, subscribe to the Development Tools section of the Customer Change Notification service on http://www.microchip.com.

Chapter 2. Integrated Language Tools

2.1 INTRODUCTION

MPLAB IDE is designed to work with many Microchip and third party language tools. These tools take your application code (written in assembly, C or BASIC language) and turn it into executable code that may be programmed on your selected Microchip device.

For information on software/hardware development tools supported by MPLAB IDE, see **Chapter 3. "Integrated Software/Hardware Tools"**.

- Language Toolsuite Overview
- Microchip Language Tools
- Third Party Language Tools

2.2 LANGUAGE TOOLSUITE OVERVIEW

MPLAB IDE supports many language toolsuites. Integrated into MPLAB IDE is the Microchip MPASM Toolsuite, but many others can be used, including the Microchip C18, C30 and C32 Toolsuites, as well as language tools from HI-TECH, IAR, CCS, microEngineering Labs and Byte Craft. These are integrated into MPLAB IDE in two ways: using "plug-ins" designed by the manufacturer, and by older style ".MTC" files that can be customized for any language toolsuite.

Language Suite Plug-Ins

Vendors of language tools can write a "plug-in" that can be installed into MPLAB IDE to provide custom dialogs to access the various tool build options. This is the preferred method of support.

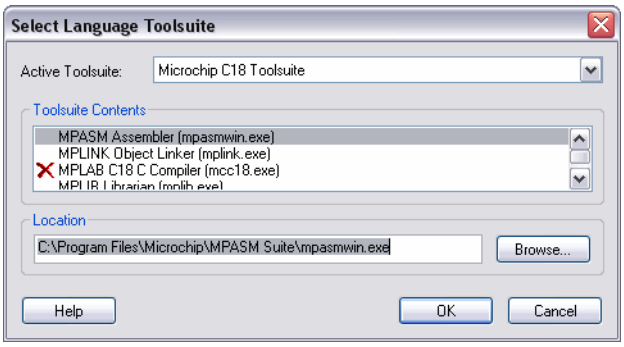
.MTC File Interface

A language tool can be integrated in a generic manner using a special file with the extension `.mtc`. These files are then integrated into a dialog to provide checkbox access to the various build options of the language tool. Using these files, most of the features of the language toolsuite can be controlled, but all may not be available.

2.2.1 Selecting the Language Toolsuite

Once a workspace is opened, a toolsuite can be selected from *Project>Select Language Toolsuite*. In some cases, MPLAB IDE will know where the various executables for the toolsuite are located, but sometimes these will need to be set manually before MPLAB IDE can use the tools.

FIGURE 2-1: SELECTING A TOOLSUITE - C18 EXAMPLE



If the selected toolsuite has red X's to the left of the toolsuite components, that means that MPLAB IDE does not know where the executables are. This can happen if the tools have been moved from their installed location. You will have to enter the full path and executable name by typing it in or browsing to the proper executable.

Also, it is possible that you do not have the tool. Select a different toolsuite or purchase the tool by visiting our website for Microchip and third-party language tools.

2.2.2 Locating Microchip Language Toolsuites

Microchip toolsuites and their locations are described below.

Microchip MPASM Toolsuite

This toolsuite includes the language tools MPASM assembler, MPLINK object linker and MPLIB object librarian. The executables for these files are `mpasmwin.exe`, `mplink.exe` and `mplib.exe` and are located in the `MPASM Suite` subdirectory of the main Microchip installation directory. For more information on these tools, see **Section 2.3.1 "Microchip 8-Bit Language Tools"**.

Microchip C18 Toolsuite

This toolsuite includes all the language tools from the Microchip MPASM Toolsuite as well as the MPLAB C Compiler for PIC18 MCUs (formerly MPLAB C18). The compiler is a separate installation from MPLAB IDE. The executable for the compiler is `mcc18.exe` and is installed in `c:\mcc18\bin` by default. For more information on these tools, see **Section 2.3.1 "Microchip 8-Bit Language Tools"**.

Microchip ASM30 Toolsuite

This toolsuite includes the language tools MPLAB Assembler for PIC24 MCUs and dsPIC DSCs (formerly MPLAB ASM30), MPLAB Object Linker for PIC24 MCUs and dsPIC DSCs (formerly MPLAB LINK30) and MPLAB Archiver/Librarian for PIC24 MCUs and dsPIC DSCs (formerly MPLAB LIB30). The executables for these files are `pic30-as.exe`, `pic30-ld.exe` and `pic30-ar.exe`, and are located in the `MPLAB ASM30 Suite\bin` subdirectory of the main MPLAB IDE installation directory. For more information on these tools, see **Section 2.3.2 "Microchip 16-Bit Language Tools"**.

Microchip C30 Toolsuite

This toolsuite includes all the language tools from the Microchip ASM30 Toolsuite as well as the MPLAB C Compiler for PIC24 MCUs and dsPIC DSCs (formerly MPLAB C30). The compiler is a separate installation from MPLAB IDE. The executable for the compiler is `pic30-gcc.exe` and is installed in the `MPLAB C30\bin` subdirectory of the main MPLAB IDE installation directory by default (unless you have previously installed to `c:\pic30\bin`, and then this the default.) For more information on these tools, see **Section 2.3.2 "Microchip 16-Bit Language Tools"**.

Note: When this toolsuite is selected, all code, including assembly code, will go through the standard C preprocessor. This means C-style directives (e.g., `#define`) are supported in assembly files.

Microchip ASM32 Toolsuite

This toolsuite includes the language tools MPLAB Assembler for PIC32MX MCUs (formerly MPLAB ASM32), MPLAB Object Linker for PIC32MX MCUs (formerly MPLAB LINK32) and MPLAB Archiver/Librarian for PIC32MX MCUs (formerly MPLAB LIB32). The executables for these files are `pic32-as.exe`, `pic32-ld.exe` and `pic32-ar.exe`, and are located in the `MPLAB ASM32 Suite\bin` subdirectory of the main MPLAB IDE installation directory. For more information on these tools, see **Section 2.3.3 "Microchip 32-Bit Language Tools"**.

Microchip C32 Toolsuite

This toolsuite includes all the language tools from the Microchip ASM32 Toolsuite as well as the MPLAB C Compiler for PIC32MX MCUs (formerly MPLAB C32). The compiler is a separate installation from MPLAB IDE. The executable for the compiler is `pic32-gcc.exe` and is installed in the `MPLAB C32\bin` subdirectory of the main MPLAB IDE installation directory by default. For more information on these tools, see **Section 2.3.3 "Microchip 32-Bit Language Tools"**.

2.3 MICROCHIP LANGUAGE TOOLS

Currently supported language tools from Microchip Technology are:

- Microchip 8-Bit Language Tools – supports most PIC MCU, KEELOQ security IC and memory devices.
- Microchip 16-Bit Language Tools – supports PIC24 MCU and dsPIC DSC devices.
- Microchip 32-Bit Language Tools – supports PIC32MX devices.

Other language tools supporting Microchip devices may be available from third parties. Please see our web site for a complete list of these suppliers.

2.3.1 Microchip 8-Bit Language Tools

The Microchip 8-bit toolsuite supports most PIC microcontroller (MCU), Microchip memory and KEELOQ security IC devices. (See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.)

TABLE 2-1: MICROCHIP TOOLSUITE SUPPORT

Tool	Old Name	New Name	Executable
C Compiler (PIC18X devices)	MPLAB® C18	MPLAB C Compiler for PIC18 MCUs	`mcc18`
Assembler	MPASM™ Assembler		`mpasmwin`
Linker	MPLINK™ Object Linker		`mplink`*
Librarian	MPLIB™ Object Librarian		`mplib`

* The executable file `_mplink.exe` is not a stand-alone program and should not be used.

Additional command-line tool executables and utilities are available. See language tool documentation for more information.

Help Files

From the MPLAB IDE Help menu, select Topics and then select one of the following under Language Tools:

- MPASM Assembler
- MPLINK Object Linker (Including MPLIB Object Librarian)
- PIC18 Configuration Bits Settings
- COFF File Format

Documentation

Please find the following C compiler documentation on the MPLAB IDE CD-ROM (DS51123) or the Microchip web site:

- "*MPLAB® C18 C Compiler Getting Started*" (DS51295)
- "*MPLAB® C18 C Compiler User's Guide*" (DS51288)
- "*MPLAB® C18 C Compiler Libraries*" (DS51297)

2.3.2 Microchip 16-Bit Language Tools

The Microchip 16-bit toolsuite supports 16-bit (data memory) devices (PIC24F/H MCUs and dsPIC30F/33F DSCs). See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

TABLE 2-2: MICROCHIP 16-BIT TOOLSUITE

Tool	Old Name	New Name	Executable
C Compiler	MPLAB® C30	MPLAB C Compiler for PIC24 MCUs and dsPIC® DSCs	`pic30-gcc`
	–	MPLAB C Compiler for dsPIC DSCs	`pic30-gcc`
	–	MPLAB C Compiler for PIC24 MCUs	`pic30-gcc`
Assembler	MPLAB ASM30	MPLAB Assembler for PIC24 MCUs and dsPIC DSCs	`pic30-as`
Linker	MPLAB LINK30	MPLAB Object Linker for PIC24 MCUs and dsPIC DSCs	`pic30-ld`
Archiver/Librarian	MPLAB LIB30	MPLAB Archiver/Librarian for PIC24 MCUs and dsPIC DSCs	`pic30-ar`

Additional command-line utilities are available. See language tool documentation for more information.

Help Files

From the MPLAB IDE Help menu, select Topics and then select under Language Tools either:

- MPLAB Assembler for PIC24 MCUs and dsPIC DSCs
- MPLAB Object Linker for PIC24 MCUs and dsPIC DSCs (includes 16-bit archiver/librarian and utilities)

Documentation

The most up-to-date 16-bit device documentation may be found on the Microchip web site: http//www.microchip.com.

Please find the following 16-bit language tools documentation on the MPLAB IDE CD-ROM (DS51123) or the Microchip web site.

- "*16-Bit Language Tools Getting Started*" (DS70094)
- "*MPLAB® Assembler, Linker and Utilities for PIC24 MCUs and dsPIC DSCs User's Guide*" (DS51317)
- "*MPLAB® C Compiler for PIC24 MCUs and dsPIC DSCs User's Guide*" (DS51284)
- "*16-Bit Language Tools Libraries*" (DS51456)

2.3.3 Microchip 32-Bit Language Tools

The Microchip 32-bit toolsuite supports 32-bit (program/data memory) devices (PIC32MX MCUs). See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

TABLE 2-3: MICROCHIP PIC32MX MCU TOOLSUITE

Tool	Old Name	New Name	Executable
C Compiler	MPLAB® C32	MPLAB C Compiler for PIC32MX MCUs	`pic32-gcc`
Assembler	MPLAB ASM32	MPLAB Assembler for PIC32MX MCUs	`pic32-as`
Linker	MPLAB LINK32	MPLAB Object Linker for PIC32MX MCUs	`pic32-ld`
Archiver/Librarian	MPLAB LIB32	MPLAB Archiver/Librarian for PIC32MX MCUs	`pic32-ar`

Additional command-line utilities are available. See language tool documentation for more information.

Documentation

The most up-to-date 32-bit device documentation may be found on the Microchip web site: http//www.microchip.com.

Please find the following 32-bit language tools documentation on the MPLAB IDE CD-ROM (DS51123) or the Microchip web site.

- "*MPLAB® C32 C Compiler User's Guide*" (DS51686)
- "*MPLAB C32 Libraries*" (DS51685)

2.4 THIRD PARTY LANGUAGE TOOLS

Currently supported third-party language tools in MPLAB IDE are:

- B Knudsen Data Language Tools
- Byte Craft Language Tools
- CCS Language Tools
- HI-TECH Language Tools
- IAR Language Tools
- microEngineering Labs Language Tools

Other language tools supporting Microchip devices may be available from third parties. Please see the Microchip web site for a complete list of these suppliers.

2.4.1 B Knudsen Data Language Tools

MPLAB IDE supports the following B Knudsen Data language tools:

Tool	Name	Executable
C Compiler for PIC12/14/16 devices	CC5X	`cc5x`
C Compiler for PIC18 devices	CC8E	`cc8e`

For more information, see the B Knudsen Data web site (http://www.bknd.com).

2.4.2 Byte Craft Language Tools

MPLAB IDE supports the following Byte Craft language tools:

Tool	Name	Executable
Assembler/C Compiler for PIC12/14/16/17 devices	MPC	`mpc`

For more information, see the Byte Craft web site (http://www.bytecraft.com).

2.4.3 CCS Language Tools

MPLAB IDE supports the following Custom Computer Services (CCS) language tools:

Tool	Name	Executable
Command-line C Compiler for PIC12/16 (12-bit) devices	PCB	`pcb`
Command-line C Compiler for PIC12/14/16 (14-bit) devices	PCM	`pcm`
Command-line C Compiler for PIC18 devices	PCH	`pch`
Windows C Compiler for PIC12/14/16 devices	PCW	`pcw`
Windows C Compiler for PIC12/14/16/18 devices (with add-on for dsPIC® DSC devices)	PCWH	`pcwh`

To set up CCS language tools for use in your project:

- Select *Project>Set Language Toolsuite*. In the Select Language Toolsuite dialog, select the CCS C Compiler for PIC12/14/16/18 devices. (If you do not see this option then the CCS MPLAB IDE 6.xx/7.xx plug-in was not installed correctly.) Click **OK** or continue by clicking **Set Tool Locations**.

  Note: Do not select the toolsuite labeled "CCS C Compiler". This is a legacy of MPLAB IDE 5.xx and will not work with later versions of MPLAB IDE.

- If you closed the previous dialog, select *Project>Set Language Tool Locations*. In the Set Language Tool Location dialog, find the toolsuite you will be using in the list and click on the first tool in this suite you will be using.

- View the executable path in the Location of Selected Tool text box. If there is no location/path listed, enter one or browse for one.
- Click **OK** until all language tool dialogs are closed.

For more information, see the CCS web site (http://www.ccsinfo.com).

2.4.4 HI-TECH Language Tools

MPLAB IDE supports the following HI-TECH language tools:

Tool	Name	Executable
C Compiler, Assembler, Linker - PIC10/12/14/16 MCUs	HI-TECH C PRO for the PIC10/12/16 MCU Family	`picc`
C Compiler, Assembler, Linker - PIC10/12/14/16/17 MCUs	PICC STD	`picc`
C Compiler, Assembler, Linker - PIC18 MCUs	HI-TECH C PRO for the PIC18 MCU Family	`picc18`
C Compiler, Assembler, Linker - PIC18 MCUs	PICC-18 STD	`picc18`
C Compiler, Assembler, Linker - PIC24 MCUs, dsPIC® DSCs	HI-TECH for dsPIC/PIC24	`dspicc`

To set up HI-TECH language tools for use in your project (for compiler versions 9.50 or above):

- Select *Project>Set Language Toolsuite*.
- Select “HI-TECH Universal Toolsuite”.

Note: This is the correct toolsuite for all Microchip devices. You do not need to adjust the Location field in this dialog, nor set the Set Language Tool Locations dialog.

For older compiler versions (less than 9.50), please go to:

http://microchip.htsoft.com/portal/mplab_hitech

For more information about HI-TECH Software products supporting PIC MCUs and dsPIC DSCs, please go to microchip.htsoft.com.

2.4.5 IAR Language Tools

MPLAB IDE supports the following IAR language tools:

Tool	Name	Executable
C Compiler for PIC16/17 devices	IAR PICmicro 16/17 C Compiler	`iccpic`
Assembler for PIC16/17 devices	IAR PICmicro 16/17 C Compiler	`apic`
C/EC++ Compiler for PIC18 devices	IAR PICmicro PIC18 C Compiler	`iccpic18`
Assembler for PIC18 devices	IAR PICmicro PIC18 C Compiler	`apic18`
C/EC++ Compiler for dsPIC DSC devices	IAR dsPIC C Compiler	`iccdspic`
Assembler for dsPIC DSC devices	IAR dsPIC C Compiler	`adspic`
Linker	IAR Linker	`xlink`

For more information, see the IAR web site (http://www.iar.com).

2.4.6 microEngineering Labs Language Tools

MPLAB IDE supports the following microEngineering Labs language tools:

Tool	Name	Executable
Basic Compiler, most PIC® MCU devices	PicBasic Compiler	pb
Basic Compiler, all PIC MCU devices	PicBasic Pro Compiler	pbp

For more information, see the microEngineering Labs web site (http://www.melabs.com).

Chapter 3. Integrated Software/Hardware Tools

3.1 INTRODUCTION

MPLAB IDE is designed to work with many Microchip and third party software/hardware development tools. For information on language tools supported by MPLAB IDE, see **Chapter 2. "Integrated Language Tools"**.

- Microchip Tools
- Third Party Tools

3.2 MICROCHIP TOOLS

The following Microchip tools are supported by MPLAB IDE:

- Editors
- Debuggers
- Programmers
- Starter Kits
- Miscellaneous Tools
- Mature Tools

3.2.1 Editors

The following **editor** is supported:

Name	Devices/Languages Supported	*Help>Topics*
MPLAB® Editor	See Editor right click menu, Text Mode, for support	System

3.2.2 Debuggers

There are three types of debuggers supported: simulators (software solution only), in-circuit debuggers (combined hardware/software solution), and in-circuit emulators (advanced combined hardware/software solution).

The following **simulator** is supported for debug:

Name	Devices Supported*	*Help>Topics*
MPLAB® SIM Simulator	PIC® MCUs, dsPIC® DSCs	Debuggers

The following **in-circuit debuggers** are supported for debug:

Name	Devices Supported*	*Help>Topics*
MPLAB® ICD 3	PIC® Flash MCUs, dsPIC® Flash DSCs	Debuggers
MPLAB ICD 2	PIC Flash MCUs, dsPIC Flash DSCs	Debuggers
PICkit™ 3 Debug Express	PIC Flash MCUs, dsPIC Flash DSCs	Programmers
PICkit 2 Debug Express	PIC Flash MCUs, dsPIC Flash DSCs	Programmers

The following **in-circuit emulators** are supported for debug:

Name	Devices Supported*	*Help>Topics*
MPLAB® REAL ICE™ in-circuit emulator	PIC® Flash MCUs, dsPIC® Flash DSCs	Debuggers
MPLAB ICE 2000 in-circuit emulator	PIC MCUs including some PIC18 MCUs	Debuggers

* See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

3.2.3 Programmers

There are two types of programmers supported: dedicated (programming function only) and multifunction (programming in just one function of the tool.)

The following programmers are supported:

Name	Devices Supported*	*Help>Topics*
Dedicated Programmers		
MPLAB® PM3 Device Programmer**	PIC® MCUs, dsPIC® DSCs, Memory devices, KEELOQ® security ICs***	Programmers
PICSTART® Plus Development Programmer	Most PIC MCUs	Programmers
AN851 Quick Programmer	PIC16F877A, PIC18F452	Programmers
Multifunction Programmers		
MPLAB REAL ICE™ In-Circuit Emulator	PIC Flash MCUs, dsPIC Flash DSCs	Debuggers
MPLAB ICD 3	PIC Flash MCUs, dsPIC Flash DSCs	Debuggers
MPLAB ICD 2	PIC Flash MCUs, dsPIC Flash DSCs	Debuggers
PICkit™ 3 Development Programmer	PIC Flash MCUs, dsPIC Flash DSCs	Programmers
PICkit 2 Development Programmer	PIC Flash MCUs, dsPIC Flash DSCs	Programmers

* See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

** Command-line versions of this tool is available as `pm3cmd.exe`. A visual version of the command-line tool is available as Visual Procmd (`vprocmd.exe`). Find all this in the MPLAB IDE installation directory under "Programmer Utilities".

*** KEELOQ security ICs support provided by the KEELOQ Plugin.

3.2.4 Starter Kits

Starter kits provide programming and debugging support, but, in most cases, only for the built-in device on the starter kit board.

The following starter kits are supported:

Name	Device(s) Supported	User's Guide
MPLAB® Starter Kit for dsPIC® DSCs	dsPIC33FJ256GP506	DS51700
MPLAB Starter Kit for PIC24F MCUs	PIC24FJ256GB110	DS51725
PIC32MX Starter Kit	PIC32MX360F512L	DS61144
MPLAB Starter Kit for Serial Memory Products	SEEXXX	DS22087

More starter kits are planned. Check the Microchip website for details.

3.2.5 Miscellaneous Tools

The following miscellaneous tools are supported:

Name	Devices Supported*	*Help>Topics*
Data Monitor and Control Interface	N/A	Tools
KEELOQ Plugin	KEELOQ® security ICs	Programmers
AN901 – Sensorless, Brushless DC (BLDC) Motor Tuning Interface	dsPIC30F DSCs	Tools
AN908 – AC Induction Motor (ACIM) Tuning Interface	dsPIC30F DSCs	Tools
MPLAB IDE Macros	N/A	System, MPLAB IDE**

* See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

** See **Chapter 10. "MPLAB Macros"**.

3.2.6 Mature Tools

The following tools are at end-of-life. Support still exists, but upgrading to the replacement tool is recommended.

Tool (Replacement)	Devices Supported*	*Help>Topics*
MPLAB® ICE 4000 in-circuit emulator **(MPLAB REAL ICE in-circuit emulator)**	PIC18 MCUs, dsPIC® DSCs	Debuggers
PRO MATE® II Device Programmer** **(MPLAB PM3 Device Programmer)**	PIC® MCUs, Memory devices, KEELOQ® security ICs***	Programmers
PICkit™ 1 Development Programmer **(PICkit 2 or 3 Development Programmer)**	Selected PIC12/16 MCUs	"*PICkit™ 1 Flash Starter Kit User's Guide*" (DS40051)

* See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

** Command-line versions of this tool is available as `procmd.exe`. A visual version of the command-line tool is available as Visual Procmd (`vprocmd.exe`). Find all this in the MPLAB IDE installation directory under "Programmer Utilities".

*** KEELOQ security ICs support provided by the KEELOQ Plugin.

3.3 THIRD PARTY TOOLS

Other development tools supporting Microchip devices may be available from third parties. Please see the Microchip web site (www.microchip.com) for a complete list of these suppliers.

The following MPLAB IDE support is available for third party tools:

Name	Devices Supported	*Help>Topics*
MATLAB/Simulink	All that support C code	Tools
Gimpel PC-Lint/MISRA	All that support C code	Tools
RTOS Viewer	See RTOS documentation	System, MPLAB IDE*

* See **Section 13.21 "RTOS Viewer Window"**.

NOTES:

Part 2 – MPLAB IDE Tutorials

NOTES:

Chapter 4. A Basic Tutorial for MPLAB IDE

4.1 INTRODUCTION

MPLAB Integrated Development Environment (IDE) is a comprehensive editor, project manager and design desktop for application development of embedded designs using Microchip PIC MCUs and dsPIC DSCs.

The initial use of MPLAB IDE is covered here. How to make projects, edit code and test an application will be the subject of a short tutorial. By going through the tutorial, the basic concepts of the Project Manager, Editor and Debugger can be quickly learned. The complete feature set of MPLAB IDE is covered in later chapters.

This section details the installation and uninstall of MPLAB IDE. It is followed by a simple step-by-step tutorial that creates a project and explains the elementary debug capabilities of MPLAB IDE. Someone unfamiliar with MPLAB IDE will get a basic understanding of using the system to develop an application. No previous knowledge is assumed, and comprehensive technical details of MPLAB IDE and its components are omitted in order to present the basic framework for using MPLAB IDE.

These basic steps will be covered in the tutorial:

- MPLAB IDE Features and Installation
- Tutorial Overview
- Selecting the Device
- Creating the Project
- Setting Up Language Tools
- Naming the Project
- Adding Files to the Project
- Building the Project
- Creating Code
- Building the Project Again
- Testing Code with the Simulator
- Tutorial Summary

4.2 MPLAB IDE FEATURES AND INSTALLATION

MPLAB IDE is a Windows® Operating System (OS) based Integrated Development Environment for the PIC MCU families and the dsPIC Digital Signal Controllers. The MPLAB IDE provides the ability to:

- Create and edit source code using the built-in editor.
- Assemble, compile and link source code.
- Debug the executable logic by watching program flow with the built-in simulator or in real time with in-circuit emulators or in-circuit debuggers.
- Make timing measurements with the simulator or emulator.
- View variables in Watch windows.
- Program firmware into devices with device programmers (for details, consult the user's guide for the specific device programmer).

Note:	Selected third party tools are also supported by MPLAB IDE. Check the release notes or readme files for details.

4.2.1 Install/Uninstall MPLAB IDE

To install MPLAB IDE on your system:

Note:	For some Windows OSs, administrative access is required in order to install software on a PC.

- If installing from a CD-ROM, place the disk into a CD drive. Follow the on-screen menu to install MPLAB IDE. If no on-screen menu appears, use Windows Explorer to find and execute the CD-ROM menu, `menu.exe`.
- If downloading MPLAB IDE from the Microchip web site (www.microchip.com), locate the download (`.zip`) file, select the file and save it to the PC. Unzip the file and execute the resulting `setup.exe` file to install.

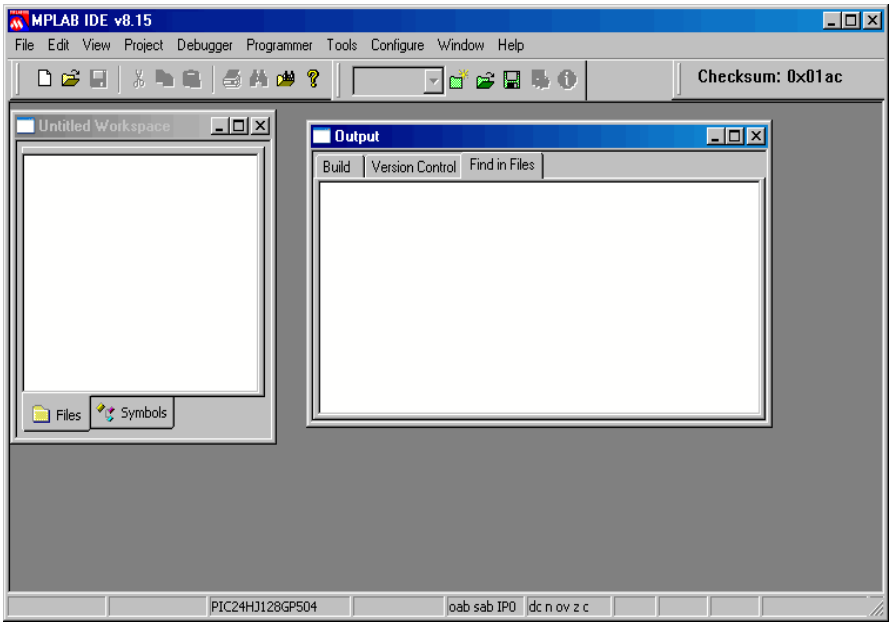
To uninstall MPLAB IDE:

- Select *Start>Settings>Control Panel* to open the Control Panel.
- Double click on Add/Remove Programs. Find MPLAB IDE on the list and click on it.
- Click **Change/Remove** to remove the program from your system.

4.2.2 Running MPLAB IDE

To start MPLAB IDE, double click on the icon installed on the desktop after installation or select *Start>Programs>Microchip>MPLAB IDE vx.xx>MPLAB IDE*. A screen will display the MPLAB IDE logo followed by the MPLAB IDE desktop (Figure 4-1).

FIGURE 4-1: MPLAB® IDE DESKTOP



4.3 TUTORIAL OVERVIEW

In order to create code that is executable by the target PIC MCU, source files need to be put into a project. The code can then be built into executable code using selected language tools (assemblers, compilers, linkers, etc.). In MPLAB IDE, the project manager controls this process.

All projects will have these basic steps:

1. Select Device

   The capabilities of MPLAB IDE vary according to which device is selected. Device selection should be completed before starting a project.

2. Create Project

   MPLAB IDE Project Wizard will be used to Create a Project.

3. Select Language Tools

   In the Project Wizard the language tools will be selected. For this tutorial, the built-in assembler and linker will be used. For other projects, one of the Microchip compilers or other third party tools might be selected.

4. Put Files in Project

   Two files will be put into the project, a template file and a linker script. Both of these files exist in sub-folders within the MPLAB IDE folder. It is easy to get started using these two files.

5. Create Code

   Some code will be added to the template file to send an incrementing value out an I/O port.

6. Build Project

   The project will be built – causing the source files to be assembled and linked into machine code that can run on the selected PIC MCU.

7. Test Code with Simulator

   Finally, the code will be tested with the simulator.

The Project Wizard will easily guide us through most of these steps.

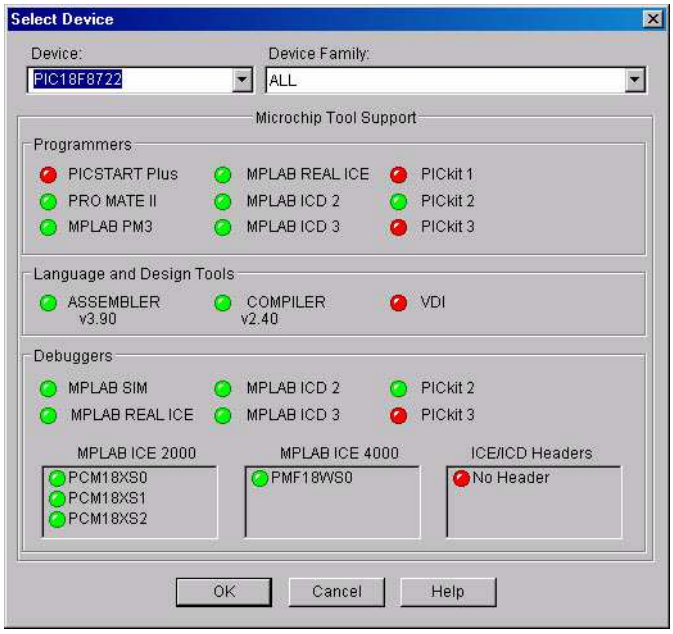
Note: Some aspects of the user interface will change in future product releases and the screen shots in this tutorial may not exactly match the appearance of the MPLAB IDE desktop in later releases. New features will be added as additional parts are released. None of the functions described in this tutorial will be removed, but more features may be added.The on-line help is the most up-to-date reference for the current version of MPLAB IDE.

4.4 SELECTING THE DEVICE

To show menu selections in this document, the menu item from the top row in MPLAB IDE will be shown after the menu name like this *MenuName>MenuItem*. To choose the *Select Device* entry in the *Configure* menu, it would be written as *Configure>Select Device*.

Choose *Configure>Select Device*. In the Device dialog, select the **PIC18F8722** from the list if it's not already selected.

FIGURE 4-2: SELECT DEVICE DIALOG



The "lights" indicate which MPLAB IDE components support this device.

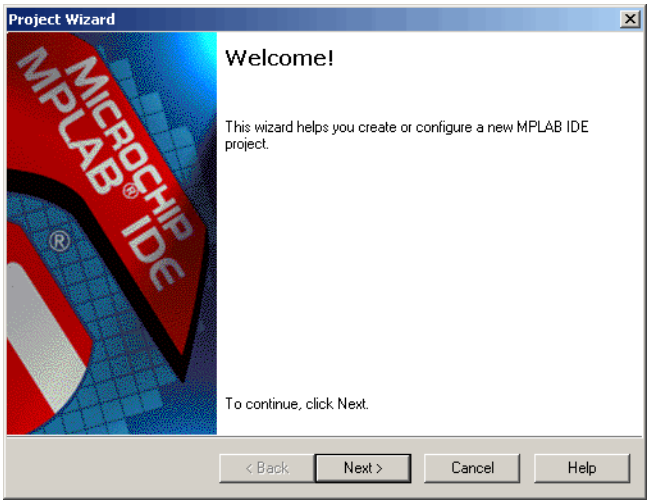
- A green light indicates full support.
- A yellow light indicates preliminary support for an upcoming part by the particular MPLAB IDE tool component. Components with a yellow light instead of a green light are often intended for early adopters of new parts who need quick support and understand that some operations or functions may not be available or operate as expected.
- A red light indicates no support for this device. Support may be forthcoming or inappropriate for the tool, e.g., dsPIC DSC devices cannot be supported on MPLAB ICE 2000.

4.5 CREATING THE PROJECT

The next step is to create a project using the Project Wizard. A project is the way the files are organized to be compiled and assembled. We will use a single assembly file for this project and a linker script. Choose *Project>Project Wizard*.

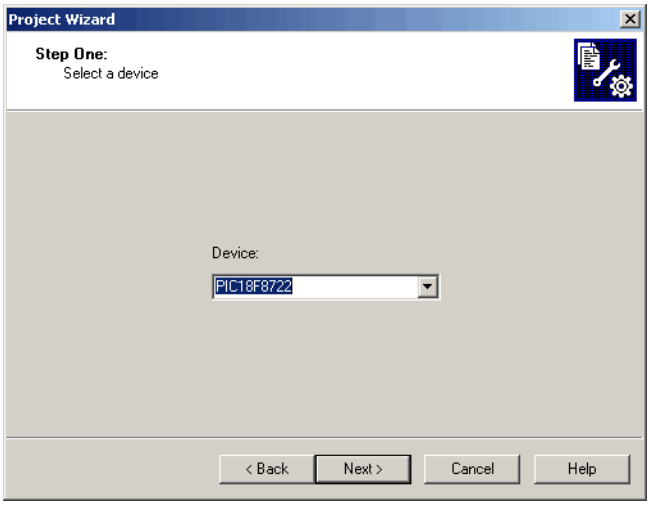
From the Welcome dialog, click on **Next>** to advance.

FIGURE 4-3: PROJECT WIZARD WELCOME



The next dialog (Step One) allows you to select the device, which we've already done. Make sure that it says PIC18F8722. If it does not, select the PIC18F8722 from the drop down list. Click **Next>**.

FIGURE 4-4: PROJECT WIZARD – SELECT DEVICE



4.6 SETTING UP LANGUAGE TOOLS

Step Two of the Project Wizard sets up the language tools that are used with this project. Select “Microchip MPASM Toolsuite” in the Active Toolsuite list box. Then “MPASM” and “MPLINK” should be visible in the Toolsuite Contents box. Click on each one to see its location. If MPLAB IDE was installed into the default directory, the MPASM assembler executable will be:

```
C:\Program Files\Microchip\MPASM Suite\mpasmwin.exe
```

the MPLINK linker executable will be:

```
C:\Program Files\Microchip\MPASM Suite\mplink.exe
```

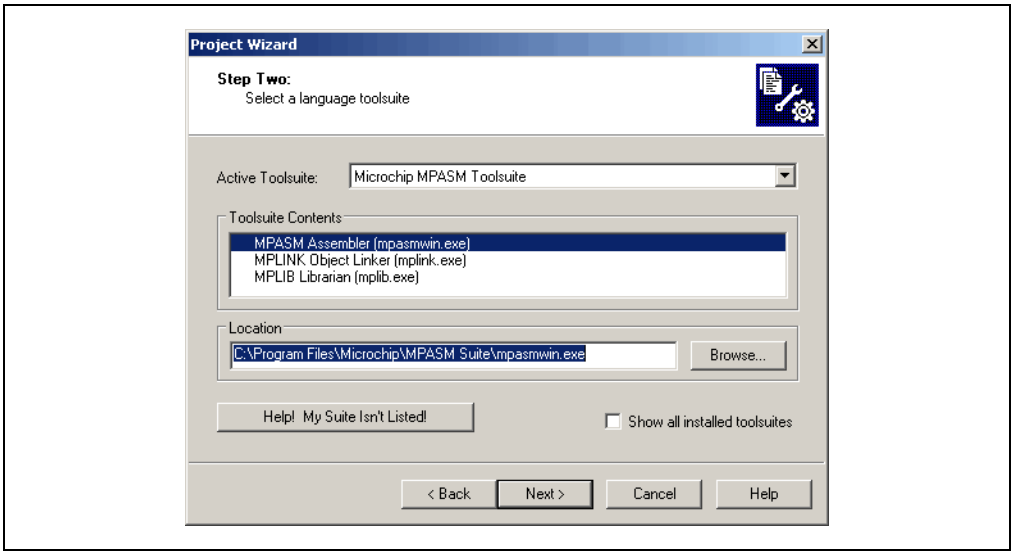
> **Note:** The executable file `_mplink.exe` is not a stand-alone program and should not be used with MPLAB IDE.

and the MPLIB librarian executable will be:

```
C:\Program Files\Microchip\MPASM Suite\mplib.exe
```

If these do not show up correctly, use the browse button to set them to the proper files in the MPLAB IDE subfolders.

FIGURE 4-5: PROJECT WIZARD – SELECT LANGUAGE TOOLS

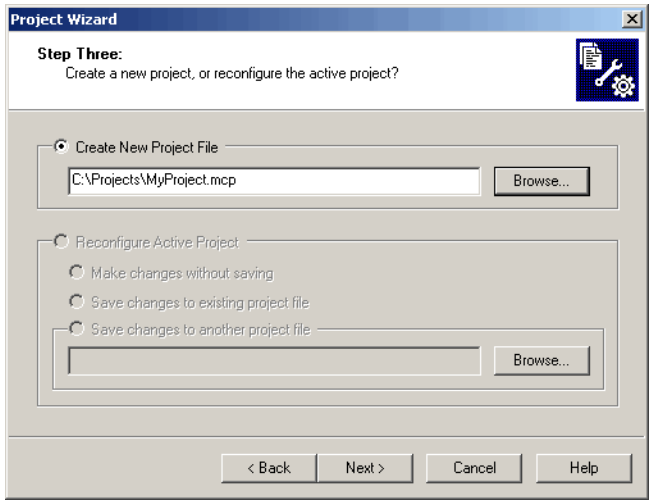


When you are finished, click **Next>**.

4.7 NAMING THE PROJECT

Step Three of the wizard allows you to name the new project and put it into a folder. This sample project will be called `C:\Projects\MyProject`. Type this into the text box and then click **Next>**. You will be prompted to create the directory since it does not exist. Click **OK**.

FIGURE 4-6: PROJECT WIZARD – NAME PROJECT



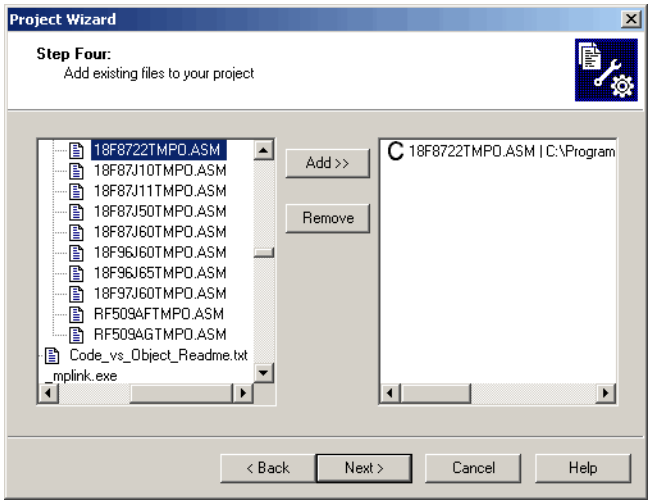
4.8 ADDING FILES TO THE PROJECT

Step Four of the Project Wizard allows file selection for the project. A source file has not yet been selected, so we will use an MPLAB IDE template file. The template files are simple files that can be used to start a project. They have the essential sections for any source file, and contain information that will help you write and organize your code.

There are two template files for each Microchip PIC MCU and dsPIC DSC device: one for absolute code (no linker used) in the `Code` directory and one for relocatable code (linker used) in the `Object` directory. Since we will be using the linker in this tutorial, choose the file named `18F8722.` in the `Object` directory. If MPLAB IDE is installed in the default location, the full path to the file will be:

```
C:\Program Files\Microchip\MPASM Suite\Template\Object
  \18F8722TMPO.ASM.
```

FIGURE 4-7: PROJECT WIZARD – SELECT TEMPLATE FILE



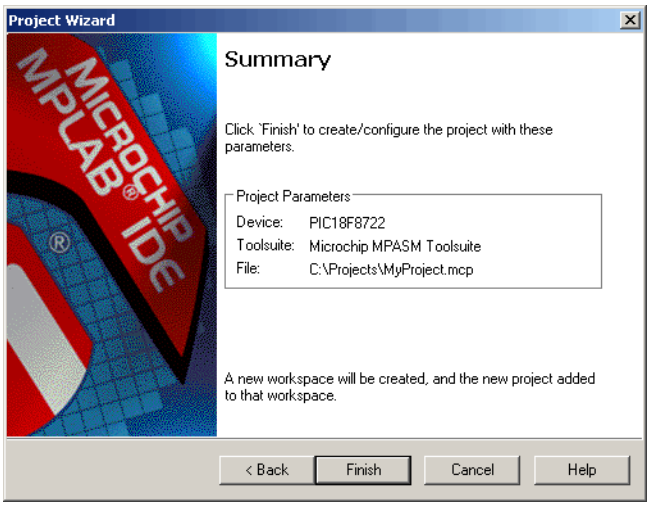
Press **Add>>** to move the file name to the right panel.Click on the “A“ at the start of the line with the file name three times until a “C” appears. This will enable this file to be copied to our project directory. (For more on the meaning of these letters, see **Section 6.2.5 “Project Wizard – Add Files”**.)

The code used in the previous file was relocatable, and therefore needs a linker script to create an executable output. For most projects, as with this one, it is no longer necessary to add a linker script to the project; the linker will automatically use the correct one. See **Section 7.3 “Linker Script Usage”** for more information.

Make sure that your dialog looks like the picture above, and then press **Next>** to finish the Project Wizard.

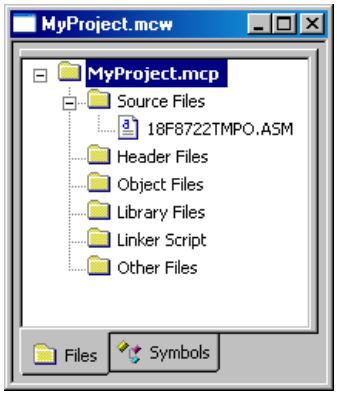
The final screen of the Project Wizard is a summary showing the selected device, the toolsuite and the new project file name.

FIGURE 4-8: PROJECT WIZARD – SUMMARY



After pressing the **Finish** button, review the Project Window on the MPLAB IDE desktop. It should look like Figure 4-9. If the Project Window is not open, select *View>Project*.

FIGURE 4-9: PROJECT WINDOW



TIP: Files can be added and projects saved by clicking the right mouse button in the project window. In case of error, files can be manually removed from the project by selecting them, clicking the right mouse button and selecting "Remove" from the menu.

4.9 BUILDING THE PROJECT

From the Project menu, we can assemble and link the current files. They don't have any of our code in them yet, but this ensures that the project is set up correctly.

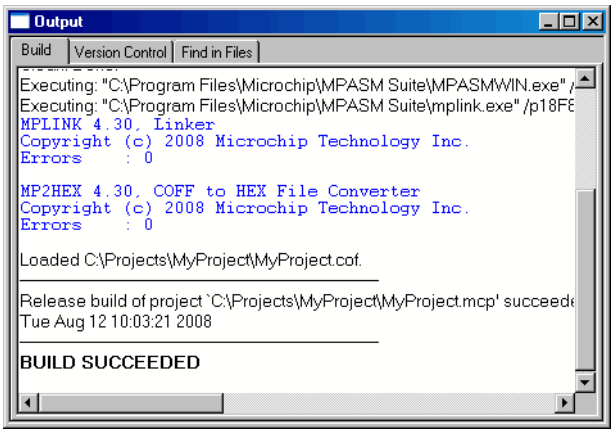
To build the project, select either:

- *Project>Build All*
- Right click on the project name in the project window and select Build All
- Click the Build All icon on the Project toolbar. Hover the mouse over icons to see pop-up text of what they represent.

The Output window shows the result of the build process. There should be no errors or warnings on any step. However, if you do receive errors, go back to the previous sections and check the project assembly steps. Errors will prevent the project from building. If you receive warnings, you may ignore them for this project as they will not prevent the project from building. To turn off the display of warnings, do the following:

- Select *Project>Build Options>Project* and click on the **MPASM Assembler** tab.
- Select "Output" from the "Categories" drop-down list.
- Select "Errors only" from the "Diagnostic level" drop-down list.
- Click **OK**.

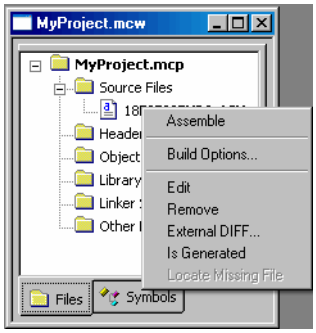
FIGURE 4-10: OUTPUT WINDOW



4.10 CREATING CODE

Open the template file in the project by double clicking on its name in the Project Window, or by selecting it with the cursor and using the right mouse button to bring up the context menu:

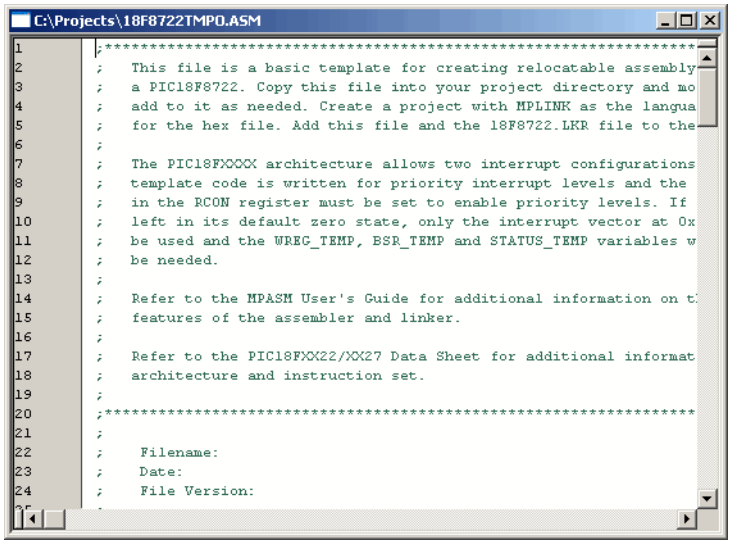
FIGURE 4-11: PROJECT CONTEXT MENU (RIGHT MOUSE CLICK)



The file has some comments at the beginning, and this area can be used as a standard comment information header for the file. For now you'll leave this as it is, but if this were a real project, you could put information about your design here.

Note: Line numbers are shown here. Line numbers may be toggled on/off by right clicking in the editor window, selecting Properties, and then checking/unchecking "Line Numbers" on the **'ASM' File Type** tab of the Editor Options dialog.

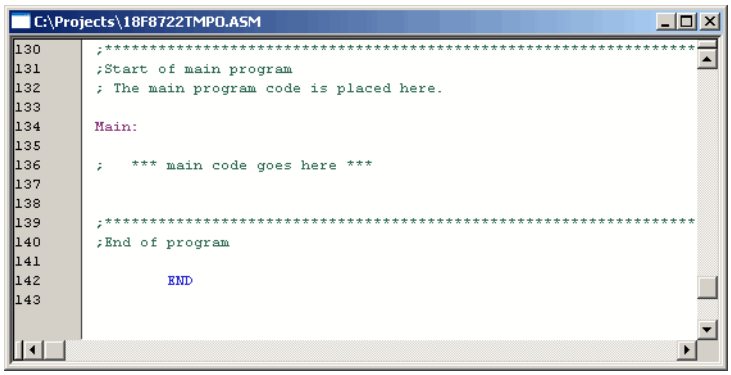
FIGURE 4-12: TEMPLATE FILE



The code in the first part of the file is for more advanced functions such as setting up interrupts and Configuration bits in a final application. These details can be ignored at this point with focus on writing the code. The new code will be placed in the file at the point after the symbol `Main` is defined.

Scroll down to the bottom of the file.

FIGURE 4-13: TEMPLATE FILE – MAIN



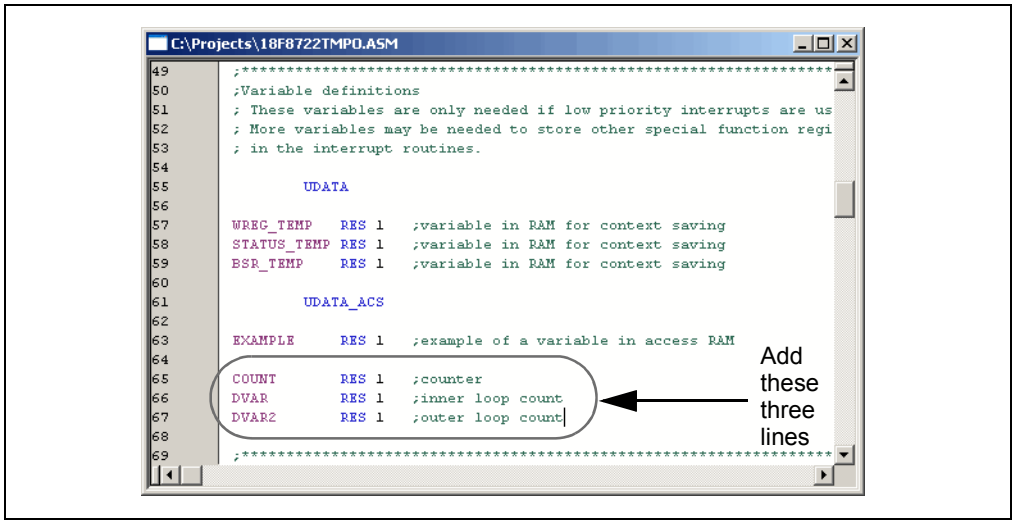
When any source file is opened, you are automatically in the editor. Type in this code beneath `Main:`

```
   clrf     WREG
   movwf    PORTC      ; clear PORTC
   movwf    TRISC      ; configure PORTC as all outputs

Init
   clrf     COUNT,A    ; initialize counter
IncCount
   incf     COUNT,F,A
   movf     COUNT,W,A  ; increase count and
   movwf    PORTC      ; display on PORTC

   call     Delay      ; go to Delay subroutine
   goto     IncCount   ; infinite loop

Delay
   movlw    0x40
   movwf    DVAR2,A    ; set outer delay loop
DelayOuter
   movlw    0xFF
   movwf    DVAR,A     ; set inner delay loop
DelayInner
   decfsz   DVAR,F,A
   goto     DelayInner

   decfsz   DVAR2,F,A
   goto     DelayOuter
   return
```

In this bit of code, we used three variables named `COUNT`, `DVAR` and `DVAR2`. These variables need to be defined in the template file in the `UDATA_ACS` section for uninitialized data using Access RAM. Using the Access bank will make the code simpler, as there will be no need to keep track of banks (`banksel`) for each variable.

There is already one variable in this section of the template file, and ours can be added at the end using the same format. Each variable is 8-bit, so only 1 byte each needs to be reserved.

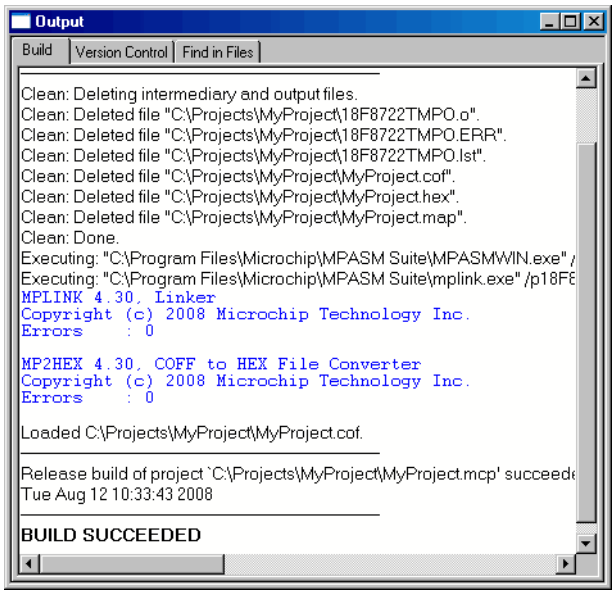
FIGURE 4-14: TEMPLATE FILE – ADD VARIABLES



4.11 BUILDING THE PROJECT AGAIN

Select *Project>Build All* to assemble and link the code. If the code assembled with no errors, the Output Window will look like Figure 4-15.

FIGURE 4-15: BUILD OUTPUT WINDOW



If the code did not assemble and link successfully, check the following items and then build the project again:

- If the assembler reported errors in the Output window, double click on the error and MPLAB IDE will open the corresponding line in the source code with a small blue pointer in the left margin of the source code window.
- Check the spelling and format of the code entered in the editor window. Make sure the new variables and the special function registers, TRISC and PORTC, are in upper case.
- Check that the correct assembler (MPASM assembler) and linker for PIC MCU devices is being used. Select *Project>Set Language Tool Locations*. Click on the plus boxes to expand the Microchip MPASM toolsuite and its executables. Click MPASM Assembler (`mpasmwin.exe`) and review their location in the display. If the location is correct, click **Cancel**. If it is not, change it and then click **OK**. The default search paths can be empty.

Upon a successful build, the output file generated by the language tool will be loaded. This file contains the object code that can be programmed into a PIC MCU and debugging information so that source code can be debugged and source variables can be viewed symbolically in Watch windows.

Note: The real power of projects is evident when there are many files to be compiled/assembled and linked to form the final executable application – as in a real application. Projects keep track of all of this. Build options can be set for each file that access other features of the language tools, such as report outputs and compiler optimizations.

4.12 TESTING CODE WITH THE SIMULATOR

In order to test the code, hardware or software is needed that will execute the PIC MCU instructions. A debug execution tool is a hardware or software tool that is used to inspect code as it executes a program (in this case `18F8722TMPO.ASM`). Hardware tools, such as the MPLAB REAL ICE in-circuit emulator, can execute code in real devices. If hardware is not available, the MPLAB SIM simulator can be used to test the code. For this tutorial use MPLAB SIM simulator.

The simulator is a software program that runs on the PC to *simulate* the instructions of the PIC MCU. It does not run in “real time,” since the simulator program is dependent upon the speed of the PC, the complexity of the code, overhead from the operating system and how many other tasks are running. However, the simulator accurately *measures* the time it would take to execute the code if it were operating in real time in an application.

Note: Hardware tools are used to test code on the application PC board. Most of the MPLAB IDE debugging operations are the same as the simulator but, unlike the simulator, these tools allow the target PIC MCU to run at full speed in the actual target application.

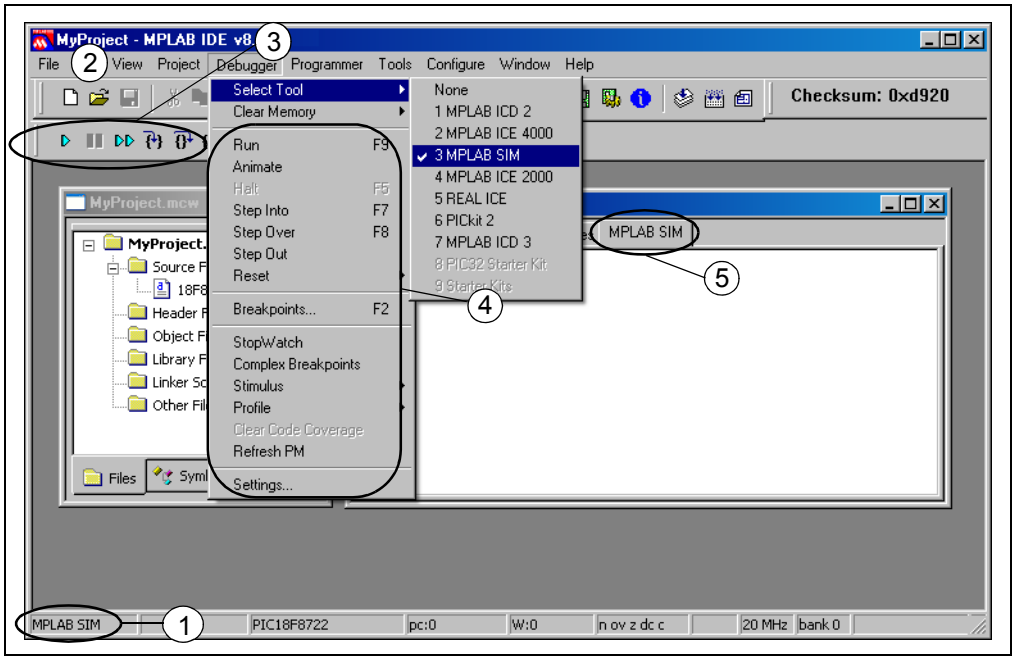
Select the simulator as the debug execution tool. This is done from the *Debugger>Select Tool* pull down menu. After selecting MPLAB SIM, the following changes should be seen (see corresponding numbers in Figure 4-16).

1. The status bar on the bottom of the MPLAB IDE window should change to "MPLAB SIM".
2. Under the View menu (not expanded), additional simulator windows.
3. Additional toolbar icons should appear in the Debug Tool Bar.

TIP: Position the mouse cursor over a toolbar button to see a brief description of the button's function.

4. Additional menu items should now appear in the Debugger menu.
5. An MPLAB SIM tab is added to the Output window.

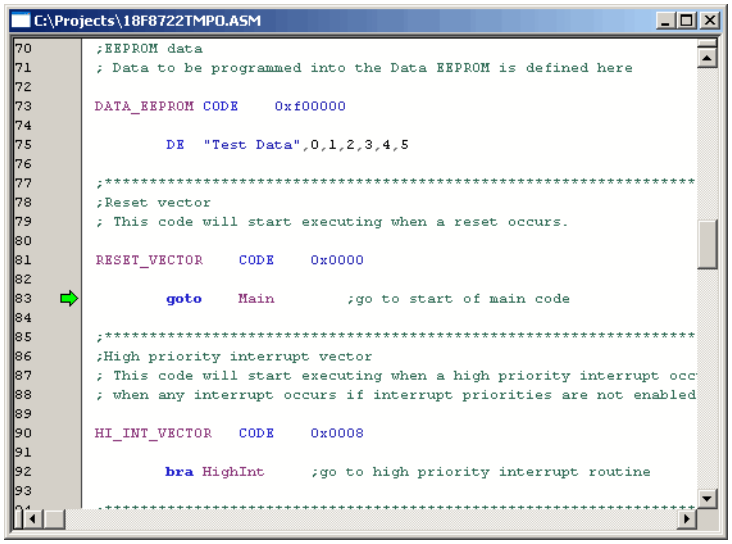
FIGURE 4-16: MPLAB® IDE DESKTOP WITH MPLAB SIM AS DEBUGGER



Now that your project is set up and the debug tool is selected, you should save your workspace setup. Select *File>Save Workspace*.

Next, select *Debugger>Reset>Processor Reset* and a green arrow shows where the program will begin. This was part of the template file. The first instruction in memory jumps to the label called `Main`, where your code was inserted, i.e., it jumps over the vector areas (reset vector, interrupt vectors, etc.) to the user memory space in program memory.

FIGURE 4-17: CODE AFTER PROCESSOR RESET



To single step through the application program, select *Debugger>Step Into*. This will execute the currently indicated line of code and move the arrow to the next line of code to be executed.

There are shortcuts for these commonly used functions in the Debug Tool Bar.

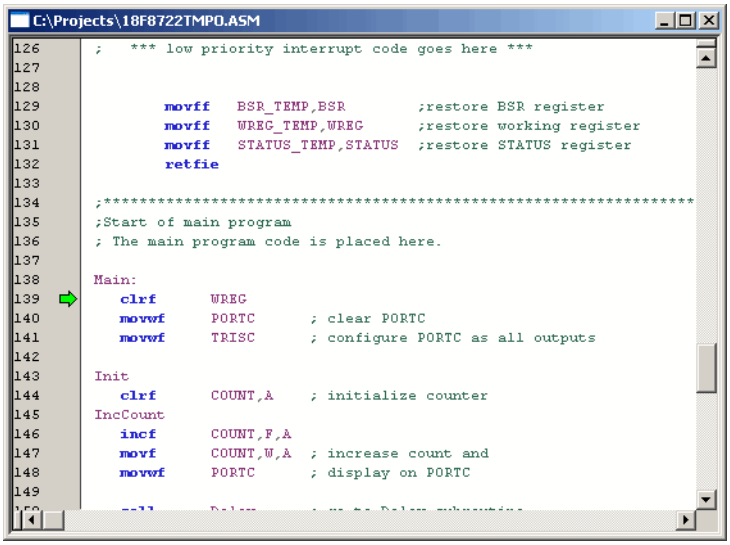
TABLE 4-1: DEBUG SHORT CUT ICONS

Debugger Menu	Toolbar Buttons	Hot Key
Run		F9
Halt		F5
Animate		
Step Into		F7
Step Over		F8
Step Out		
Reset		F6

TIP: Click on the appropriate icon on the toolbar or use the hot key shown next to the menu item. This is usually the best method for repeated stepping.

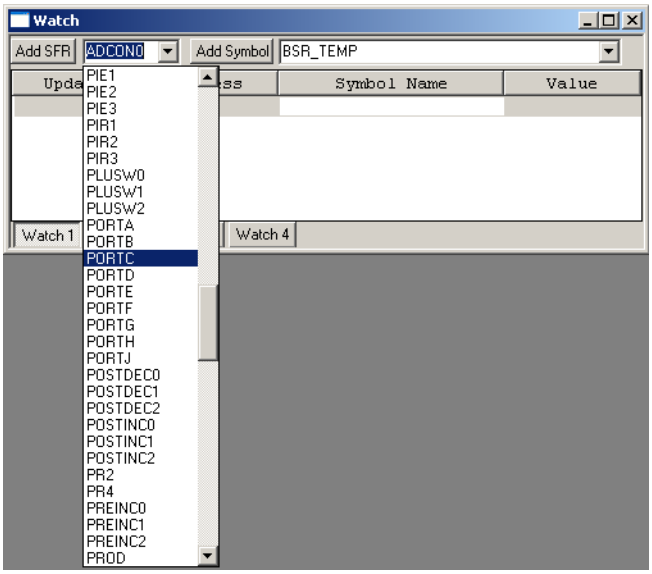
Next, press the Step Into icon or select *Debugger>Step Into* to single step to the code at `Main`.

FIGURE 4-18: CODE AFTER STEP INTO



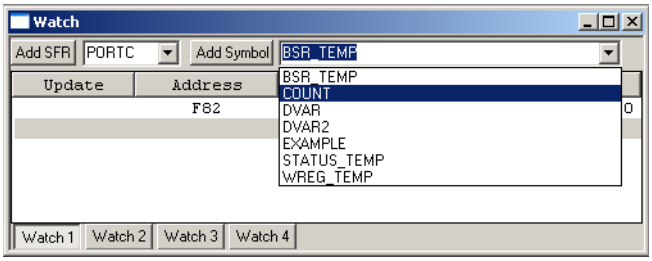
In order to see if the code is operating as intended (writing incremented values to `PORTC`), watch the values being written to `PORTC`. Select *View>Watch* to bring up an empty Watch window. There are two pull downs on the top of the Watch window. The one on the left labeled "Add SFR" can be used to add the Special Function Register, `PORTC`, into the watch. Select `PORTC` from the list and then click **Add SFR** to add it to the window.

FIGURE 4-19: WATCH – SELECT PORTC



The pull down on the right, allows symbols to be added from the program. Use this pull down to add the COUNT variable into the Watch window. Select COUNT from the list and then click **Add Symbol** to add it to the window.

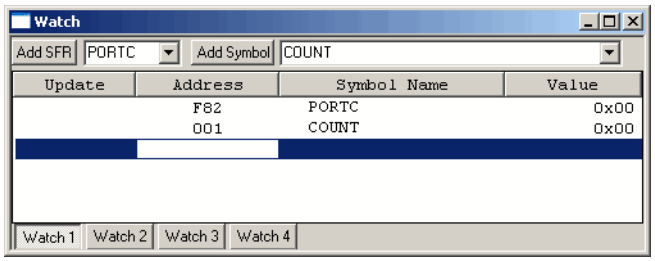
FIGURE 4-20: WATCH – SELECT VARIABLE "COUNT"



The Watch window should now show the address, name and value of the two registers. (The Update column is for use with the MPLAB REAL ICE in-circuit emulator.) At this point, the value for both symbols will be zero.

Note: Items can also be added to the Watch window by either dragging them from the SFR, File Register or Editor window or clicking a blank entry in the Watch window under symbol name and typing in the item.

FIGURE 4-21: WATCH – RESET VALUES



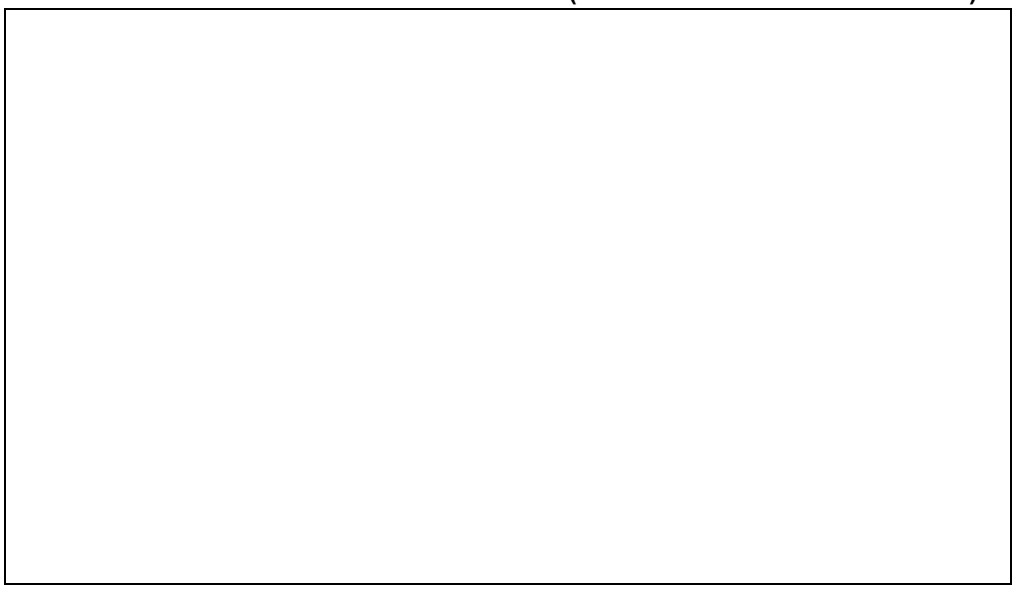
Now that Watch windows have been added, it is good idea to again save your workspace using *File>Save Workspace*.

You could continue single stepping through the code, but instead, set a breakpoint just before the first value is sent out to PORTC. To set a breakpoint, put the cursor on the following line:

```
movwf     PORTC    ; display COUNT on PORTC
```

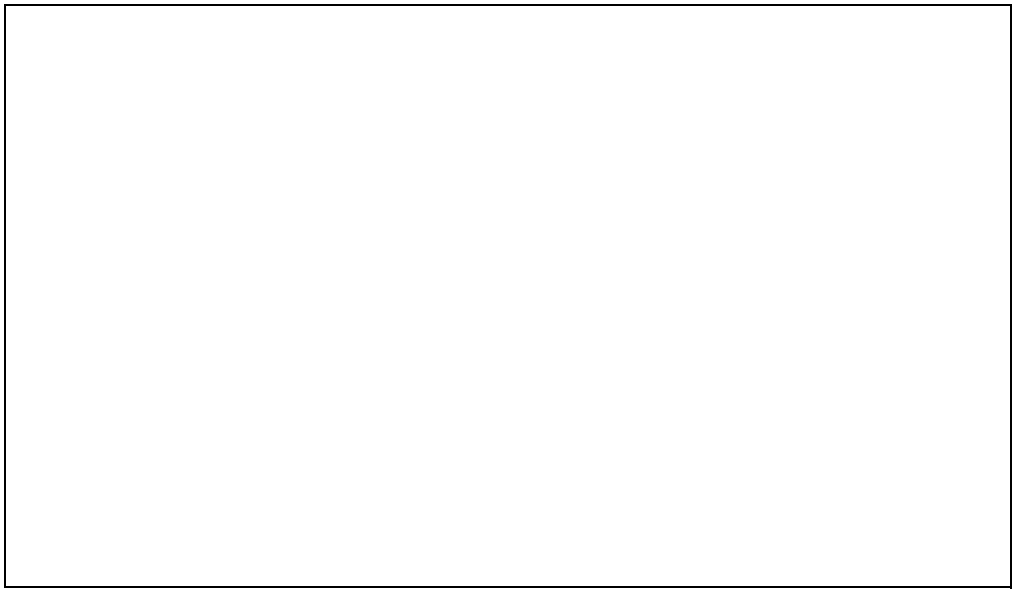
and click the right mouse button.

FIGURE 4-22: DEBUG CONTEXT MENU (RIGHT MOUSE CLICK ON LINE)



Select "Set Breakpoint" from the context menu. A red "B" will show on the line. You can also double click on a line to add a breakpoint if the Editor Options have been set up to do so (*Edit>Properties*, Editor Options dialog, **File Types** tab.)

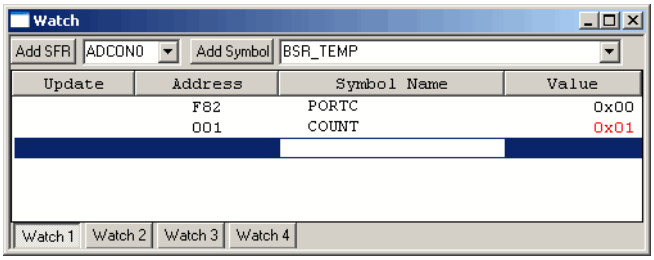
FIGURE 4-23: EDITOR WINDOW – SET BREAKPOINT



Select *Debugger>Run* to run the application. A text message “Running…” will briefly appear on the status bar before the application halts at this first breakpoint.

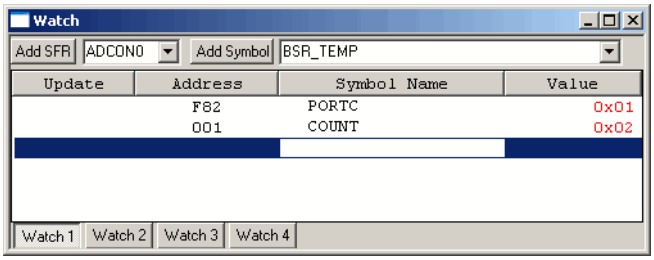
The Watch window should now show that the variable COUNT was incremented by one, but since the breakpoint is at the line before the move to PORTC executes, PORTC still has a value of zero.

FIGURE 4-24: WATCH – AT BREAKPOINT



Press the Run icon to execute the code until it hits this point again. The Watch window should now show both values incremented by one from their previous value.

FIGURE 4-25: WATCH – NEXT BREAKPOINT



This would seem to indicate that the program is working as designed. You can single step through the code, or run the code more times to verify that it is executing properly. If you single step into the delay loop, you will get stuck executing thousands of steps until reaching the end. To exit out of the delay loop, use *Debugger>Step Out*.

If you are interested in calculating your delay time, the data book could be used to determine how long each instruction would take in your delay loop and you would come up with a pretty accurate number. You can also use the MPLAB IDE Stopwatch to measure the delay. Your main interest should be the time each new value of COUNT is being displayed. If you set your breakpoint as was initially done, on the instruction that moves COUNT to PORTC, you can run to the next breakpoint at the same place to measure the time.

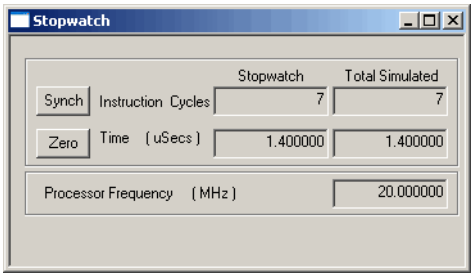
To use the Stopwatch, remove the breakpoint on PORTC by right clicking on the line and selecting “Remove Breakpoint”. Then, right click on the line

```
movf      COUNT,W,A  ; increase count and
```

and select “Set Breakpoint”. Finally, select *Debugger>Reset>Processor Reset*.

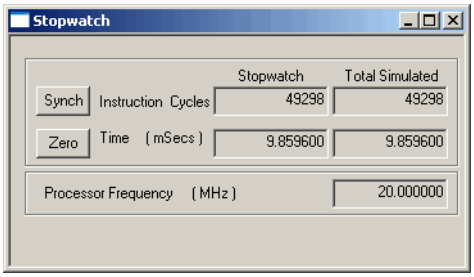
Use *Debugger>StopWatch* to bring up the Stopwatch dialog. Press *Debugger>Run* to run and then halt at the breakpoint. With the default processor frequency of 20 MHz, the Stopwatch should show that it took 1.4 microseconds to reach the first breakpoint.

FIGURE 4-26: STOPWATCH – AT FIRST BREAKPOINT



Execute Run again to go around the loop once, and note that the Stopwatch shows that it took about 9.8596 milliseconds. To change this, you can change the values in the delay loop. To change the Processor Frequency, select *Debugger>Settings*, **Osc/Trace** Tab.

FIGURE 4-27: STOPWATCH – AFTER DELAY



4.13 TUTORIAL SUMMARY

By completing this tutorial, you have performed the major steps for creating, building and testing a simple project. Tasks completed include:

- Selecting the device – the PIC18F8722.
- Using the Project Wizard to create a project, and using the wizard to:
  - select the MPLAB IDE built in MPASM assembler and MPLINK linker language tools,
  - add files for the project: a template file for the device selected and a linker script to build it properly.
- Writing some simple code to write a changing value out an I/O port.
- Building the project.
- And finally, testing the code with the simulator.

These are the essential steps for getting started with MPLAB IDE. You are now ready to continue exploring the capabilities of MPLAB IDE.

Chapter 5. Walk-Through and Detailed Tutorial

5.1 INTRODUCTION

This walk-through and tutorial is designed to help both new and experienced application developers. Step-by-step instructions for developing with MPLAB IDE and various Microchip development tools are given, where each step first discusses general information and then provides a specific example of use. All step examples tie together to form an example application.

Getting Started

- Selecting a Device
- Setting Up Configuration Bits
- Creating Source Code

Creating, Setting Up and Building a Project

- Creating a New Project
- Using the Project Wizard
- Setting Up the Language Toolsuite
- Naming and Locating the Project
- Adding Files
- Completing the Project
- Viewing the Project Window
- Setting Build Options and Configuration
- Building The Project

Debugging

- Choosing a Debugger
- Running Your Code
- Viewing Debug Windows
- Using Watch Windows
- Using Breakpoints
- Using Other Tools

Programming

- Choosing a Programmer
- Programming Your Part

Additional Resources

- Using Microchip Help

5.2 SELECTING A DEVICE

To begin application development, select the Microchip device you wish to develop code for.

To choose a device:

1. Select *Configure>Select Device*.
2. In the Select Device dialog, choose a device from the Device list box. Typing in the first few letters of the name will allow you to locate your device more quickly. The Microchip Tool Support section will reflect the available tool support for that device.
3. Click **OK**.

The device you have selected should now be reflected in the status bar on the bottom of the MPLAB IDE window.

Tutorial Step 1:

Select the PIC18F8722 microcontroller as the device. Many Microchip tools support this device (e.g., HPC Explorer Board).

5.3 SETTING UP CONFIGURATION BITS

Device configuration is set up in configuration register(s) by selecting appropriate bits. Consult your device data sheet for more information.

MPLAB IDE recognizes configuration bits set in code with `config` directives or commands. These values are displayed in the Configuration Bits window (*Configure>Configuration Bits*). Values may be set in the Configuration Bits window as well by deselecting "Configuration Bits set in code". Values set in the window override code definitions in MPLAB IDE but do not change the source code. See **Section 9.2 "Configuration Bits"** for more information.

Configuration bit window values are saved/restored on close/open of the workspace. If "Clear program memory upon loading a program" is checked in the Settings dialog (*Configure>Settings*), **Program Loading** tab, Configuration bits are cleared upon loading a program (i.e., build, import or open a project).

If your device supports external memory, and you select a mode which uses external memory in the Configuration bits, you will also need to set the amount of external memory used in the External Memory Settings dialog (*Configure>External Memory*). See **Section 9.4 "External Memory"** for more on information.

Tutorial Step 2:

For your selected device:

- You will use the default Configuration bits values, and therefore will not make any changes to the Configuration Bits window.
- Check the settings on the *Configure>Settings*, **Program Loading** tab. The following should be selected: "Clear memory before building a project" and "Clear program memory upon loading a program".
- The PIC18F8722 device can use external memory. However, for this tutorial, you will use microcontroller mode, or all on-chip memory.

5.4 CREATING SOURCE CODE

Select *File>New* to open an empty editor window in which to type your source code. To save your work, select *File>Save*. For more on the editor, see **Chapter 16. "Using the Editor"**.

Tutorial Step 3:

Create/locate a folder (directory), using Windows Explorer, where you will place application project files. Then, type the following code into an MPLAB IDE editor window, or cut and paste from here. Save the file as `cnt8722.asm` to the project folder when complete.

```
        title "PIC18F8722 Counting Program"
        #include <p18f8722.inc>

        udata  0x60      ;declare COUNT variable
COUNT  res 1             ; in bank 1

RST     code   00h       ;reset vector
        goto   Start

PGM     code             ;program vector
Start   clrf   WREG      ;clear W register
        movwf  PORTC     ;clear PORTC
        movwf  TRISC     ;config PORTC as outputs
Init
        clrf   COUNT     ;clear COUNT
IncCount
        incf   COUNT,F   ;increment COUNT
        movf   COUNT,W
        movwf  PORTC     ;display on Port C
        goto   IncCount  ;loop

        end
```

For more information on the assembly instructions, see the "*PIC18F8722 Family Data Sheet*" (DS39646), Instruction Set. For more information on MPASM assembler directives and expressions, see the *"MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User's Guide"* (DS33014) or MPASM assembler on-line help in MPLAB IDE.

5.5 CREATING A NEW PROJECT

A project contains the source files needed to build and test an application. A project configuration (workspace) includes the following: processor and display information, such as the nature, size and position of all open windows, toolbars, execution and debug settings. For more on projects and workspaces, see **Chapter 6. "Projects and Workspaces"**.

Project and workspace global setting may be selected on the **Project** and **Workspace** tabs of the Settings dialog (*Configure>Settings*).

Tutorial Step 4:

For the tutorial, keep the default setup for projects and workspaces. Most notably, this means you will be using the one-to-one project-workspace model.

5.6 USING THE PROJECT WIZARD

To create a new project in the current workspace, use the Project Wizard.

- Select *Project>Project Wizard*.
- The Project Wizard Welcome screen will appear. Click **Next>** to continue.
- The Project Wizard Select a Device screen will appear. Verify that the device you have already selected using the Select Device dialog is shown here. To change the device, use the drop-down menu. Click **Next>** to continue.

Tutorial Step 5:

You will use the device selected in step 1, so there is no need to make any changes.

5.7 SETTING UP THE LANGUAGE TOOLSUITE

You will now add language tools for use in the project. The Project Wizard Select a Language Toolsuite screen should be visible.

- Select a language toolsuite (Microchip or Third Party) for your project from the "Active Toolsuite" drop-down menu. Only those language toolsuites that are installed and work with the previously-selected device are available to select from this menu. To see all available (installed) toolsuites, check "Show all installed toolsuites".

  If you still don't see the desired toolsuite in this list, click **Help! My Toolsuite Isn't Listed!** for more information.
- A list of tools in the selected toolsuite are shown in a box under "Toolsuite Contents". A tool with a red "X" preceding it is not installed, or the path to the executable is not known to MPLAB IDE. To assign or check assignments of tools to executable files, click on the tool to show the executable and path under "Location of Selected Tool". Type in this text box to enter or change the assignment, or click **Browse** to find the executable.

  Note: Some language tools, like C compilers, must be purchased separately.
- Click **Next>** to continue.

Tutorial Step 6:

Set up the language tools for this tutorial as follows:

- For the "Active Toolsuite", choose "Microchip MPASM Toolsuite".
- Click on each of the "Toolsuite Contents" – MPASM Assembly, MPLINK Object Linker and MPLIB Object Librarian – to verify their "Location". Click **Browse** to find the executable if it is not listed.

This toolsuite is installed with MPLAB IDE and its tool executables are located in the `MPASM Suite` subdirectory of the Microchip installation directory.

5.8 NAMING AND LOCATING THE PROJECT

You will now enter a name and location for your new project. You can “Create New Project File” or “Reconfigure Active Project”.

To create a new project:

- Type in the path to an existing directory. If you type in the path to a new directory, you will be prompted to create the directory when you click **Next>**.
- Click **Browse** to browse to an existing directory or to one level above where you wish to place a new directory. Click **OK** in the Browse for Folder dialog. Complete the path if you are creating a new directory and then click **Next>**. You will be prompted to create the directory if it does not exist.

To reconfigure the current active project to be a new project, select one of the following:

- Make changes without saving – Make changes, but do not save to the project file.
- Save changes to existing project file – Make changes and save to the project file.
- Save changes to another project file – Enter or browse to another project and save changes to that file.

Tutorial Step 7:

Select “Create New Project File”. Click **Browse** and locate the application project directory you set up when you created your source code in step 3. Enter “cnt8722” for the project name. Click **OK**.

5.9 ADDING FILES

You will now add files to the project. The Project Wizard Add Existing Files screen should be visible.

- Choose the file(s) to add. Click on a file name to select that file. <Ctrl> + Click to select more than one file. Click **Add>>** to list file(s) to be added to the new project.
- To remove file(s) from the list, click on the file name(s) to select and then click **Remove**.
- Select a file-addition mode by clicking on the letter preceding the file name. For more on the meaning of each letter, see **Section 6.2.5 “Project Wizard – Add Files”**.
- You usually no longer need to add a linker script file to your project. For more on this, see **Section 7.3 “Linker Script Usage”**.
- Click **Next>** to continue.

Tutorial Step 8:

You will be adding one files to your project:

- Locate the file “cnt8722.asm” and click on it. Then click **Add>>** to add it to the project.

5.10 COMPLETING THE PROJECT

You should now review your project setup. The Project Wizard Summary screen should be visible.

- Check the summarized information on the project. If anything is incorrect, use **Back** to return to the dialog you need and change the information.
- Click **Finish** when you are satisfied with the project setup.

Tutorial Step 9:

Click **Finish** to complete the project setup.

5.11 VIEWING THE PROJECT WINDOW

If it is not already open, open the Project window by selecting *View>Project*. The workspace name should be visible in the top bar of the Project window. The name of the project should be visible inside the Project window at the top of the display. A tree structure listing file types and any added files will appear below.

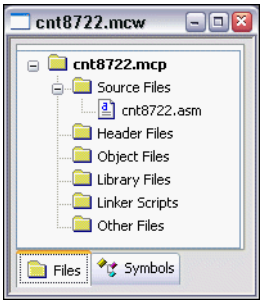
For more on this window, see **Section 13.20 "Project Window"**.

Tutorial Step 10:

Your project window should look like Figure 5-1.

- Right click on tree items to view various pop-up menus.
- Right click on "Source Files" and you will see "Add Files", which means you can add more source files to your project after you have initially set it up. Right click on the file `cnt8722.asm` and you will see "Remove", which is how you can delete files from your project.
- Right click in an open area of the window and select "Preferences" from the pop-up menu. This will open the Project-Display Preferences dialog.
- Although you will not be changing any preferences for this tutorial, notice the Version Control section. For more on using version control files in your projects, see **Section 6.6 "Using A Version Control System (VCS)"**.
- Click **Cancel** to close the dialog.

FIGURE 5-1: EXAMPLE PROJECT WINDOW



5.12 SETTING BUILD OPTIONS AND CONFIGURATION

MPLAB IDE has default settings for tools and files contained in a project. However, you may want or need to modify these settings.

To set build options, select *Project>Build Options>Project* or right click on the project name in the Project window and select Build Options from the pop-up menu.

- The Build Options dialog will open.
- Click the **Directories** tab and enter or **Browse** to paths for output files, include files, library files or linker script files.

Note: These are MPLAB IDE paths. Not all language tools use this information.

- Click a specific language tool tab (e.g., MPASM Assembler) and set up operational features for that tool.
- Click **OK**.

To override project settings for a particular project file, e.g., `ProjFile1.asm`, select *Project>Build Options>ProjFile1.asm* or right click on `ProjFile1.asm` in the Project window and select Build Options from the pop-up menu.

- The File Settings dialog will open.
- Click a specific language tool tab (e.g., MPASM Assembler) and set up operational features for that tool.
- Click **OK**.

To set up the build configuration, use the “Build Configuration” drop-down box on the Project Manger toolbar (Figure 5-2), if it is not grayed out. Determine whether you want to debug your code (select “Debug”) or if it is ready for release, meaning you are ready to program it into a device (select “Release”).

FIGURE 5-2: BUILD CONFIGURATION DROP-DOWN BOX

For more on this control, see **Section 8.3 “Build Configuration (Debug/Release)”**.

Tutorial Step 11:

Set up project build options:

- Select *Project>Build Options>Project*.
- Click the **MPLINK Linker** tab. Check the checkbox by “Generate map file”.
- Click **OK**.

The map file produced by the linker shows the memory layout after linking and indicates used and unused memory regions. This can be useful to verify code operation and correct memory allocation. For more information, see the “*MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User’s Guide”* (DS33014) or MPLINK linker on-line help in MPLAB IDE.

5.13 BUILDING THE PROJECT

Now you are ready to build your project. Select Build All from either:

- the menu bar (*Project>Build All*)
- the pop-up menu (right click on the project name in the Project window)
- the toolbar ().

During the build:

- For MPASM assembler, a status window will open showing you the progress and final outcome of the build. It will close when complete.
- The Output window will also open. This will contain information about the build, including any errors encountered

If your project does not build successfully, please check the following items and then build the project again:

- If there are reported errors in the Output window, double clicking on an error will indicate the corresponding line in your source code with a small blue pointer in the gutter of the source code window.
- Check the spelling and format of any code you entered in the editor window.
- Check that the correct language tool is being used for the project and for the project files.

Upon a successful build, the debug file (`*.cof`, `*.elf`, etc.) generated by the language tool will be loaded. This file allows you to debug using your source code and view your variables symbolically in Watch windows.

Tutorial Step 12:

Build your project by selecting Build All as described above. The code should build successfully. If you do encounter build errors, follow the instructions above for troubleshooting build errors.

5.14 CHOOSING A DEBUGGER

Choose a tool to help debug your code from *Debugger>Select Tool*. Microchip provides several types of debug tools:

- **MPLAB SIM** simulator – Simulate device operation in software.
- **MPLAB REAL ICE** in-circuit emulator – Debug your code in the device using this tool and special Flash devices with built-in emulation circuitry.
- **MPLAB ICE 2000** in-circuit emulator – Emulate most PIC MCUs operation in hardware, with access to device memory locations.
- **MPLAB ICE 4000** in-circuit emulator (End of life) – Emulate larger-memory PIC18 MCUs and dsPIC30F DSCs operation in hardware, with access to device memory locations.
- **MPLAB ICD 2** and **MPLAB ICD 3** in-circuit debuggers – Debug your code in the device using an ICD and special Flash devices with built-in debug circuitry.
- **PICkit 2** and **PICkit 3 Debug Express** – Debug your code in the device using this tool and special Flash devices with built-in debug circuitry.

See the tool-specific Readme file in the MPLAB IDE directory for a list of supported devices.

Once you have chosen a debug tool, you will see changes in the following on the IDE:

- Debugger menu – Several standard options will be available. Other tool-specific options may also appear.
- View menu – Depending on the debugger chosen, other debug options may appear.
- Debug toolbar – Several standard options will be available. Other tool-specific options may also appear.
- Status bar – The debug tool you have selected should now be shown.

Tutorial Step 13:

If you have purchased a hardware or third-party tool for application development, select it now (*Debugger>Select Tool*). If it is not visible, please consult the documentation that came with the tool for proper installation and setup with MPLAB IDE.

If you do not have an additional tool, select the built-in simulator, MPLAB SIM.

5.15 RUNNING YOUR CODE

No matter what debug tool you select, you will need to run (execute) your code to debug it.

- Select *Debugger>Reset>Processor Reset* or press <F6>. There should be a solid green arrow in the gutter of your source code window, indicating the first source code line that will be executed.
- Select *Debugger>Run* or press <F9> to run your application. You will see "Running…" appear on the status bar. Also, the arrow in the gutter will appear hollow.
- To halt program execution, select *Debugger>Halt* or press <F5>. The line of code where the application halted will be indicated by the solid green arrow.
- You may also single step through the application program by selecting *Debugger>Step Into* or press <F7>. This will execute the currently indicated line of code and move the arrow to the next line of code that will be executed.

Note:	You may click on the appropriate icon on the toolbar or use the hot key shown next to the menu item instead of selecting the menu item. This is usually the best approach for repeated running, halting and stepping when debugging.

Tutorial Step 14:

Reset your code (*Debugger>Reset*) and then run it by clicking the Run icon on the toolbar. (Hover your mouse over a toolbar icon to see its meaning.) Then stop the program by clicking the Halt icon.

5.16 VIEWING DEBUG WINDOWS

Standard debug windows allow you to view the contents of program, data or other types of device memory to aid in the debugging of your application. These windows are always visible on the View menu. Some items may be grayed out, however, if they are not supported by the device or debug tool. Additional debug windows may appear on the menu depending on the debug tool you select.

For a list of available windows, see **Section 12.2.3 "View"**.

Tutorial Step 15:

You will likely use the following windows to observe your running code:

- File (Editor) Window – Displays the actual assembly or C code.
- Program Memory Window – Displays your code as Opcode Hex, Machine or Symbolic.
- Disassembly Listing Window – Displays the disassembled code.
- Watch Window – Displays the values of registers and bits that you specify to "watch".

The file window should already be open, containing your code. You will be using the Watch window in the next step.

5.17 USING WATCH WINDOWS

In order to see what the application is doing, you will need to open a debug window to observe register values. Although many registers and their values are displayed in the File Register window and Special Function Register (SFR) window, you usually only wish to see a few application-specific registers. To set up your own list of registers to view, use Watch windows (*View>Watch*).

To add an item to the Watch window:

- Choose an SFR or Symbol from the drop-down list and then click the corresponding **Add** button
- Drag an item from the SFR, File Register or File window
- Click directly in the window under symbol name and typing in the item

The file register address of the symbols is listed first, followed by the symbol name and finally the value of the symbol.

For more on Watch windows, see **Section 8.6 "Watch Window"**.

Tutorial Step 16:

You will first set up the Watch window:

- Select COUNT from the symbol selection box at the top of the window. Click **Add Symbol** to add it to the Watch window list. If you want to quickly advance through the list, start typing COUNT after selecting the pull down icon.
- Select WREG from the SFR selection box at the top of the window. Click **Add SFR** to add it to the Watch window list. If you want to quickly advance through the list, start typing WREG after selecting the pull down icon.
- Select PORTC from the SFR selection box at the top of the window. Click **Add SFR** to add it to the Watch window list. If you want to quickly advance through the list, start typing PORTC after selecting the pull down icon.

You should now have these symbols in the Watch window. Next, you will run your code and watch the symbol values change as you step through the program.

- Select *Debugger>Reset>Processor Reset* to reset your application.
- Select *Debugger>Step Into* (or click the equivalent toolbar icon) until you have stepped to the following program line:

```
incf COUNT,F ;increment count
```

- Step one more time to see the value of COUNT in the Watch window change from 00 to 01.
- Step one more time to see the value of WREG in the Watch window change from 00 to 01.
- Step one more time to see the value of PORTC in the Watch window change from 00 to 01.

Note: The values in the Watch window are in color if they were changed by the previous debug operation, and are black if they were not changed by the previous debug operation.

5.18 USING BREAKPOINTS

Breakpoints are useful for conditionally halting the execution of your program in either the file (editor) window, the program memory window or the disassembly window. Single breakpoints may be set by double clicking on a line of code, or right clicking on the line and selecting Set Breakpoint from the pop-up menu. Multiple breakpoints may be set using the Breakpoint dialog (*Debugger>Breakpoints*). The number of breakpoints available will vary by debug tool and will be shown on the Breakpoint dialog.

For more on breakpoints, see **Section 8.4 "Breakpoints"**.

Tutorial Step 17:

To run your program again using a breakpoint:

- Select *Debugger>Reset>Processor Reset* to reset your application.
- Set a breakpoint by double clicking in the gutter next to the following line of code:

```
incf COUNT,F ;increment count
```

- Select *Debugger>Run* to run your program. Program execution should stop at the breakpoint. The values for all items in the Watch window should be 0x00. You may step through your code from this point to watch the values in the Watch window change again.

This concludes the tutorial. If you have a programmer and wish to program this code into a device, follow the instructions in the next two sections.

5.19 USING OTHER TOOLS

Other tools available for developing and debugging your code may be found under the Tools menu. These include:

- **DMCI - Data Monitor Control Interface** – Use this to view and control application variable values in real-time. Graphs are available to view array data.
- **RTOS Viewer** – Use for easy viewing of supported real-time operating systems' parameters. See **Section 13.21 "RTOS Viewer Window"**.
- **MPLAB Macros** – Use macros to record and playback repeated movements. This may prevent errors when procedures need to be repeated again and again. See **Chapter 10. "MPLAB Macros"**.
- ***Plug-ins*** – Other plug-in tools may be available, such as **MATLAB/Simulink**, for application development.

5.20 CHOOSING A PROGRAMMER

Once you have your code debugged and running smoothly, it is time to program a device and try it in your application.

Select a programmer from *Programmer>Select Programmer*. The programmer you use will depend on the device that you wish to program.

- **MPLAB PM3** device programmer – Used to program most PIC MCU, dsPIC DSC, Memory and KEELOQ security IC* devices.
- **PRO MATE II** device programmer (End of life) – Used to program most PIC MCU, Memory and KEELOQ security IC* devices.
- **PICSTART Plus**, **PICkit 1** or **2** or **3**, and **AN851** development programmers – Used to program specified PIC MCU devices in a development (nonproduction) environment.
- **MPLAB REAL ICE** in-circuit emulator – Can be used to program certain PIC MCU and dsPIC DSC Flash devices.
- **MPLAB ICD 2** and **MPLAB ICD 3** in-circuit debuggers – Can be used to program certain PIC MCU and dsPIC DSC Flash devices.

See the tool-specific Readme file in the MPLAB IDE directory for a list of supported devices.

* To program KEELOQ security IC devices, you will need to select either **MPLAB PM3** or **PRO MATE II** as the programmer, and then *Tools>KeeLoq Plugin* to generate the Hex code to program into the device.

Once you have chosen a programmer, you will see changes in the following on the IDE:

- Programmer menu – Several standard options will be available. Other tool-specific options may also appear.
- Programmer toolbar – Several standard options will be available. Other tool-specific options may also appear.
- Status bar – The programmer you have selected should now be shown.

5.21 PROGRAMMING YOUR PART

If you set the build configuration to "Debug" under **Section 5.12 "Setting Build Options and Configuration"**, you will now need to set it to "Release" and rebuild your project before you can program the device with the finished code.

In general, to program your part:

- Check the values for the Configuration bits in your code or in *Configure> Configuration Bits*.
- Check the programming settings in the *Programmer>Settings* dialog.
- Select *Programmer>Program*.

Once you have programmed your part, you are ready to try it in your application. If the part does not function as you had expected, you will need to debug your code and reprogram your part (or program a new part) before placing it in your application again.

5.22 USING MICROCHIP HELP

Microchip Technology provides on-line HTML help for MPLAB IDE and each of its development tools. Select *Help>Topics* to bring up a list of available help files in the Help Topics dialog. You may also bring up help topics for each dialog by clicking on its Help button or pressing **F1** if there is no help button. **F1** also brings up help topics for windows.

For more on Help, see **Section 8.8 "Microchip Help"**.

NOTES:

Part 3 – MPLAB IDE Features

NOTES:

Chapter 6. Projects and Workspaces

6.1 INTRODUCTION

Two major features of MPLAB IDE are projects and workspaces.

A **project** contains the files needed to build an application (source code, linker script files, etc.) along with their associations to various build tools and build options.

A **workspace** contains information on the selected device, debug tool and/or programmer, open windows and their location and other IDE configuration settings.

The best way to set up your project and its associated workspace is by using the Project Wizard. This will set up one project in one workspace.

To set up more advanced applications, you may set up your project and workspace manually. You can take advantage of the workspace setup and open multiple projects in one workspace. Also, you can tailor each project in a multiple-project workspace to create one part of a larger application (concurrent projects).

If you are working with a single assembly file application, you can use Quickbuild (*Project>Quickbuild*) to assemble your code and not create a project. You will, however, have an associated workspace, so your settings will be saved.

Projects

- Using the Project Wizard
- Creating/Updating any Project
- Setting Up a Project Structure – Relative Paths
- Project Folders and Files
- Using A Version Control System (VCS)
- Setting Up/Changing a Project

Projects and Workspaces

- Using a Single Project and Workspace (Default configuration)
- Using Multiple Projects in a Single Workspace
- Building an Application without a Project (Quickbuild)

Related Menus, Windows and Dialogs

- **Section 12.2.4 "Project"** (Menu)
- **Section 13.20 "Project Window"**
- **Section 14.20 "New Project Dialog"**
- **Section 14.21 "Project-Display Preferences Dialog"**
- **Section 14.28 "Settings Dialog"**

6.2 USING THE PROJECT WIZARD

The project wizard consists of several dialogs which will walk you through setting up your project.

- Project Wizard – Welcome
- Project Wizard – Select Device
- Project Wizard – Select a Language Toolsuite
- Project Wizard – Create the Project
- Project Wizard – Add Files
- Project Wizard – Summary

6.2.1 Project Wizard – Welcome

Select *Project>Project Wizard* to begin the setup wizard for projects.

Follow the dialogs in the Project Wizard to set up your new project.

You may also use the browse sequence in the Help viewer to view the steps in the wizard.

Click **Next** to continue.

6.2.2 Project Wizard – Select Device

Step 1: Select a Device – The value shown will be either the default MPLAB IDE device or a device that was previously selected using the Select Device dialog. Select a device for your project from the list or start typing in a device name.

MPLAB IDE is device-centric, meaning you must select a device first before tools that support that device are active on MPLAB IDE menus.

CAUTION
When using hardware tools, be careful not to choose a 5V device when a 3V device is physically attached to that tool. If 5V are applied to the 3V device, damage could occur. Check your device data sheet for voltage requirements.

Click **Next** to continue.

6.2.3 Project Wizard – Select a Language Toolsuite

Step 2: Select a Language Toolsuite

- Select a language toolsuite for your project from the "Active Toolsuite" drop-down menu. Although third-party language tools may be shown regardless of the device selected, only those Microchip language toolsuites that are installed and work with the selected device are available from this menu. To see all available (installed) toolsuites, check "Show all installed toolsuites."

If you still don't see the desired toolsuite in this list, click **Help! My Toolsuite Isn't Listed!** for more information.

- A list of tools in the selected toolsuite are shown in a box under "Toolsuite Contents". A tool with a red "X" preceding it is not installed or the path to the executable is not known to MPLAB IDE. To assign or check assignments of tools to executable files, click on the tool to show the executable and path under "Location of Selected Tool". Type in this text box to enter or change the assignment, or click **Browse** to find the executable.

Click **Next** to continue.

6.2.4 Project Wizard – Create the Project

Step 3: Create the project – Create a new project, or reconfigure the active project.

Create a New Project: Enter a name and location for your new project. To set the directory:

- Type in the path to an existing directory, or type in the path to a new directory, which will prompt you to create the directory when you click **Next**.
- Click **Browse** to browse to an existing directory or to one level above where you wish to place a new directory. Click **OK** in the Browse for Folder dialog. Complete the path if you are creating a new directory and then click **Next**. You will be prompted to create the directory if it does not exist.

Reconfigure the Active Project: Reconfigure the current active project to be a new project. This is useful for creating a new project based on the settings of an existing project.

- Make changes without saving – Make changes, but do not save to the project file.
- Save changes to existing project file – Make changes and save to the project file.
- Save changes to another project file – Enter or browse to another project and save changes to that file.

6.2.5 Project Wizard – Add Files

Step 4: Add Any Existing Files to Your Project – If you already have files that you would like to add to the new project, select them now:

- Choose the file(s) to add. Click on a file name to select that file. <Ctrl> + Click to select more than one file. Click **Add>>** to list file(s) to be added to the new project.
- Click the icon next to the file name to cycle through available modes (see below).
- To remove file(s) from the list, click on the file name(s) to select and then click **Remove**.

Most projects will require the addition of a linker script file. Consult your language tool documentation for details.

Click **Next** to continue.

File-Addition Modes

The MPLAB IDE project wizard now supports file-addition modes. Modes affect whether the project references the file by a relative or absolute path, which impacts project portability and the behavior of “Save Project As”.

Relative paths typically improve project portability, where as absolute paths tend to make projects less portable. “Save Project As” using relative paths causes a file to be copied when a project is saved to a new directory. “Save Project As” using absolute paths merely adjusts the file’s path when a project is saved to a new directory.

- [A] Auto – Allow MPLAB IDE to guess whether the file's path should be relative or absolute based upon the file's location. If the project directory contains the file, the reference is a relative path. Otherwise, the reference is an absolute path.
- [U] User – Reference the file with a relative path. This is the preferred way to add files created specifically for a project.
- [S] System – Reference the file with an absolute path. There are cases when this mode is useful, but these paths often require correction when a project is moved from one system to another.
- [C] Copy – Copy a file to the project directory and add a relative path to the copied file. Optionally, edit the file name to rename the local copy of the file. This mode provides a convenient way to incorporate linker scripts and template source files into a project without introducing an absolute path.

6.2.6 Project Wizard – Summary

Check the summarized information on the project. If anything is incorrect, use **Back** to return to the dialog you need and change the information.

Click **Finish** when you are satisfied with the project setup.

6.3 CREATING/UPDATING ANY PROJECT

The process for creating or updating a project is the same regardless of whether you are using a single or multiple project workspace.

1. Create/open a project in the current workspace. The project will appear in the Project window.
   a) Create a new project by selecting *Project>New*. Enter the new project name and location in the dialog.
   b) Open an existing project by selecting *Project>Open*.
2. Assign/change paths for project files by selecting *Project>Build Options>Project*, **Directories** tab.
3. Assign/change language tools in a project by first selecting *Project>Set Language Tool Locations* to specify the path to each language tool. Then select *Project>Set Language Toolsuite* to set the toolsuite for the project.
4. Setup/change language tool properties for the project by selecting *Project>Build Options>Project* or by right clicking on the project name in the Project window and selecting Build Options. Then click the appropriate language tool tab.
5. Enter files in the project by selecting *Project>Add Files* to Project, by right clicking on the project name in the Project window and selecting Add Files to Project, or by right clicking on the type of file in the Project window and selecting Add Files to enter that specific type of file.

> **Note:** You can add a file to a project, even if the file does not yet exist. This is useful for structuring a project before you actually developed the files. Just remember to create the files before you build the project or, obviously, the build will fail.

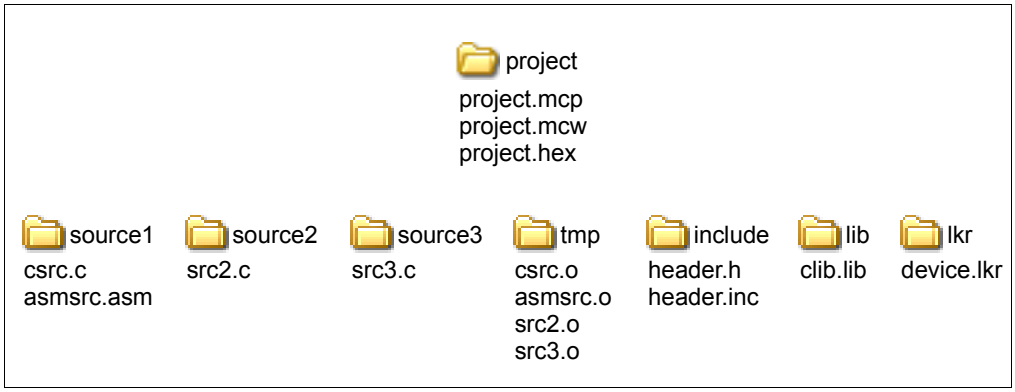
6. Delete files in the project by selecting *Project>Remove File From Project* or by right clicking on the file in the Project window and selecting Remove.
7. If you want language tool properties for individual files to differ from those set for the project, select *Project>Build Options>filename*, where filename is the name of the individual file, or right click on the file name in the Project window and select Build Options. These settings will apply only to the individual file.

6.4 SETTING UP A PROJECT STRUCTURE – RELATIVE PATHS

The following is an example of recommended project directory structure for using relative paths on the Build Options **Directories** tab. Keeping all project subfolders parallel allows for ease of grouping intermediates, include, library and linker script files.

Note:	Make sure you use unique names for your source code, or object files will be overwritten. E.g., `file.asm` is assembled producing `file.o`, and then `file.c` is compiled producing `file.o` which overwrites the first `file.o`.

FIGURE 6-1: RECOMMENDED PROJECT STRUCTURE



Project>Build Options>Project, Directories Tab

Output Directory: `C:\project`

Intermediates Directory: `..\tmp`

Assembler Include Search Path: `..\include`

Include Search Path: `..\include`

Libraries Search Path: `..\lib`

Linker Script Search Path: `..\lkr`

6.5 PROJECT FOLDERS AND FILES

Depending on how you have structured your project (see **Section 6.4 "Setting Up a Project Structure – Relative Paths"**) and how you have defined the project paths (*Project>Build Options>Project*, **Directories** tab), your project files may be all in the main project folder or spread out into separate subfolders. The structure represented in the Project window tree, however, will be based on which files you have specifically added to the project and the file type. The Project window "virtual" folders only contain files of a type specified by that folder's filter.

To add files, either (1) right click on the main project folder, select "Add Files" and the file will be placed in the correct subfolder based on its type or (2) right click on one of the folders and select "Add Files" if you already know the type.

You may also create subfolders by right clicking on a main folder and selecting "Create Subfolder". Then drag and drop the files into these folders.

The following sections describe the types of files in a project. The asterisk (*) denotes files that are shown in the Project window, Files tab.

6.5.1 Source Files*

Example files: `file.asm`, `file.c`, `file.s`

These are the only files that the toolsuite will accept as input to its file commands.

6.5.2 Header Files*

Example files: `header.inc`, `header.h`

MPLAB® IDE does not use this category when building. Consider it a means to document a project's dependency on a header file, and a convenient method to access these files. You can double click on a file in the Project window to open the file in an editor.

6.5.3 Object Files*

Example files: `ofile.o`

This category is for precompiled object files, not object files that result from assembling or compiling the project's source files. Functionally, the files in this category are indistinct from the library files, because both are merely linked in at the final phase of the build.

6.5.4 Library Files*

Example files: `fft.o`, `libfft.a`

The toolsuite should take all of the files in this folder, as well as the object files, and include them in the final link step.

6.5.5 Linker Scripts*

Example files: `linker.lkr`, `linker.gld`, `linker.ld`

There should be only one file in this folder. If the user has more than one linker script, only the first one has any effect. This is the linker script that the tool will use in the link step.

6.5.6 Intermediary Files

The project maintains this folder, but does not display it to the user. As source files are assembled and compiled, links are created to the resulting object files in this folder. Each and every object file in this folder should be included in the link step, in addition to the contents of the “Object Files” and “Library Files” folder.

Note:	A Clean will delete all of these files.

6.5.7 Output Files

Example files: `project.coff`, `project.map`

This folder is maintained by the project but not displayed. These are the final output files that result from the link step.

Note:	A Clean will delete all of these files.

6.5.8 Other Files*

Example files: `file.txt`, `stim.scl`, `39582b.pdf`

Any file that does not fit into any of the other categories will end up in this one.

You can add simulator files – SBS, STC, SCL – and double click on them to open the SCL Generator, Stimulus Controller, or editor, respectively.

You can add project-specific data sheets (PDFs) to this location in the Project window. Then, you can double click on the PDF to launch the data sheet. (This requires that a PDF reader is installed.)

Note:	You can add any file as an “Other” file. Be careful not to add source, header or other files necessary for a build to this folder or the language tools will not be able to find them.

6.6 USING A VERSION CONTROL SYSTEM (VCS)

If you want to use a version control system when developing your application code, you may now do so in MPLAB IDE using Projects.

- Select *Project>Version Control* to set up the version control application for use with MPLAB IDE. (See **Section 14.31 “Version Control Dialog”**.)

Note:	Folders/files must already exist in the VCS for MPLAB IDE to use them in a project.

To save an entire project (and workspace) in the VCS:

- Outside of MPLAB IDE, check in the entire project. When you next check it out and open it in MPLAB IDE, VCS-specific information will be available in MPLAB IDE (see below).

To save only some project files in the VCS:

- If you have common files contained in the VCS that you want to use in a specific project, you will need to check out those files outside of MPLAB IDE.
- Add the checked-out files to your project. You should now see VCS-specific information in MPLAB IDE (see below).

Once your project is set up for a VCS, you should notice the following changes to MPLAB IDE:

- The Project window will show version control information for the project files, i.e., display check-out status.
- Version control commands for VCS files may be selected by right clicking on a file in the project window.
- You may set up refresh options in the Project-Display Preferences Dialog, so you can be aware of file check-in/check-out status.
- The output window will contain a Version Control tab displaying activity between MPLAB IDE and the version control system.

Additional information on using version control systems currently supported is listed below.

6.6.1 Microsoft Visual Source Safe

Having some knowledge of VSS and MPLAB IDE is necessary to use VSS from within MPLAB IDE.

For more information on this version control system, please see the related web site.

- Microsoft Visual Source Safe: http://www.microsoft.com

Certain VSS files are required to be available locally (on your PC) for VSS to work with MPLAB IDE. If you are using VSS on a network drive, you will need to run `NETSETUP.EXE` to install the proper files on your PC.

6.6.2 PVCS

Having some knowledge of PVCS and MPLAB IDE is necessary to use PVCS from within MPLAB IDE.

For more information on this version control system, please see the related web site.

- PVCS: http://www.serena.com/Products/professional/vm/home.asp

6.6.3 CVS

Having some knowledge of CVS and MPLAB IDE is necessary to use CVS from within MPLAB IDE.

For more information on this version control system, please see the related web site.

- CVS: http://www.nongnu.org/cvs/

6.6.3.1 IMPORTING FILES

If your source files aren't already in the repository, you will have to import them to get started. This creates the directory structure in the repository and places all of the files there as well.

CAUTION
Do not place the workspace (MCW) or build-state (MCS) files on the repository. MPLAB IDE needs to be able to write to these files while running.

An example of an import command is given below. The `-d` argument is not given, since the assumption is that you have assigned the appropriate value to the `CVSROOT` environment variable. You want to be inside the very top directory of your source tree when you give this command. The command will automatically recurse into subdirectories.

```
$ cd TopOfSrcTree
$ cvs import -m "message" DirWithinCVSRoot VendorTag ReleaseTag
```

6.6.3.2 CHECKING OUT FILES

Even if you have your source files locally, you will have to perform a `cvs checkout` once (first-time use) before you can check out files from CVS in MPLAB IDE. This checkout creates the management files that CVS requires to perform commands on various files. All of these management files are stored in a hidden subdirectory of each directory in the source tree. Each of these hidden subdirectories is called `CVS`.

Trying to do a checkout overtop of existing source files without these supporting CVS directories will cause CVS to complain. You should either delete or move away the local copy of the sources before performing the checkout.

An example of performing the checkout is listed below:

```
$ cd DirABOVEWhereIWantMySrcTree
$ cvs checkout MyProject
cvs server: Updating MyProject
U MyProject/main.c
U MyProject/such-and-such-a-file.c
U MyProject/foo.h
```

It is important that you give this command from *above* the directory where you want your source tree to reside, because CVS will actually create the entire source tree within the current working directory.

6.6.3.3 ADDING A NEW DIRECTORY

You can use the `cvs add` command from within MPLAB IDE to add new source files as long as the required directory structure already exists in the repository. The `add` command will NOT work on a file if any of the required directories don't exist in the repository.

To create the directory structure in the repository, you must use the `cvs import` command once again. An example of this is as follows:

```
$ cd TopOfSrcTree
$ cvs import -m "message" MyProject/NewSubDir VendorTag ReleaseTag
N MyProject/NewSubDir/yet-another-file.h

No conflicts created by this import
```

Once again, you should give this command from within the directory that is the top-level directory of your source tree. But this time, when you specify the directory within the repository, you should give the path in the repository that will represent the new directory that you are adding. (That is what `MyProject/NewSubDir` represents in this example.)

6.6.4 Subversion

Having some knowledge of Subversion and MPLAB IDE is necessary to use Subversion from within MPLAB IDE.

For more information on this version control system, please see the related web site.

- Subversion: http://subversion.tigris.org/

If your source files aren't already in the repository, you will have to import them to get started. This creates the directory structure in the repository and places all of the files there as well.

CAUTION
Do not place the workspace (MCW) or build-state (MCS) files on the repository. MPLAB IDE needs to be able to write to these files while running.

6.7 SETTING UP/CHANGING A PROJECT

You can use the following steps to create a new project manually (using the Project Setup Wizard is recommended) or change an existing project's settings.

- Create a New Project
- New Project Window
- Setting Up a Project – Set Up Language Tools
- Setting Up a Project – Choose Toolsuite
- Setting Up a Project – Add Files
- Setting Up a Project – Set Build Options
- Setting Up a Project – Save and Build

6.7.1 Create a New Project

To create a new project in the current workspace:

- Using Windows Explorer, create a directory for your project (e.g. `C:\Proj1`).
- In MPLAB IDE, select *Project>New*. The New Project dialog will open asking you the name and location of the new project.
- Enter the Project name, e.g., `Proj1`.
- Enter the path to your new project. Click Browse to locate your directory.
- Hit **OK**.

To save the workspace with the new project in it:

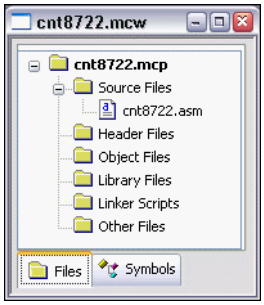
- Select *File>Save Workspace*.
- Enter the Workspace name, i.e., `cnt452`.
- Hit **OK**.

MPLAB IDE will create a project file with the extension `.mcp` and a workspace file with the extension `.mcw`.

6.7.2 New Project Window

If it is not already open, open the Project window by selecting *View>Project*. The workspace name should be visible in the top bar of the Project window. The name of the project should be visible inside the Project window at the top of the display. A tree structure listing file types will appear below.

FIGURE 6-2: EXAMPLE PROJECT WINDOW



6.7.3 Setting Up a Project – Set Up Language Tools

To add language tools for use in the project, select *Project>Set Language Tool Locations* or right click on the project name in the Project window and select Set Language Tool Locations from the pop-up menu.

- Click on a registered tool in a toolsuite to select it.
- Type in the “Location” or browse to the location of the tool executable by clicking **Browse**.
- Repeat this for each tool you wish to use in the project.
- When you are finished, click **OK**.

Now you will need to assign a toolsuite to the project.

6.7.4 Setting Up a Project – Choose Toolsuite

To select a toolsuite for use in the project, select *Project>Set Language Toolsuite* or right click on the project name in the Project window and select Set Language Toolsuite from the pop-up menu.

- Select your project toolsuite from the “Active Toolsuite” list.
- Click **OK**.

The toolsuite you use will depend on the device for which your are developing code.

- Microchip MPASM/C18 Toolsuites – for most PIC MCU, Microchip memory and KEELOQ® security IC devices
- Microchip ASM30/C30 Toolsuites – for PIC24 MCU and dsPIC DSC devices
- Microchip ASM32/C32 Toolsuites – for PIC32MX MCU devices
- Third Party Toolsuites – for various Microchip devices

6.7.5 Setting Up a Project – Add Files

You will now add files to the project. The basic types of files that are included in a project may be seen by clicking on the "+" sign to the left of the project name in the Project window.

Note:	You may enter the name of a file that does not yet exist. However, before you can build the project, you must create the file with a text editor such as MPLAB Editor.

To insert files in the project, select *Project>Add Files* to Project or right click on the project name in the Project window and select "Add Files" from the pop-up menu. Additionally, if you know the type of file you will be adding, you may right click on that type in the project window and select "Add Files" from the pop-up menu.

- The Add Files to Project dialog will open.
- Click on a file or enter its name to select it for insertion. If you do not see your file(s) listed, open the drop-down list under Files of Type to select the appropriate type of file. If you are unsure, select All Files (*.*).
- To select multiple files for insertion, use either Shift-Click to select a continuous range or Control-Click to select several discontinuous files.
- Click **Open**.

6.7.6 Setting Up a Project – Set Build Options

MPLAB IDE has default settings for tools and files contained in a project. However, you may want or need to modify these settings.

To set build options, select *Project>Build Options>Project* or right click on the project name in the Project window and select Build Options from the pop-up menu.

- The Build Options dialog will open.
- Click the **Directories** Tab and enter or **Browse** to paths for output files, include files, library files or linker script files.

Note:	These are MPLAB IDE paths. Not all language tools use this information.

- Click a specific language tool tab (e.g., MPASM Assembler) and set up operational features for that tool.
- Click **OK**.

To override project settings for a particular project file, e.g., `ProjFile1.asm`, select `Project>Build Options>ProjFile1.asm` or right click on `ProjFile1.asm` in the Project window and select Build Options from the pop-up menu.

- The File Settings dialog will open.
- Click a specific language tool tab (e.g., MPASM Assembler) and set up operational features for that tool.
- Click **OK**.

6.7.7 Setting Up a Project – Save and Build

At this point you should save your project by selecting *Project>Save* or right clicking on the project name in the Project window and selecting “Save”.

Now you are ready to build your project. Right click on the project name in the Project window and select “Build All” from the pop-up menu.

- For MPASM assembler, a status window will open showing you the progress and final outcome of the build. It will close when complete.
- The Output window will also open. This will contain information about the build, including any errors encountered.

If your project does not build successfully, please check the following items and then build the project again:

- Check the spelling and format of any code you entered in the editor window. If there are reported errors in the Output window, double clicking on an error will indicate the corresponding line in your source code with a green arrow in the gutter of the source code window.
- Check that the correct language tool is being used for the project and for the project files.

Upon a successful build, the debug file (`*.cof`, `*.elf`, etc.) generated by the language tool will be loaded. This file allows you to debug using your source code and view your variables symbolically in Watch windows.

6.8 USING A SINGLE PROJECT AND WORKSPACE

The most common configuration for application development in MPLAB IDE to is have one project in one workspace. Select *Configure>Settings*, **Projects** tab, and check “Use one-to-one project-workspace model”.

1. Select the device you will use for development (*Configure>Select Device*).
2. Create/open and set up a project in the current workspace.
3. Build the project.
   a) To build the whole project, select either *Project>Build All* or right click on the project name in the Project window and select Build All.
   b) To build only the files that have changed in a project, select either *Project>Make* or right click on the project name in the Project window and select Make.

Your application is now ready to debug. When your debug session is complete:

4. Close the project and its associated workspace by selecting *Project>Save Project* and then *Project>Close*.
5. Reopen the project and its associated workspace by selecting *Project>Open*.

6.9 USING MULTIPLE PROJECTS IN A SINGLE WORKSPACE

You may wish to have multiple projects in one workspace. This makes use of one workspace setup (i.e., selected device, debug tool, window setups, etc.) for many projects.

Some applications can benefit from using more than one project in a workspace. Consider the following examples.

EXAMPLE 6-1: BOOTLOADER

A bootloader has its own source code and its only function is to download and program a new version of an application in flash memory. The bootloader would always exist in a fixed location in program memory, and would be executed by a special function of the application. The bootloader would then download a program through a communications port and overwrite the existing application with a new version. Both the bootloader and the application share the same settings for the dsPIC30F device since they will exist on the same part, but they are independent pieces of code. The application knows only the entry address for executing the bootloader. The bootloader knows only the start address for the upgrade code.

EXAMPLE 6-2: TESTS

Another use of multiple projects is when developing a series of tests for a single application. These tests might exist in the same locations in program memory, but never be loaded simultaneously.

EXAMPLE 6-3: APPLICATION CONFIGURATION

A third possibility is that the application has multiple configurations for the end product. An application might be sold under multiple brands or with alternate languages. Each project could contain the branding information or language strings for the application in a separate resource file.

6.9.1 Setting Up Multiple Projects

To set up the workspace to use multiple projects:

1. Select the device you will use for development (*Configure>Select Device*).
2. Create/open and set up the first project in the current workspace.
3. Make sure that the following options are selected. Then repeat #2 for each project you wish to add to the workspace.
   a) *Configure>Settings*, **Projects** tab: uncheck "Use one-to-one project-workspace model"
   b) *Configure>Settings*, **Program Loading** tab: check "Clear program memory upon loading a program"
4. Make a project active by selecting *Project>Set Active Project>projectname.mcp*, where `projectname.mcp` is the project name, or right click on the project in the Project window and select Set As Active Project. The project name will appear in bold in the Project window.
5. Build the active project.
   a) To build the whole project, right click on the project name in the Project window and select Build All.
   b) To build only the files that have changed in a project, right click on the project name in the Project window and select Make.

Your application is now ready to debug. When your debug session is complete:

6. Save the current active project by selecting *Project>Save Project* or by right clicking on the project in the Project window and selecting Save Project.
7. Select another project as the active project, build this project and debug.
8. Remove the active project from the workspace by selecting *Project>Close* or by right clicking on the project in the Project window and selecting Close Project.
9. Close and save all projects and the associated workspace by selecting *File>Close Workspace*. You will be prompted to save.
10. Reopen all projects and the associated workspace by selecting *File>Open Workspace*.

6.9.2 Determining an Active Project

An active project is the project in the workspace that the following Project menu and toolbar commands will act on:

- Clean
- Find in Project Files
- Save Project
- Add Files to Project
- Select Language Toolsuite

All other Project menu, toolbar and right mouse menu commands will act on the project specified when the command is chosen. As an example, if you right click on a project in the Project window that is not the active project, you can build this project by selecting Build All or Make and this project's build results will be loaded into program memory.

To avoid confusion, it is recommended that you make active a project you will be developing and building.

The last project opened in a workspace will automatically be set as the active project. To set a project as the active project, select *Project>Set Active Project> project-name.mcp*, where projectname.mcp is the project name, or right click on the project in the Project window and select Set As Active Project. The project name will appear in bold in the Project window.

6.9.3 Using Concurrent Projects

You may wish to have multiple projects that create code to occupy different regions of program memory, thus creating one application. Projects used this way are considered concurrent projects.

To create a concurrent-projects workspace:

1. Begin with a multiproject workspace.
2. Uncheck the “Clear program memory upon loading a program” option on the *Configure>Settings*, **Program Loading** tab.

6.10 BUILDING AN APPLICATION WITHOUT A PROJECT

If you only want to assemble a single assembly file with MPASM assembler, you do not need to use projects but may use Quickbuild instead.

1. Select the device you will use for development (*Configure>Select Device*).
2. Make sure Quickbuild is active (*Project>Set Active Project>None* (Quickbuild mode)).
3. Open a file (editor) window and enter your assembly code (*File>New*), open an existing assembly file (*File>Open*) or click on an open edit window with your code to make the window active.

> **Note:** If a file window is not active, the Quickbuild option will be grayed out on the menu.

4. Select *Project>Quickbuild* file.asm to assemble your application, where file.asm is the name of your active assembly file.

Your application is now ready to debug. When your debug session is complete:

5. Save the current MPLAB IDE configuration to a file by selecting *File>Save Workspace*.
6. Restore the MPLAB IDE configuration for another debug session by selecting *File>Open Workspace*.

> **Note:** If you experience problems debugging your code, you may want to consider creating a project. The MPASM assembler creates a COD debug file which has known issues (**Section 11.5 "Limitations"**). Using the assembler with the MPLINK object linker removes these issues through the use of a COFF debug file.

Chapter 7. Programming Language Features

7.1 INTRODUCTION

MPLAB IDE supports the use of numerous assemblers and compilers for building code in several programming languages. Microchip provides free assemblers and linkers for PIC MCU and dsPIC DSC devices, as well as compilers (free student/academic/demo editions and for-purchase full versions). Third parties provide additional coverage with language tools for assembly, C and BASIC languages.

- Language Tool Setup
- Linker Script Usage
- Language Support Windows and Dialogs
- Language Support Tools

7.2 LANGUAGE TOOL SETUP

The installation of MPLAB IDE includes the assemblers and linkers for 8-bit (MPASM assembler/MPLINK object linker), 16-bit and 32-bit devices. Other language tools must be installed and then set up in MPLAB IDE to work with MPLAB IDE.

7.2.1 Language Tool Locations/Versions

Language tool locations can be set up when you create a project (*Project>Project Wizard*) or independently at any time using *Project>Set Language Tool Locations* and *Project>Select Language Toolsuite*. For more on these dialogs, see:

- **Section 6.2 "Using the Project Wizard"** - Select your language toolsuite while setting up your project.
- **Section 14.27 "Set Language Tool Location Dialog"** - Tell MPLAB IDE where to find the language tool executables in your desired language toolsuite.
- **Section 14.26 "Select Language Toolsuite Dialog"** - Select the language toolsuite for the project. You may also tell MPLAB IDE where to find the language tool executables in your desired language toolsuite.

Default language tool locations, and thus versions, are specified in the Set Language Tool Location dialog. If you want to use a non-default version of language tools for a specific project, use the Select Language Toolsuite dialog to set different executables for the toolsuite and then check the "Store tool locations in project" checkbox to use these tool locations, instead of the defaults, in this project.

7.2.2 Language Tool Options and Configuration

Once the location of language tool executables is defined and a project is set up, you can set options for individual language tools (e.g., MPASM assembler) on tabs of the same name on the Build Options dialog (*Project>Build Options>Project*). For more information, see **Section 14.5 "Build Options Dialog"**.

Language tools also need to be configured for generating code that can be debugged or code that is to be released in a production. Use the build configuration selection item, discussed in **8.3 "Build Configuration (Debug/Release)"**, to set your desired configuration.

if you want to use a specific language toolsuite with your project, but not change the default toolsuite, you can check "Store tool locations in project" on the Select Language Toolsuite dialog. This is on by default for starter kit installations.

Language tool setting can be declared for the entire project (as above) or can be overridden on a per-file basis (*Project>Build Options*, choose a file from the list).

7.3 LINKER SCRIPT USAGE

Linker script files are used by a linker to generate application code. You do not need to add a device-specific linker script file to your project; the linker will find the appropriate file for you. Exceptions are:

- Projects using the MPLAB Assembler for PIC24 MCUs and dsPIC DSCs
- Projects where you want to specifically add an edited linker script file

Standard linker script files are provided for each device and are located, by default, under `C:\Program Files\Microchip` in the following subdirectories:

- `MPASM Suite\LKR`
- `MPLAB ASM30 Suite\Support\`*`DeviceFamily`*`\gld`
  where *`DeviceFamily`* can be `dsPIC30F`, `dsPIC33F`, `PIC24F`, `PIC24H` or `generic`.
- `MPLAB C32 Suite\pic32mx\lib\ldscripts`

Special linker scripts are provided for use with C compilers. These files are located, by default, under `C:\Program Files\Microchip` in the subdirectories:

- `C:\MCC18\lkr`
  (or `MPLAB C18\lkr` for C Compiler version 3.30 or later.)
- `MPLAB C30\support\`*`DeviceFamily`*`\gld`
  where *`DeviceFamily`* can be `dsPIC30F`, `dsPIC33F`, `PIC24F`, `PIC24H` or `generic`.
- `MPLAB C32\pic32mx\lib\ldscripts`

Current linker scripts use conditional text to make the files flexible enough for different situations.

TABLE 7-1: CURRENT LINKER SCRIPT FILES

Tool	PIC10/12/16	PIC18	PIC24F/H, dsPIC30F/33F	PIC32MX
General	*DevNum*_g.lkr	*DevNum*_g.lkr	*DevNum*.gld	*DevNum*.ld
MPLAB® ICE 2000	*DevNum*_g.lkr	*DevNum*_g.lkr*	N/A	N/A
MPLAB ICE 4000 (End of Life)	N/A	*DevNum*_g.lkr*	*DevNume*.gld	N/A
MPLAB ICD 2/3	*DevNum*_g.lkr	*DevNum*_g.lkr	*DevNum*.gld	*DevNum*.ld
MPLAB REAL ICE™ In-Circuit Emulator	*DevNum*_g.lkr	*DevNum*i_g.lkr	*DevNum*.gld	*DevNum*.ld

* Not all PIC18 MCUs are supported.
N/A - Not applicable.
DevNum – the number associated with the device, e.g., 18f8722.lkr for the PIC18F8722 device
_g – Generic linker script includes conditional text to accommodate debug resources and PIC18 extended instruction.

Past linker scripts used separate files for different situations.

TABLE 7-2: PAST LINKER SCRIPT FILES

Tool	PIC10/12/16	PIC18	PIC24F/H, dsPIC30F/33F	PIC32MX
General	*DevNum*.lkr	*DevNum*.lkr *DevNum*_e.lkr	*DevNum*.gld	*DevNum*.ld
MPLAB® ICE 2000	*DevNum*.lkr	*DevNum*.lkr* *DevNum*_e.lkr*	N/A	N/A
MPLAB ICE 4000	N/A	*DevNum*.lkr* *DevNum*_e.lkr*	*DevNume*.gld	N/A
MPLAB ICD 2 (Debug Mode)	*DevNum*i.lkr	*DevNum*i.lkr *DevNum*i_e.lkr	*DevNum*.gld	*DevNum*.ld
MPLAB REAL ICE™ In-Circuit Emulator (Debug Mode)	*DevNum*i.lkr	*DevNum*i.lkr *DevNum*i_e.lkr	*DevNum*.gld	*DevNum*.ld

* Not all PIC18 MCUs are supported.
N/A - Not applicable.
DevNum – the number associated with the device, e.g., 18f8722.lkr for the PIC18F8722 device
No suffix – Standard linker script.
i – Reserved memory for debug resources is specified.
_e – For PIC18 devices: Extended instruction (XINST) mode is specified.
e – For PIC24/dsPIC devices: XY data is specified for use with MPLAB ICE 4000.

7.4 LANGUAGE SUPPORT WINDOWS AND DIALOGS

Several windows and dialogs are available to view language-related information:

- **Section 14.25 "Select Device Dialog"** – Determine if there is Microchip assembler/compiler support for your selected device. Green indicates that there is full support. Yellow indicates that there is beta support. Red indicates that there is currently no support. See third party sources listed on our website for other language tools.
- **Section 13.20 "Project Window"** – On the Files tab, you can arrange the source code files of your project, even into multiple projects (see **Chapter 6. "Projects and Workspaces"**.) Also, make use of predefined, device-specific header (include) and linker files found in the install directory of the Microchip assembler or compiler tool.
  On the Symbols tab, you can view the symbols in your code in a structured format.
- **Section 13.4 "Call Stack Window"** – View the software stack.
- **Section 13.14 "Locals Window"** – View local variables.
- **Section 13.7 "Disassembly Listing Window"** – View the corresponding assembly code for higher-level languages.
- **Section 8.6 "Watch Window"** – Several C-language constructs are supported in the Watch window.

7.5 LANGUAGE SUPPORT TOOLS

In addition to built-in windows and dialogs, there are several tools (under the Tools menu) that can be used to support programming efforts:

- Data Monitor and Control Interface – Control the input values of variables and display results graphically using this interface. Useful for motor control applications, but can be used in many others. For specific versions of this interface, see the AN901 and AN908 interfaces, too. See more on these tools under the Help menu.
- RTOS Viewer – For those using a real time operating system, this window is useful. See **Section 13.21 "RTOS Viewer Window"**.
- Third party support – Several tools are available as plug-ins to support language development. See **Section 3.3 "Third Party Tools"**.

Chapter 8. Debug Features

8.1 INTRODUCTION

MPLAB IDE contains additional features to aid in code debugging and general support. For information on debug features associated with a specific tool (including tool-specific windows and dialogs), see the documentation for that debug tool.

- Run vs. Step/Animate
- Build Configuration (Debug/Release)
- Breakpoints
- Trace Buffer Windows
- Watch Window
- Stopwatch
- Microchip Help

8.2 RUN VS. STEP/ANIMATE

Once a debug tool has been chosen, MPLAB IDE provides two basic modes of code execution: run and step (which includes animate).

8.2.1 Run Mode

In this mode, program code is executed until a breakpoint is encountered or until Halt is selected. Once program code execution is halted (stopped), MPLAB IDE status information will be updated.

Enter Run mode by either clicking the **Run** button on the Debug toolbar, selecting *Debugger>Run* from the menu bar, or pressing <F9> on the keyboard. Some tools provide additional types of run, such as "Run to cursor" from the right-mouse menu.

Automatically halt the Run mode by using a breakpoint or other type of break, e.g., "Halt on Trace Buffer Full".

Manually halt the Run mode by either clicking the **Halt** button on the Debug toolbar, selecting *Debugger>Halt* from the menu bar, or pressing <F5> on the keyboard.

8.2.2 Step Mode

There are several types of Step modes. However, all Step modes execute a limited number of program code lines and then halt.

Step Into

This is the most-used form of Step mode and is synonymous with single stepping.

For assembly code, this command executes one instruction (single or multiple cycle instructions) and then halts. After execution of one instruction, all the windows are updated.

For C code, this command executes one line of C code, which may mean the execution of one or more assembly instruction, and then halts. After execution, all the windows are updated.

Enter this Step mode by either clicking the **Step Into** button on the Debug toolbar, selecting *Debugger>Step Into* from the menu bar, or pressing <F7> on the keyboard.

Step Over

Execute the instruction at the current program counter location. At a `CALL` instruction, Step Over executes the called subroutine and halts at the address following the `CALL`. If the Step Over is too long or appears to "hang", click Halt.

Enter this Step mode by either clicking the **Step Over** button on the Debug toolbar, selecting *Debugger>Step Over* from the menu bar, or pressing <F8> on the keyboard.

For information on breakpoints and this function, see **Section 8.4.4.2 "MPLAB IDE Features Impacted When All Breakpoints are Used"**.

Step Out

Step out of a subroutine. If you are single stepping through subroutine code, you may finish executing the rest of the subroutine code and halt at the address following the subroutine `CALL` by using Step Out.

Enter this Step mode by either clicking the **Step Out** button on the Debug toolbar, or by selecting *Debugger>Step Out* from the menu bar.

For information on breakpoints and this function, see **Section 8.4.4.2 "MPLAB IDE Features Impacted When All Breakpoints are Used"**.

Animate

Animate causes the debugger to actually execute single steps while running, updating the values of the registers as it runs.

Animate runs slower than the Run function, but allows you to view changing register values in the Special Function Register window or in the Watch window.

To Halt Animate, use the menu option *Debugger>Halt* instead of the toolbar **Halt** or <F5>.

8.2.3 Mode Differences

Debug functions and/or features that work in running code do not always work when stepping or animating code. These include:

- Peripherals - Some peripheral modules will not work or not work as expected when code is stepped. See the documentation for your debug tool to see if the peripherals you are using will work when the MPLAB IDE is in a Step mode. Some tools allow you to choose to keep peripherals running even when code is being stepped, e.g., uncheck "Freeze peripherals on halt".
- Interrupts - Some interrupts will not work when code is stepped. Since some peripherals are not running, their interrupts will not occur.
- Speed - Stepping involves executing one line of code and then halting. Because open MPLAB IDE windows are updated on halt, stepping may be slow. A solution is to minimize as many windows as possible to improve execution speed.

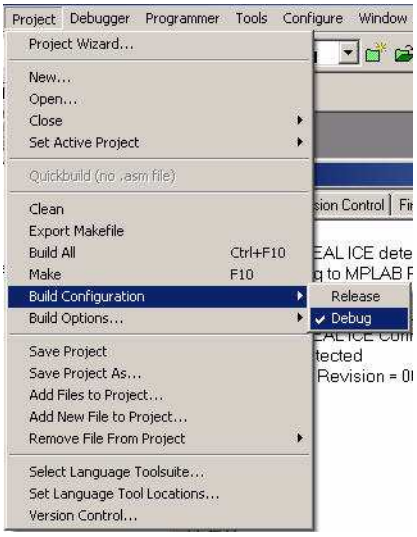
> **Note:** For some tools (MPLAB REAL ICE in-circuit emulator, MPLAB ICD 3), you can use the internal free-running clock (FRC) to minimize this effect. (The peripherals are not halted so no window updates.) However, this may result in peripherals running at different speeds.

8.3 BUILD CONFIGURATION (DEBUG/RELEASE)

For some language tools, MPLAB IDE provides an automated way to perform debug setup using the “Build Configuration” drop-down box on the Project Manager toolbar (Figure 8-1) or the “Build Configuration” item on the Project menu (Figure 8-2).

FIGURE 8-1: BUILD CONFIGURATION DROP-DOWN BOX

FIGURE 8-2: BUILD CONFIGURATION MENU ITEM



To set up language tools for debug, select “Debug”. When your code is debugged and you are ready to release it (program the finished code into a device), select “Release”.

8.3.1 Debug Details

When “Debug” is selected as the build configuration, a `__DEBUG` symbol will be created when the project is built. MPLAB IDE uses this symbol to set some debug options. You can make use of this symbol in your code to specify debug-specific sections of code.

For PIC18 MCU C code:

```
#ifdef __DEBUG
  fprintf(stderr,"This is a debugging message\n");
#endif
```

For MPASM assembly:

```
ifdef __DEBUG
  :
elseif
  :
endif
```

Other options necessary to debug operation will be set as needed for the selected language tools. As an example, the `-g` option needs to be used for the 16-bit compiler to enable debug features.

When you select your debug tool (*Debugger>Select Tool*), options specific to that tool will be set up. If you select either the MPLAB ICD 2 or 3 in-circuit debugger or the MPLAB REAL ICE in-circuit emulator as your debug tool and your selected device is a 16-bit device, the option "`--defsym=__ICD2RAM=1`" will be set for the 16-bit linker. This will set up the linker script to allocate reserved memory for debug.

Also, MPLAB IDE will check to see if you have changed debug tools or build configuration settings between builds and warn you.

8.3.2 Release Details

When you have finished debugging your code, selecting "Release" will release debug resources for regular operation.

8.4 BREAKPOINTS

Breakpoints allow you to specify conditional program halts so that you may observe memory, register or variable values after a run-time execution. You may set breakpoints in either the file (editor) window, the program memory window or the disassembly window.

8.4.1 Breakpoint Setup

There are several ways to set up breakpoints:

1. **Double Click on Line** – Double click on the line of code where you want the breakpoint. Double click again to remove the breakpoint. You must have set "Double Click Toggles Breakpoint" in *Editor>Properties*, ***ASM/C/BAS* File Types** tab, for this to work.
2. **Double Click in Gutter** – Double click in the window gutter next to the line of code where you want the breakpoint. Double click again to remove the breakpoint.
3. **Pop-up Menu** – Place the cursor over the line of code where you want the breakpoint. Then, right click to pop up a menu and select "Set Breakpoint". Once a breakpoint is set, "Set Breakpoint" will become "Remove Breakpoint" and "Disable breakpoint". Other options on the pop-up menu under Breakpoints are for deleting, enabling or disabling all breakpoints.
4. **Breakpoint Dialog** – Open the Breakpoint dialog (*Debugger>Breakpoints*) to set, delete, enable or disable breakpoints. You must select a debug tool before this option is available.

8.4.2 Breakpoint Symbols

Symbols relating to breakpoint operation are displayed in the gutter of each supported window. For information on these symbols, see **Section 13.3 "Code Display Window Symbols"**.

8.4.3 Breakpoints in Code

Breakpoints will auto-adjust when source code is modified. This means:

- Breakpoints set on addresses will stay with those addresses, regardless of the code associated with that address.
- Breakpoints set on source code lines will move with those lines of code.

When setting breakpoints in C code, care must be taken if switching between the C code listing and the associated assembly listing when debugging.

- When you set a breakpoint on a line of C code, a breakpoint is set at the first corresponding line of assembly code as well. Both listing displays will show a red breakpoint symbol at the corresponding line of code.

- When you set a breakpoint in the assembly listing on any line of assembly code that represents a single line of C code, a red breakpoint symbol will appear next to the assembly code line and a yellow breakpoint symbol will appear next to the line of C code.
- When you set multiple breakpoints in the assembly listing on lines of assembly code that represents a single line of C code, red breakpoint symbols will appear next to the assembly code lines and a yellow breakpoint symbol will appear next to the line of C code. The yellow breakpoint symbol will remain until *all* assembly breakpoints are removed.

8.4.4 Limited Number of Breakpoints

Starter kits, in-circuit debuggers (including PICkit 1, 2 and 3) and the MPLAB REAL ICE in-circuit emulator support a limited number of breakpoints. The number of breakpoints available is dependent on the device selected. To see the number of breakpoints available and keep track of the number you have used, view the **Section 12.3.4 "Device Debug Resource Toolbar"** on the MPLAB IDE desktop.

8.4.4.1 SETTING BREAKPOINTS

For starter kits and most in-circuit debuggers, breakpoints may be set in the Breakpoint dialog (*Debugger>Breakpoints*). Also, advanced breakpoints may be set by selecting *Debugger>Advanced Breakpoints*. (The contents of this dialog are device-dependent.)

For the MPLAB ICD 3 or MPLAB REAL ICE in-circuit emulator, event, sequenced and ANDed breakpoints may be set in the Breakpoint dialog (*Debugger>Breakpoints*).

8.4.4.2 MPLAB IDE FEATURES IMPACTED WHEN ALL BREAKPOINTS ARE USED

The following MPLAB IDE features use breakpoints to accomplish their functions:

- Step Over
- Step Out
- Run to Cursor
- Reset to Main

To use these features without using breakpoint resources, or when all available breakpoints have been used, select *Configure>Settings*, **Debugger** tab, and check the "Auto step to the target address..." checkbox. The above features will function by stepping code until they match the target address required. This will impact the speed of operation, sometimes severely, depending on the amount of code that must be stepped through.

If you attempt to use one of these features when no breakpoints are available, a dialog will be displayed telling you to enable Auto Step as described above.

8.5 TRACE BUFFER WINDOWS

Depending on the debugger you select, you may have trace capability. Trace data from the debug tool is captured in a trace buffer. Information in the trace buffer is used to populate the following MPLAB IDE windows:

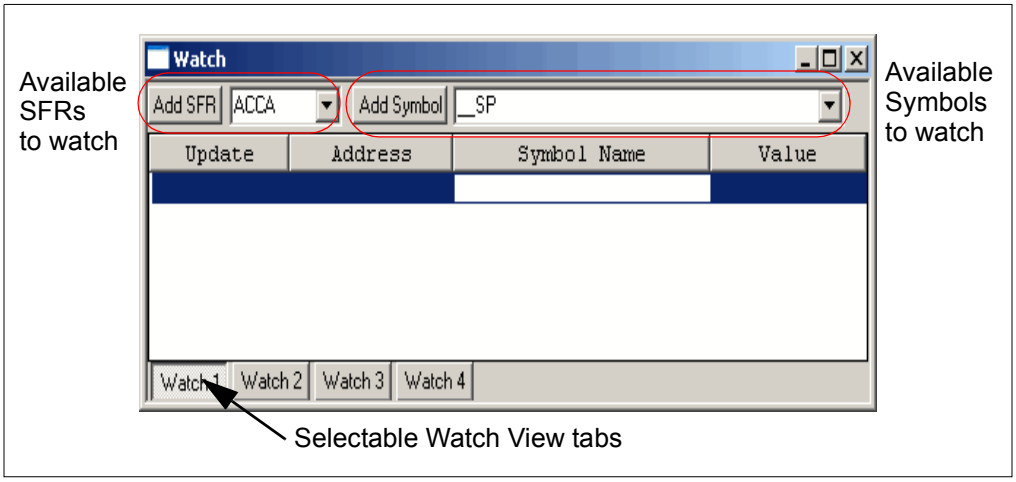
- **Section 13.24 "Trace Memory Window"**
- **Section 13.15 "Logic Analyzer Window"**

8.6 WATCH WINDOW

Although there are many debug windows available with MPLAB IDE via the View menu, a particularly useful one is the Watch window (or Watch windows if you count each Watch view as a separate window). This window allows you to select only the variables you wish to observe, instead of having to open several windows to see all the variables or one cluttered window with variables you don't need to observe.

The Watch window is made visible on the desktop by selecting *View>Watch*. It contains four selectable Watch views (via tabs) in which to view variables (SFRs and symbols).

FIGURE 8-3: WATCH WINDOW



Basic Usage

- Adding to a Watch View
- Changing a Watch View
- Deleting from a Watch View
- Updating a Watch View

Advanced Usage

- Moving Variables to/from a Watch View
- Saving/Restoring Watch Data
- C Language Usage – Watch Window
- C Language Usage – Local Variable Window
- Watch Window and Associated Dialogs

8.6.1 Adding to a Watch View

8.6.1.1 ADD AN SFR

To add a Special Function Register (SFR) to a Watch view:

- Select the SFR from the drop-down list and then click **Add SFR**.
- Right click in the window and select Add to open the Add Watch dialog. Select the SFR from the drop-down list. Select the Format for the SFR value. Click **Add SFR**.
- Drag-and-drop an SFR from either the SFR, File Register or Editor window to the Watch window.

8.6.1.2 ADD A SYMBOL

To add a symbol to a Watch view:

- Select the symbol from the drop-down list and then click **Add Symbol**.
- Right click in the window and select Add to open the Add Watch dialog. Select the symbol from the drop-down list. Select the Format for the symbol value. Click **Add Symbol**.
- Drag-and-drop a symbol from either the SFR, File Register or Editor window to the Watch window.

8.6.1.3 ADD TWO SAME-NAMED SYMBOLS

To view two same-named variables from different files:

The variable name will appear once in the Add Symbols list.

- Add from the Add Symbols list or type it in. The variable name will now appear in the Symbol Name column.
- Double click on the name and place a colon (:) in front of it. The Watch File Scope dialog will open.
- Select the file that relates to the variable you want to watch. Click **OK**.
- The file name will appear before the colon.

8.6.1.4 ADD AN ABSOLUTE ADDRESS

To add an absolute address to a Watch view:

- Click in the Address column of the first available row and enter a hex address.
- Right click in the window and select Add to open the Add Watch dialog. Fill in the Absolute Address information. Click **Add Address**.

8.6.1.5 ADD AN EXPRESSION

To add an expression to a Watch view:

- Click in the Symbol Name column of the first available row and enter a variable expression, e.g., `2*COUNT`, `PORTA + PORTB`.

8.6.1.6 ADD RADIX INFORMATION

To add (show) columns containing radix information (Hex, Decimal, Binary, Char):

- Right click on the column header bar and select each unchecked radix to add.
- Right click on the column header bar and select More to open the Watch dialog. On the **Column Setting** tab, check each radix to add. Click **OK**.

8.6.1.7 ADD A WATCH TAB

To add a Watch view to the window:

- Right click in the window and select Add Watch Tab. You can have as many as 16 Watch tabs.

8.6.2 Changing a Watch View

8.6.2.1 CHANGE A VALUE

To change a value in a Watch view:

- Click in the Value column and type the new value. Then click outside the column to see the value update.

8.6.2.2 CHANGE ENTRY PROPERTIES

To change the properties of an entry:

- Click on the entry to select it. Right click on the entry and select Properties to open the Watch dialog. Click on the **Watch Properties** tab to view and change properties of the entry. Click **OK**.

8.6.2.3 REORDER ENTRIES

To reorder entries in the Watch view:

- Click an entry to select it. Then drag-and-drop the entry to the desired location.
- Click on a column header to order the entries by the data in that column.

8.6.2.4 REORDER COLUMNS

To reorder columns in the Watch view:

- Click-and-drag a column header to the desired location.
- Right click on the column header bar and select More to open the Watch dialog. On the **Column Setting** tab, select a column header and use **Move Up**/**Move Down** to change the column location. Click **OK**.

8.6.2.5 CHANGE WATCH PREFERENCES

To change Watch preferences:

- Right click in the window and select Properties to open the Watch dialog. Click on the **Preferences** tab to change floating point information. Click the **General** tab to change font and color information. Click **OK**.

8.6.2.6 CHANGE WATCH TAB NAME

To change the name of a Watch view:

- Click on a tab to select the Watch view. Right click in the window and select Rename Watch Tab. Enter the new name in the dialog and click **OK**.

8.6.3 Deleting from a Watch View

8.6.3.1 DELETE AN ENTRY

To delete an entry:

- Click on the entry to select it, and then hit **Delete**.
- Click on the entry to select it. Right click on the entry and select Delete.

8.6.3.2 HIDE A COLUMN

To delete (hide) a column:

- Right click on the column header bar and select each checked header to delete.
- Right click on the column header bar and select More to open the Watch dialog. On the **Column Setting** tab, uncheck each header to delete. Click **OK**.

8.6.3.3 DELETE A WATCH TAB

To delete a Watch view:

- Click on a tab to select the Watch view. Right click in the window and select Remove Watch Tab. You can have no fewer than four Watch tabs.

8.6.4 Updating a Watch View

By default, Watch variable values are updated on a program halt. However, some tools (MPLAB SIM simulator, MPLAB REAL ICE in-circuit emulator) allow real-time updates.

To update values in real time:

- Select *Debugger>Settings*, **Animation/Realtime Updates** tab.
- Select “Enable realtime watch updates” and enter an update value.

8.6.5 Moving Variables to/from a Watch View

Variables may be moved to a Watch view from the following windows via drag-and-drop:

- SFR
- File Register
- Editor window

Variables may be moved from a Watch view to the following windows via drag-and-drop:

- Editor window

8.6.6 Saving/Restoring Watch Data

8.6.6.1 SAVE WATCH TAB

To save a Watch view:

- Click on a tab to select the Watch view. Right click in the window and select Save Watch Tab. In the Save As dialog, name the watch file for saving the data and click **OK**.
- Click on a tab to select the Watch view. Right click in the window and select Output to File. In the Save As dialog, name the text file for saving the watch data and click **OK**.

8.6.6.2 LOAD WATCH TAB

To load a saved Watch view:

- Click on a tab to select the Watch view. Right click in the window and select Load Watch Tab. In the Open dialog, select the watch file to load the data and click **OK**.

8.6.6.3 EXPORT WATCH VARIABLES

To export Watch variable values to a table:

- Click on an entry to select the SFR/Symbol. Right click in the window and select Export Table. In the Table Setup dialog, check or change export information and click **OK**.

8.6.6.4 IMPORT WATCH VARIABLES

To import Watch variable values from a table:

- Click on an entry to select the SFR/Symbol. Right click in the window and select Import Table. In the Table Setup dialog, check or change import information and click **OK**.

8.6.7 C Language Usage – Watch Window

If you are developing your application code in C, the following features are available in the Watch window to you:

- Bitfield value mouseover available. Enable/disable this feature using the right mouse button menu in the Watch window. The Watch window must be the active window (in focus) for this feature to work.
- Click in the Symbol Name field and enter a pointer name, e.g., `*count`. The variable must be a pointer to an intrinsic type or structure.
- Click in the Symbol Name field and enter a structure member name, e.g., `porta.ra0`. The variable must be of the form `struct.membername`.
- Click in the Symbol Name field and enter a structure pointer member name, e.g., `porta->pin`. The variable must be of the form `struct->pointername`.

Symbol names may be typed, copied/pasted or dragged/dropped. Also, you can select from the SFR/Symbol lists, and then edit the Symbol name to add the extra info.

Note: The Watch window will display only 32 bits for the 16-bit compiler type `long long int` and only 16 bits for the PIC18 MCU compiler type `short long int`. Change the display through the Watch dialog.

Example 1:

```
int J = 2;
int* pint = &J;
```

Enter in Watch: `*pInt`, Value: 2

Example 2:

```
char[] string = "Test";
char* pstring = string;
```

Enter in Watch: `*pstring`

Right click on the symbol name (`pstring`) and select Properties. In the Watch dialog, on the Watch Properties tab, select "ASCII" from the "Format" drop-down menu.

Value: "Test"

Example 3:

```
typedef struct
{
int str_int;
char str_ch;
} TestStruct;
TestStruct struct1;

TestStruct* pstruct1 = &struct1;
pstruct1->str_ch = 'A';
```

Enter in Watch: `*pstruct1`, Value: tree for struct.

Enter in Watch: `struct1.str_int`, Value: 'A'

8.6.8 C Language Usage – Local Variable Window

In addition to the Watch window, the Local Variable window may be used to watch local C variables. See **Section 13.14 "Locals Window"** for more details.

8.6.9 Watch Window and Associated Dialogs

The following MPLAB IDE windows and dialogs are associated with Watches:

- **Section 13.25 "Watch Window"**
- **Section 14.3 "Add Watch Dialog"**
- **Section 14.29 "Table Setup Dialog"**
- **Section 14.32 "Watch File Scope Dialog"**
- **Section 14.33 "Watch/Locals Dialog"**

8.7 STOPWATCH

Several debug tools have a stopwatch feature to time the execution of sections of code.

- MPLAB SIM: *Debugger>Stopwatch*
- MPLAB ICE 4000: *Debugger>Stopwatch*
- MPLAB ICD 2 in-circuit debugger: *Debugger>Advanced Breakpoints*, **Emulator Features** tab
- MPLAB ICD 3 in-circuit debugger: *Debugger>Breakpoints*, **Stopwatch**
- MPLAB REAL ICE in-circuit emulator: *Debugger>Breakpoints*, **Stopwatch**

8.8 MICROCHIP HELP

Microchip Technology provides on-line HTML help for MPLAB IDE and each of its development tools. Select *Help>Topics* to bring up a list of available help files in the Help Topics dialog.

8.8.1 Types of Help Available

Tool help is available from the MPLAB IDE Help menu. Help on specific dialog boxes or windows is available from those items. Limitations of a tool are available from the Settings dialog.

8.8.1.1 HELP MENU

Help for each development tool is available from the Help menu. Most help files include the following:

- Tool Overview
- Tutorials/Examples of Use
- General Usage Information
- Troubleshooting Information
- Reference Material

8.8.1.2 DIALOG BOX

Dialog box help is available from a **Help** button on each dialog. Clicking Help displays step-by-step instructions for using a dialog box to complete a task.

8.8.1.3 WINDOW

Window help is available from the <F1> key. Press the <F1> key on your keyboard while a window is active to see information on the contents of that window.

8.8.1.4 LIMITATIONS

Debug tool limitations are available for display after you have selected a device (*Configure>Select Device*) and set the debug tool (*Debugger>Select Tool*). You may then open the Settings dialog (*Debugger>Settings*) and click on the **Limitations** tab.

Both debug and programmer tool limitations are accessible by opening the tool help file (*Help>Topics*) and looking for "Limitations" in the table of contents or index.

8.8.1.5 MOUSEOVERS

If you want more information on a toolbar control, but don't know what it's called, hover the mouse pointer over the control. A box should pop up with the name of the control. Now you can search for information on that item in the Help file.

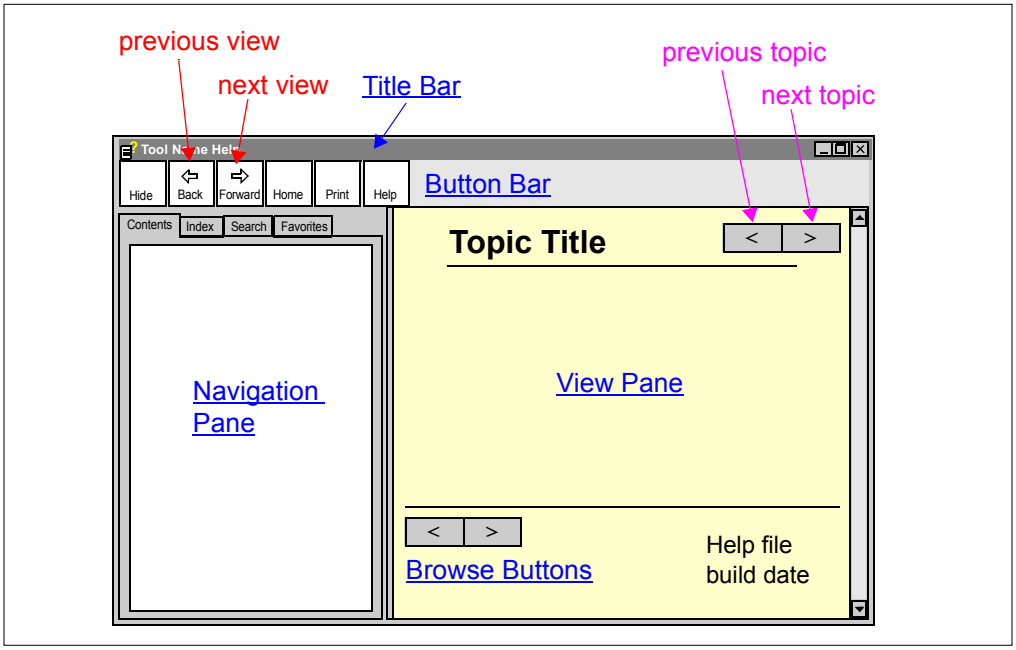
Note:	Some toolbar controls are tool-specific and do not belong to the IDE. If you are not sure, deselect any tools you are using to see if the items disappear. If they do, then those items belong to that tool, so you should consult the Help file for that tool for more information on the specific control.

To set up mouseovers of variables in code, see MPLAB Editor Help.

8.8.2 Navigating in the Help Viewer

The main features of the help viewer are shown below. Click on the blue, underlined text to go to more information.

FIGURE 8-4: HELP VIEWER - MAIN FEATURES



Title Bar

The title bar displays the name of the current help file.

Button Bar

The button bar contains the following functions:

- Show/Hide – Show or Hide the Navigation Pane.
- Back/Forward – Go back/forward to previously-viewed topics. These buttons will be grayed until you begin moving through help topics.

- Home – Return to first topic.
- Print – Print currently displayed topic.
- Help – Show this topic.

Browse Buttons

Browse buttons are located on each displayed topic in the view pane.

- Click **<** to browse to the previous help topic, according to the order on the **Contents** tab.
- Click **>** to browse to the next help topic, according to the order on the **Contents** tab.

Navigation Pane

This pane helps you navigate through the topics in the current help file.

- **Contents** tab – Click to view a functionally organized list of all Microchip Help topics. Click on the + next to a folder to open the folder and see the topics contained within. Click on a topic listed here to view it in the right pane of the viewer.
- **Index** tab – Click to view an alphabetized list of help topics.
- **Search** tab – Click to search for words in the help topics.
- **Favorites** tab – Click to create and maintain a list of “favorite” topics.

To close this pane, click on the **Hide** button (which will change to a **Show** button) on the button bar. To view this pane, click on the **Show** button (which will change to a **Hide** button.)

View Pane

This pane displays the help topic contents, along with browse buttons and the help file build date.

Click on the blue, underlined text to activate a hyperlink:

- Jump to another topic. To return, click **Back** or browse to other topics.
- Open a pop-up. To close the pop-up, click outside the pop-up box.
- Open another help file.

> **Note:** This function uses an ActiveX control to open the other compiled HTML help file. Therefore, you may receive an Internet Explorer warning, since the Help viewer uses some IE components and security settings. Click “Yes” to launch the other help file.

Click on the **arrow** to the right of blue, underlined text to activate drop-down text:

- Example drop-down text
  Click on the small arrow to the right of the text to alternately open/close drop-down text.

NOTES:

Chapter 9. Device-Related Features

9.1 INTRODUCTION

Many MPLAB IDE features are related to viewing and changing device components. After selecting a device (*Configure>Select Device*), MPLAB IDE determines the memory and related settings for that device and configures its displays accordingly.

- Configuration Bits
- Program and Data Memory
- External Memory
- Stack
- ID Memory
- Peripherals

9.2 CONFIGURATION BITS

Configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations. Consult your device data sheet for more information.

9.2.1 Setting in MPLAB IDE or in Code

Configuration bits may be set in code (i.e., using the `__config`/`config` command or init code) or using the Configuration Bits window (*Configure>Configuration Bits*). Although you can use both methods, you probably should choose one to avoid confusion and overwriting of bit values.

If you want to change Configuration bit values in code, do not make changes to the window. Every time you build, the code values will be loaded into the window and you will lose any changes you have made in the window.

If you want to change Configuration bit values in the window, do not enter any in code until you have completed development. Then put your window values into code, rebuild the project (which will then enter the code values into the window) and perform a final run to make sure all is correct.

You can also deselect "Configuration bits set in code" in the Configuration bits window to override code values with values in the columnar display portion of this window *until the project is rebuilt*; i.e., if the configuration bits are set in code, then the configuration window will take those values when the project is built even if "Configuration bits set in code" check box is unchecked.

9.2.2 Saving/Clearing Values

Configuration bit window values are saved/restored on close/open of the workspace.

If "Clear configuration bits upon loading a program" is checked in the Settings dialog (*Configure>Settings*), **Program Loading** tab, Configuration bits are cleared upon loading a program (i.e., build, import or open a project).

9.2.3 Setting the Watchdog Timer

By default, the watchdog timer (WDT) is enabled on most devices. This could interfere with program execution. You may wish to disable the WDT unless you know you will be using it in your program.

The WDT pre/postscaler is set in the Configuration bits as well.

9.2.4 Setting for External Memory Use

If your device supports external memory, and you select a mode which uses external memory in the Configuration bits, you will also need to set the amount of external memory used in the External Memory Settings dialog (*Configure>External Memory*).

9.3 PROGRAM AND DATA MEMORY

Program and data memory windows are set up by MPLAB IDE for the selected device. For devices with external memory options, see **Section 9.4 "External Memory"** for how to setup and display that type of memory.

Device selection will determine the memory ranges you see in the memory windows.

TABLE 9-1: MEMORY BY DEVICE

Device	Program Memory (words)	Data Memory (bits)	Architecture
PIC10F2XX, PIC12X5XX, PIC16X5X, PIC16X5XX MCUs	12-bit	8	Harvard
PIC12X6XX, PIC16 MCUs	14-bit	8	Harvard
PIC18 MCUs	16-bit / 2-byte	8	Harvard
PIC24 MCUs, dsPIC DSCs	24-bit	16	Modified Harvard
PIC32 MCUs	32-bit	32	MIPS

9.3.1 Viewing Program Memory

Program memory may be viewed by selecting *View>Program Memory*. Device default values are visible until a project is built. Then, values corresponding to program code will be seen. External program memory will also be visible in this window, if used.

Program memory may be cleared on a build by setting options on the *Configure>Settings*, **Program Loading** tab.

For more information, see:

- **Section 13.19 "Program Memory Window"**

For PIC32MX MCU device, see:

- **Section 13.6 "CPU Registers Window (PIC32MX Devices Only)"**
- **Section 13.16 "Memory Window (PIC32MX Devices Only)"**

9.3.2 Viewing Data Memory

Data memory may be viewed by selecting:

- *View>File Registers* - View all data memory.
- *View>Special Function Registers* - View only the SFR portion of data memory.
- *View>Watch* - Select specific data memory registers to view.
- *View>EEPROM* - View EEPROM data.

Device default values are visible until a project is built. Then, depending on the code, data memory values may change. Data values change most when the program is run.

Data memory may be cleared on a build by setting options on the *Configure>Settings*, **Program Loading** tab.

For more information, see:

- **Section 13.10 “File Registers Window”**
- **Section 13.23 “Special Function Registers Window”**
- **Section 13.25 “Watch Window”**
- **Section 13.8 “EEPROM Window”**

For PIC32MX MCU devices, see:

- **Section 13.16 “Memory Window (PIC32MX Devices Only)”**
- **Section 13.22 “SFR/Peripherals Window (PIC32MX Devices Only)”**

9.3.3 Determining Memory Usage

To see how much program or data memory is being used by a program at any time, select *View>Memory Usage Gauge*. This will give a visual and well as numeric value of used memory.

For more information, see **Section 13.17 “Memory Usage Gauge”**

9.3.4 Determining the Checksum

There is a checksum toolbar that may be used to find the checksum. Select *View>Toolbars>Checksum* to make this visible.

For more information, see **Section 12.3.6 “Checksum Toolbar”**

9.4 EXTERNAL MEMORY

Some Microchip devices allow the extension or replacement of program memory resources with external (off-chip) memory devices. MPLAB IDE handles this device feature as described here.

- Configuration Bits and External Memory Setup
- External Memory Enabled in MPLAB IDE
- Program Memory Window

9.4.1 Configuration Bits and External Memory Setup

A device that supports external memory must be set up to use external memory via the device Configuration bits.

For PIC18F6XXX/8XXX devices, a Program Memory mode is set by using Configuration bits. Depending on the device, the Processor Mode Select bits are located in a configuration register which is programmed when the device is programmed.

For PIC18C601/801 devices, External Bus Data Width is set as a Configuration bit.

To set the values of Configuration bits for your selected device, open the Configuration Bits window by selecting *Configure>Configuration Bits*. In the Category column, find the bits you need to set and click on them to change their value. See **Section 13.5 “Configuration Bits Window”** for more information on this window.

Note:	Configuration bits may also be set in code using `__config`. Refer to the device data sheet and header file (`.inc` or `.h`) for more information.

Information on device memory modes and the external bus is provided below. For more information, consult the device data sheet.

9.4.1.1 PIC18F6XXX/8XXX PROGRAM MEMORY MODES

Certain PIC18F6XXX/8XXX devices are capable of operating in any one of several Program Memory modes, using combinations of on-chip and external program memory. Please see the data sheet for your device to determine if it supports external memory.

For supported devices, available Program Memory modes are as follows:

- The **Microprocessor** mode permits access only to external program memory; the contents of the on-chip Flash memory are ignored. The program counter permits access to a linear program memory space and defines the amount of accessible program memory (see **Section 9.4.1.3 "Program Counter"**).
- The **Microprocessor with Boot Block** mode accesses on-chip Flash memory from address 000000h to the end of the boot block. Above this, external program memory is accessed all the way up to the program counter accessible limit (see **Section 9.4.1.3 "Program Counter"**). Program execution automatically switches between the two memories, as required.
- The **Microcontroller** mode accesses only on-chip Flash memory. Attempts to read above the physical limit of the on-chip Flash causes a read of all '0's (a NOP instruction).
- The **Extended Microcontroller** mode allows access to both internal and external program memories as a single block. The device can access its entire on-chip Flash memory; above this, the device accesses external program memory up to the program counter accessible limit (see **Section 9.4.1.3 "Program Counter"**). As with Boot Block mode, execution automatically switches between the two memories, as required.

In all modes, the device has complete access to data RAM and EEPROM. For more information, consult the device data sheet section "Memory Organization".

9.4.1.2 PIC18C601/801 ROMLESS DEVICES

For PIC18C601/801 devices, all program memory address space is external. The on-chip program counter permits access to a linear program memory space and defines the amount of accessible program memory (see **Section 9.4.1.3 "Program Counter"**).

There is a provision for configuring the last 512 bytes of general purpose user RAM as program memory, called "Boot RAM". See the device data sheet for more information.

9.4.1.3 PROGRAM COUNTER

The size of the program counter will determine how much program memory can be accessed, e.g., a 21-bit program counter permits access to a 2-Mbyte (1-Mword) program memory space (on-chip, off-chip or a combination of both types of program memory).

9.4.1.4 EXTERNAL MEMORY INTERFACE

The External Memory Interface is a feature that allows the microcontroller to access external memory devices (such as Flash, EPROM, SRAM, etc.) as program or data memory.

PIC18 Devices

Using the MEMCON register, the following may be configured:

- External bus enable and disable
- 16-Bit mode – Word Write mode, Byte Select mode or Byte Write mode
- Wait – Table read/write bus cycle wait counts (0-3 TCY)

For more information, see the External Memory Interface section of the device data sheet.

9.4.2 External Memory Enabled in MPLAB IDE

To set up MPLAB IDE (and a selected debug tool) to recognize external memory, select *Configure>External Memory*. Then check “Use External Memory” and enter a range. See **Section 14.10 “External Memory Setting Dialog”** for more on this dialog.

Some debug tools may require additional setup for external memory, usually on the *Debugger>Settings* dialog. See the documentation for the tool for more information.

9.4.3 Program Memory Window

The Program Memory window will contain all enabled program memory, i.e., on-chip, external or a combination of both. See **Section 13.19 “Program Memory Window”** for more on this window.

For hardware tools, external memory contents may need to be uploaded before the current values are reflected in this window.

9.5 STACK

The hardware stack may be viewed by selecting *View>Hardware Stack*. For more information, see **Section 13.12 “Hardware Stack Window”**.

For 16 and 32-bit devices, a call (software) stack window is available. See **Section 13.4 “Call Stack Window”**.

For PIC32MX devices, some CPU registers contain stack information. To view these registers, see **Section 13.6 “CPU Registers Window (PIC32MX Devices Only)”**.

9.6 ID MEMORY

The ID memory may be viewed and set by selecting *Configure>ID memory*. ID memory may be cleared on a build by setting options on the *Configure>Settings*, **Program Loading** tab.

For information on ID memory, see **Section 14.30 “User ID Memory Dialog”**.

9.7 PERIPHERALS

Special function registers are used for peripheral operation and control. To view SFR’s, select *View>Special Function Registers*. For more information, see **Section 13.23 “Special Function Registers Window”** or **Section 13.22 “SFR/Peripherals Window (PIC32MX Devices Only)”**.

A special window for viewing LCD control on devices that support this is available by selecting *View>LCD Pixel*. For more information, see **Section 13.13 “LCD Pixel Window”**.

SFR’s are part of data memory and as such are visible also in the File Register window (*View>File Registers*), along with General Purpose Registers. To view only a few SFR’s, use a Watch window, described in **Section 8.6 “Watch Window”**.

NOTES:

Chapter 10. MPLAB Macros

10.1 INTRODUCTION

MPLAB IDE supports the creation of macros, or the recording and playback of keyboard and/or mouse operations. Once a macro has been created, it may be saved to a file for future use.

- Using Macros
- Macro Menu and Toolbar
- Macros Dialog

10.2 USING MACROS

To enable macros, select *Tools>MPLAB Macros*. A Macro menu and toolbar will appear on the desktop (see **Section 10.3 "Macro Menu and Toolbar"**).

To disable macros, select *Macros>Exit Macros*.

By default, macros record only keyboard keystrokes. To record both keystrokes and mouse movements:

- Select *Macros>Macro Options* to open the Macros dialog (see **Section 10.4 "Macros Dialog"**).
- On the **Options** tab, check "Record Mouse actions" under "Recording".
- Uncheck "Full Speed" under "Playback"
- Click **Apply** to apply the changes or **OK** to apply the changes and close the dialog.

To begin recording a macro, select *Macros>Record Macro* or click the Record Macro toolbar button.

To end recording a macro, select *Macros>Stop Recording*, click the Stop Recording toolbar button or hit <Ctrl> + <Break>.

To play back a macro, select *Macros>Play Macro* or click the Play Macro toolbar button.

To cancel either recording or playback, hit <Ctrl> + <Esc>.

To save a macro to a file, select *Macros>Save Macro* or click the Save Macro toolbar button.

To open an existing macro file, select *Macros>Open Macro* or click the Open Macro toolbar button.

10.3 MACRO MENU AND TOOLBAR

The Macro menu contains the following functions:

- Record Macro – Begin recording keystrokes and/or mouse movements.
- Stop Recording – End recording keystrokes and/or mouse movements.
- Play Macro – Play back keystrokes and/or mouse movements.
- Open Macro – Open a macro file.
- Save Macro – Save recorded keystrokes and/or mouse movements to a macro file.
- Macro Options – Set macro options in a dialog. See **Section 10.4 "Macros Dialog"**.
- Exit Macro – Disable macros.

The Macro toolbar contains several of the same functions as the Macro menu. Mouse over a toolbar button to see its function name pop up.

10.4 MACROS DIALOG

Set up macro options in this dialog.

- Recording – Record Mouse actions: To record mouse movements in addition to keystrokes, this checkbox must be checked.
- Playback – Full Speed: If no mouse movements are recorded, this checkbox must be checked. If mouse movements are recorded, this checkbox must be unchecked.
- Windows System Command: A list of keystrokes that can be used instead of using menu or toolbar commands.
  - <Ctrl> + <Break>: Concludes macro recording.
  - <Ctrl> + <Esc>: Cancels recording and/or playback.

For more information, see **Section 10.2 "Using Macros"**.

Part 4 – MPLAB IDE Reference

NOTES:

Chapter 11. Troubleshooting

11.1 INTRODUCTION

This section is designed to help you troubleshoot any problems, errors or issues you encounter while using MPLAB IDE. If none of this information helps you, please see the Preface for ways to contact Microchip Technology.

Topics covered are:

- Common Problems – A list of commonly-encountered problems.
- Frequently-Asked Questions (FAQ) – A list of frequently asked questions.
- Error Messages – A list of error messages produced by this tool.
- Limitations – Limitations of the tool.

11.2 COMMON PROBLEMS

The following is a list of common problems and their solutions:

- **Your code builds with errors or does not run correctly once a tool is selected.**

  You need to select "Debug" from the "Build Configuration" drop-down box when using some hardware debug tools. For programmer tools, select "Release".

- **Your code keeps resetting.**

  For many devices, the Watchdog Timer is enabled by default. Be sure to turn off this device feature in the configuration bits (e.g., add the assembler directive `CONFIG WDT=OFF` to the beginning of code for a PIC18 device.) Also, check other configuration bits settings (*Configure>Configuration Bits*) for other device resets.

- **You cannot get your hardware tool to work with MPLAB IDE.**

  (1) Did you install the correct drivers for the tool? With the tool powered and connected to your PC, locate the Device Manager for your Windows OS and check under "Microchip Tools" to see if your driver is listed. If not, go to the MPLAB IDE installation directory, locate the directory for your tool and its "Drivers" subdirectory, and open the HTML instruction file to determine how to install the drivers.

  (2) Is everything connected properly? Please consult the tool documentation for proper setup, even if it seems obvious.

  (3) Is the tool powered correctly? Make sure you use the power supply provided with the tool for correct operation.

  (4) Is the target powered? Is its oscillator functioning?

  (5) Is the device code-protected? Please check the device configuration bits to determine this.

  (6) Are you using a USB hub? Is it powered? Try connecting directly into the PC.

11.3 FREQUENTLY-ASKED QUESTIONS (FAQ)

- **MPLAB IDE used to work, but now it will not launch**

  Items to check are:

  (1) Did you recently install another version of MPLAB IDE? Please see the MPLAB IDE Readme file for information on switching between versions of MPLAB IDE.

  (2) Did you recently install other software? If so, it may have overwritten a system dll in such a way that other applications cannot use it. Consult the software documentation and the Windows OS documentation to determine how to repair or reinstall the correct system dll.

  (3) Did you recently edit your PC registry? MPLAB IDE uses the registry to find related files. Try reinstalling MPLAB IDE.

  (4) Do you have virus protection software on your machine? Try other programs on your PC to see if they are functioning correctly. If not, you may have a virus.

- **I cannot get my language tools to work with MPLAB IDE.**

  You may need a newer version of the language tools to work with MPLAB IDE. Please go to our web site or third party web site to get the latest version of your language tool.

  Also, check the Set Language Tool Location dialog and Select Language Toolsuite dialog to verify that you are using the correct language tools version for your project.

- **The Project window contents are skewed from the window frame.**

  This is a rendering issue. Grab and resize the window to make the contents snap to the window.

- **I have recently ported my projects from one PC to another. They don't work or worked for a while but now do not.**

  Project files (.mcp) are portable. Workspace files (.mcw) may or may not be portable. If you try to move your projects and workspaces between PCs with different operating systems or different hardware configurations, you will need to update your workspace or create a new one.

  See **Section 15.5 "Saved Information"** for more on what is saved in these files.

- **My project was working, I made some changes, and now it won't work or works incorrectly**

  Rebuild your project. If you are using a hardware tool, you may also need to reprogram/download code to the tool.

- **Some MPLAB IDE objects are grayed out.**

  There are many reasons why an MPLAB IDE object may be grayed out:

  (1) Certain context-sensitive Project menu and toolbar items need an active project set before they will become available. Check *Project>Set Active Project* and make sure None (Quickbuild) is not chosen. An active project's name will be in bold in the Project window.

  (2) Some tools do not support all MPLAB IDE functions. See the tool documentation for details.

  (3) Some objects are part of a mutually exclusive group, i.e., you have the other option selected, so this option is grayed out.

  (4) If you use all available breakpoints, some MPLAB IDE features cannot operate. See **Section 8.4.4.2 "MPLAB IDE Features Impacted When All Breakpoints are Used"**.

  If you believe the object being grayed out is an error, consult the Readme for MPLAB IDE file under "Known Problems".

- **When stepping through high-level code, if I step into a function with no source code, a bunch of Program Memory windows start popping up.**

  MPLAB IDE will open another Program Memory window if you step into an instruction that does not have an associated line of source code (e.g., library routines), and the program memory window does not have focus.

  To avoid this, first select *Configure>Settings*, **Debugger** tab. Then check “Browse for source if file not found” and “Show disassembly if source is unavailable”.
- **I cannot find all of my object files.**

  Make sure you have your project directory structure set up in a parallel manner, or your object files will be grouped in the same directory as your source files instead of the project `tmp` directory.

  Also, make sure your source files use unique names (`file1.asm` and `file2.c`, not `file.asm` and `file.c`) so that they will not produce object files with the same name, which will overwrite each other.
- **I can't Halt Animate.**

  To Halt Animate, use the menu option *Debugger>Halt* instead of the toolbar Halt or **F5**.
- **I tried to Step Over / Step Out of a function/subroutine, and now my program is hung up.**

  Stepping over or out of a function or subroutine will actually execute all or some of the code in that function/subroutine. Therefore, there is the potential to be caught in an endless loop. Select Halt to regain control in MPLAB IDE.
- **I don’t see my problem here**

  Because MPLAB IDE is an integrated environment, errors or problems that appear to be caused by MPLAB IDE may actually be caused by an integrated tool. Please check on-line help files and other documentation for:

  - the language tool you are using (e.g., MPASM assembler).

  - the debug tool you are using (e.g., MPLAB ICD 2).

  - the programmer tool you are using (e.g., PICSTART Plus).

11.4 ERROR MESSAGES

- **Build Failed**

  There are many reasons why a build might fail. In general, check the following:

  (1) The code in the editor window. Look for misspelling of directives, functions, or variables, and incorrect formatting. Also, non-standard characters can cause the build to fail.

  (2) The language tools used. Make sure the tools you are using for the build are the correct ones for your selected device, e.g., you cannot build a dsPIC DSC project with the MPLAB C18 C compiler.
- **Cannot Find *filename***

  The file *filename* was not created due to an error in the build. This means you will need to debug your code before a build can complete successfully.
- ***filename* is missing**

  Check your installation for missing files.
- **Hex file not found**

  The project did not generate a hex file. Errors in your code will prevent hex file generation and cause this error.
- **Language Tool Not Properly Installed**

  After installing a new version of MPLAB IDE, make sure you provide MPLAB IDE with the correct location of any language tools.

- **No Debugging Information Available**

  Make sure you have selected the option to create a `.cod` or `.cof` (recommended) file with your build.
- **Unresolved Breakpoints**

  This message displays when breakpoints aren't set on a valid line, i.e., when breakpoints are set on optimized code, or code that does not have a valid assembly instruction associated with the breakpoint line. For information on how to fix an unresolved breakpoints, see **Section 14.4 "Breakpoints Dialog"**.

11.5 LIMITATIONS

MPASM/COD Limitations

Some MPASM assembler limitations can appear to be MPLAB IDE limitations. These include:

- When the assembler is used alone to generate executable code (i.e., without the linker as in **Quickbuild**), it creates a COD debug file, which has limitations.

  The COD format is an old format that is maintained for legacy reasons. Therefore, it is recommended that you use the linker with the assembler to generate a COFF file, which has none of the following limitations:
  - A **62** character length restriction for file and path names
  - COD files with EEData may not view properly in the Disassembly window
  - Only the first word of two-word instructions show in the Disassembly window
  - No source code debug beyond program memory address 0xFFFF.
- Include file path information entered in MPLAB IDE will NOT be used by the assembler.

Using the MPLINK object linker with the MPASM assembler will resolve these issues. See MPASM assembler documentation for more information.

Recent Workspaces List

If a workspace's name has a '.' in the name, you will not be able to select the workspace from the Recent Workspaces list.

Floating Frame with Docked Windows

MPLAB IDE may have rendering issues when opening a workspace that contains multiple docking windows in a floating frame.

Configuration Bits Window

For PIC32MX MCUs, some configuration registers are too large (have too many bits) to be displayed in the configuration window. These registers can only be set in code.

Chapter 12. Desktop

12.1 INTRODUCTION

The MPLAB IDE desktop is a resizable window that operates independent of the rest of the menu items. The desktop consists of a menu bar, tool bars, a status bar and any open windows and/or dialogs. The bars are covered here. Windows and dialogs are discussed in **Chapter 13. "Windows"** and **Chapter 14. "Dialogs"** respectively.

Topics covered in this chapter:

- Menu Bar
- Toolbars
- Status Bar
- Grayed out or Missing Items and Buttons

12.2 MENU BAR

All MPLAB IDE functions are accessible as menu items through the menu bar located across the top of the desktop. Menu items followed by ellipses (...) will open a dialog.

Shortcut (hot) keys for menu items are listed next to the menu item. Example: The shortcut keys for Print are Control-P (CTRL+P). You may set up hot keys under *Configure>Settings*, **Hot Keys** tab.

Menu items may be grayed out for various reasons. See **Section 12.5 "Grayed out or Missing Items and Buttons"**.

Available menus are:

- File
- Edit
- View
- Project
- Debugger
- Programmer
- Tools
- Configure
- Window
- Help

12.2.1 File

Below are the menu items in the File menu. For more on the editor and the items shown here, see **Chapter 16. "Using the Editor"**.

New

Displays an empty editor window named "Untitled" for the file. (See **Section 13.9 "File (Editor) Window"**.) When you close the window, you will be prompted to name the file.

Add New File to Project

Requests a name for the new file in a dialog (see **Section 14.11 "File Management Dialog"**). On **Save**, the file is added to the project and a new file window is opened.

Open

Opens an existing source file. You may select multiple files in the Open dialog (see **Section 14.11 "File Management Dialog"**) by clicking the filenames while holding down the CTRL or Shift key.

Close

Closes the active editor window. If the file has changed since it was saved last, you will be prompted to save changes.

Save

Saves the active editor window to disk under its original file name.

Save As

Opens the Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of an "Encoding" list box. Select the type of file encoding here. Allows you to save the active editor window to disk under a new file name.

Save All

Saves all open editor windows to disk.

Open Workspace

Opening a workspace prompts the user to close the previous workspace before opening the new one. For more on workspaces, see **Chapter 6. "Projects and Workspaces"**.

Save Workspace

Saving a workspace saves the current workspace. Workspaces may be automatically saved on a close by setting this under *Configure>Settings*, **Workspace** tab.

Save Workspace As

Opens the Save As dialog (see **Section 14.11 "File Management Dialog"**), allowing you to rename/relocate the current workspace before saving.

Close Workspace

Closing a workspace returns you to the default startup workspace configuration.

Import

Import a debug or hex file into your MPLAB IDE project. (See **Section 14.17 "Import Dialog"**.)

Export

Export a Hex file from your MPLAB IDE project. (See **Section 14.9 "Export Hex File Dialog"**.)

Print

Prints the active edit window. The Print dialog will open to allow you to set the printer and print options.

Recent Files

Displays a list of files opened during the current MPLAB IDE session. Set the amount of displayed files under *Configure>Settings*, **Workspace** tab.

Recent Workspaces

Displays a list of workspaces opened during the current MPLAB IDE session. Set the amount of displayed workspaces under *Configure>Settings*, **Workspace** tab.

Exit

Prompts you to save the file(s)/workspace, if opened, and then closes the MPLAB IDE application.

12.2.2 Edit

Below are the menu items in the Edit menu. For more on the editor and the items shown here, see **Chapter 16. "Using the Editor"**.

Undo

Reverses the last change made to the active window. When there is no edit action to undo, the menu command is grayed out and you cannot select the command.

Redo

Reverses the effect of the most recent Undo operation in the active window. When there is no edit action to redo, the menu command is grayed out and you cannot select the command.

Cut

Deletes the selected text in the current window and places it on the clipboard. After this operation you can paste the deleted text into another MPLAB Editor window, into a different location in the same MPLAB Editor window, or into another Windows application.

Copy

Copies the selected text in the current window onto the clipboard. After this operation, you can paste the copied text into another MPLAB Editor window, into another location in the same MPLAB Editor window, or into another Windows application.

Paste

Pastes the contents of the clipboard into the current window at the insertion point. You can only perform this operation if the clipboard contains data in text format. **MPLAB Editor does not support pasting of bitmaps or other clipboard formats**.

Delete

Deletes the selected text.

Select All

Selects all text in the Edit window.

Find

Opens the Find dialog. (See **Section 14.14 "Find and Replace Dialogs"**.)

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Find In Files

Opens the "Find in Files" dialog. (See **Section 14.13 "Find In Files Dialog"**.)

Replace

Opens the Replace dialog. (See **Section 14.14 "Find and Replace Dialogs"**.)

Go to

Go to the specified item of the Go To dialog. (See **Section 14.15 "Go To Dialog"**.)

GoTo Locator

Go to where the selected C code symbol is defined. For example, right click on a function (in the Project window, Symbol tab, or in code in a File window) and select "GoTo Locator" to go to the line in code where this function is declared.

To use this option, "Enable Tag Locators" must be selected. Select "Properties" and in the Editor Options dialog, **General** tab, click on "Enable Tag Locators".

Go Backward

Based on the history of the MPLAB Editor "GoTo" and "GoTo Locator", move backward through this history.

Go Forward

Based on the history of the MPLAB Editor "GoTo" and "GoTo Locator", move forward through this history.

External DIFF

MPLAB IDE does not provide a file compare or difference (DIFF) utility. However, you may install a utility of your choosing and configure it to work with MPLAB IDE using the dialog(s) from this command. For more on this function, see the MPLAB Editor on-line help file.

Advanced

Set advanced text features. These include:

- Making selected text all uppercase or all lowercase.
- Making selected text a comment or not a comment.
- Formatting a block comment. See MPLAB Editor help for more information.
- Indenting or outdenting text.
- Matching if a brace, bracket or parenthesis. Go to the brace that is the matching brace for the brace at the cursor. This function works for curly braces, parentheses, angle brackets and square brackets.

Bookmarks

Work with bookmarks. Toggle a bookmark (alternately enable/disable a bookmark), go to the next or previous bookmark or disable all bookmarks.

Properties

Opens the Editor Options dialog (see **Section 16.2.1 "Editor Options Dialog"**).

12.2.3 View

Below are references to items in the View menu. Any item not applicable to the selected device will be disabled. Selecting an item in this menu will make that item visible on the desktop.

- **Section 12.3 “Toolbars”**
- **Chapter 13. “Windows”**

12.2.4 Project

Below are the menu items in the Project menu. For more on projects, see **Chapter 6. “Projects and Workspaces”**.

Project Wizard

Use the Project Wizard to help you set up a new project. For more on the wizard, see **Section 6.2 “Using the Project Wizard”**.

New

Create a new project in the workspace.

Opens the New Project dialog. You will be asked to enter a name and path for the new project. (See **Section 14.20 “New Project Dialog”**.)

Open

Add an existing project to the workspace and set as active. Opens the Open Project dialog (see **Section 14.11 “File Management Dialog”**).

Close

Close the current project in the workspace. You will be prompted to save the current project before closing.

Closing a project does not close the workspace. Use *Edit>Workspace>Close* to close the workspace.

Set Active Project

Select a project as the active project in the workspace. For more on active projects, see **Section 6.9 “Using Multiple Projects in a Single Workspace”**.

To enable Quickbuild, select None.

Quickbuild (*filename*)

Build a single assembly file using MPASM assembler without having to create a project (no linker). None/Enable Quickbuild must be selected under Set Active Project. The assembly file must be open in a file window and the window must be active.

There may be assembler limitations with this procedure. See **Section 11.5 “Limitations”**.

Package in .zip

Package the current project into a `.zip` file. While MPLAB IDE has the ability to zip files up, it cannot unzip them. So a separate program will be required to unzip the project.

Clean

Removes all intermediary project files, such as object, hex and debug files, for the active project. These files are recreated from other files when a project is built.

Locate Headers

Locates all the header files (`.h`) called out in the project files and presents them in checklist window so they may be added to the project.

Although header files don't need to be added to your project in order for that project to build, there are several reasons why you might want to add them to the Project window under "Header Files":

- The files can be opened from the Project window for editing instead of hunting for them on your computer.
- The "Save Project As" feature works better
- The "Package Project as .zip" feature works better

The project must be a C code project.

Export Makefile

Under *Project>Build Options>Project*, **Directories** tab, you must have selected "Assemble/Compile/Link in the project directory" under "Build Directory Policy" for this feature to work.

Export to a file the instructions to Make the project. Use this file to Make your project outside of MPLAB IDE, i.e., with a `make`.

Build All

Build the project by compiling/assembling all files. A project must be open before this item will appear.

Make

Build the project by compiling/assembling only files that have changed since the last build. A project must be open before this item will appear.

Build Configuration

Select the type of build, either debug or release (see **Section 8.3 "Build Configuration (Debug/Release)"**).

Build Options

Set and view options for the active project (Project) or individual files using the Build Options dialog (see **Section 14.5 "Build Options Dialog"**).

Save Project

Save the active project.

Save Project As

Opens the Save Project As dialog. (See **Section 14.24 "Save Project As Dialog"**.)

Add Files to Project

Insert files into the active project. (See **Section 14.11 "File Management Dialog"**.) Depending on the type of file, MPLAB IDE will sort the files into the correct type within the project window tree.

Add New File to Project

Requests a name for the new file in a dialog (see **Section 14.11 "File Management Dialog"**). On **Save**, the file is added to the project and a new file window is opened.

Remove File from Project

Remove (delete) file from active project. The file is not deleted from the directory.

Select Language Toolsuite

Select the toolsuite you will use for your project, e.g., Microchip Toolsuite. (See **Section 14.26 "Select Language Toolsuite Dialog"**.)

Set Language Tool Locations

Set the paths/directories to the language tools you will be using in your project, i.e., match a language tool name in MPLAB IDE (ex: MPASM assembler) with an executable (ex: `C:\Program Files\Microchip\MPASM Suite\mpasmwin.exe`). (See **Section 14.27 "Set Language Tool Location Dialog"**.)

Version Control

Set up your project to use files from a version control system (e.g., Visual Source Safe.) For more information, see **Section 6.6 "Using A Version Control System (VCS)"**.

12.2.5 Debugger

Below are the menu items in the Debugger menu.

Note:	Single stepping may be very slow when using a debugger if your selected device has EEPROM data and (1) you have a programmer enabled or (2) you have the EEPROM window open, either of which will attempt to access the data on each step. To improve speed, disable the programmer or close/minimize the EEPROM window.

Select Tool

Select a debug tool. The default is None. The list of available debuggers will depend on which ones you have installed. The order of items on the list is subject to installation order.

Clear Memory

Clear all or only certain types of MPLAB IDE memory used in the project, e.g., program, data, EEPROM, configuration.

Once a tool is selected, you may see:

Section 12.2.5.1 "Basic Debug Options" (See below)

Section 12.2.5.2 "Tool-Specific Options"

12.2.5.1 BASIC DEBUG OPTIONS

Once you have selected any debug tool, the Debugger menu will add the following options:

Run

Execute program code until a breakpoint is encountered or until Halt is selected.

Execution starts at the current program counter (as displayed in the status bar). The current program counter location is also represented as a pointer in the Program Memory window. While the program is running, several other functions are disabled.

Animate

Animate causes the debugger to actually execute single steps while running, updating the values of the registers as it runs.

Animate runs slower than the Run function, but allows you to view changing register values in the Special Function Register window or in the Watch window.

To Halt Animate, use the menu option *Debugger>Halt* instead of the toolbar Halt or **F5**.

Halt

Halt (stop) the execution of program code. When you click Halt, status information is updated.

Step Into

Single step through program code.

For assembly code, this command executes one instruction (single or multiple cycle instructions) and then halts. After execution of one instruction, all the windows are updated.

For C code, this command executes one line of C code, which may mean the execution of one or more assembly instruction, and then halts. After execution, all the windows are updated.

Note:	Do not step into a `SLEEP` instruction.

Step Over

Execute the instruction at the current program counter location. At a `CALL` instruction, Step Over executes the called subroutine and halts at the address following the `CALL`. If the Step Over is too long or appears to "hang", click Halt.

See also **Section 8.4.4.2 "MPLAB IDE Features Impacted When All Breakpoints are Used"**.

Step Out

Step out of a subroutine. If you are single stepping through subroutine code, you may finish executing the rest of the subroutine code and halt at the address following the subroutine `CALL` by using Step Out.

Reset

Issue the specified Reset, either $\overline{\text{MCLR}}$, Watchdog Timer, Brown-Out or Processor Reset. Reset options and actions depend on the device and tool selected.

Breakpoints

Open the Breakpoint dialog. Set multiple breakpoints in this dialog. For other ways to set a breakpoint, see **Section 5.18 "Using Breakpoints"**.

Settings

Open a tool-specific settings dialog. Set up tool functions here. Also, find tool limitations.

12.2.5.2 TOOL-SPECIFIC OPTIONS

Depending on what debug tool you have selected (*Debugger>Select Tool*), additional tool-specific items, such as "Stopwatch", may appear on this menu between "Breakpoints" and "Settings".

12.2.6 Programmer

Below are the menu items in the Programmer menu.

Select Programmer

Select a programmer. The default is None. The list of available programmers will depend on which ones you have installed. The order of items on the list is subject to installation order.

Once a programmer is selected, you may see:

Section 12.2.6.1 "Basic Programmer Options" (See below.)

Section 12.2.6.2 "Programmer-Specific Options"

12.2.6.1 BASIC PROGRAMMER OPTIONS

Depending on the programmer chosen, different options may appear on the Programmer menu. Basic items that will generally be available are:

Enable Programmer

Establish communications between MPLAB IDE and the programmer. This is grayed out if the programmer is already enabled.

Disable Programmer

End communications between MPLAB IDE and the programmer. This is grayed out if the programmer is already disabled.

Program

Program specified memory areas: program memory, Configuration bits, ID locations and/or EEPROM data.

Verify

Verify programming of specified memory areas: program memory, Configuration bits, ID locations and/or EEPROM data.

Read

Read specified memory areas: program memory, Configuration bits, ID locations and/or EEPROM data.

Blank Check All

Check to see that all device memory is erased/blank.

Blank Check OTP

For One-Time-Programmable (OTP) devices, check to see that program, data and EEPROM memory is erased/blank.

Erase Flash Device

Erase all data on the PIC Flash MCU device including memory, ID and Configuration bits.

Reset Program Statistics

Set programming statistics (e.g., errors) to default values.

Download OS

Download the latest operating system for your programmer.

Note:	PICSTART Plus must be upgraded before this feature is usable. Follow the instructions in the help for this tool.

About

View information about your tool in this dialog.

Settings

Opens a tool-specific settings dialog. Set up information about your tool in this dialog, i.e., Memory Ranges and Communications Port Setup, as well as Voltages and SQTP, if applicable.

12.2.6.2 PROGRAMMER-SPECIFIC OPTIONS

Depending on what programmer you have selected (*Programmer>Select Programmer*), additional programmer-specific items, such as "Load SQTP File", may appear on this menu.

12.2.7 Tools

Below are the alphabetically-listed menu items in the Tool menu. Your tool order will vary.

AN901 BLDC Tuning Interface

Use this interface in support of application note AN901 (DS00901). See the separate help file for this interface (*Help>Topics*, "Tools").

AN908 ACIM Tuning Interface

Use this interface in support of application note AN908 (DS00908). See the separate help file for this interface (*Help>Topics*, "Tools").

Data Monitor and Control Interface

Use this interface in support of various applications. See the separate help file for this interface (*Help>Topics*, "Tools").

Gimpel PC-LINT/MISRA

Select this to create a Gimpel PC-LINT/MISRA menu in MPLAB IDE. See the separate help file for using these tools (*Help>Topics*, "Tools").

KeeLoq Plugin

Create an SQTP file for the selected KeeLoq security IC device (*Configure>Select Device*). See programmer help for information on how to program an SQTP file into a device. Full programmer support coming later.

Matlab/Simulink

Select this to create a MATLAB/Simulink menu in MPLAB IDE. See the separate help file for using these tools (*Help>Topics*, "Tools").

MPLAB Macros

Enable Microsoft macro capability for use with MPLAB IDE. See **Chapter 10. "MPLAB Macros"**.

Memory Starter Kit

Program Microchip memory devices (serial EEPROMs) selected in *Configure>Select Device*.

RTOS Viewer

If you have a Real-Time Operating System (RTOS) installed and included in your project, you can open the viewer. See **Section 13.21 "RTOS Viewer Window"**.

12.2.8 Configure

Below are the menu items in the Configure menu.

Note: Not all items may be available depending on device and debug tool selected.

Select Device

Select the device for your development mode. (See **Section 14.25 “Select Device Dialog”**.) Select the development tool under the Debugger or Programmer menu.

Configuration Bits

Select values for the device Configuration bits. (See **Section 13.5 “Configuration Bits Window”**.) Setting these values will affect both debugger and programmer operation.

External Memory

Select whether to use external memory or not. Also specify external memory range. (See **Section 14.10 “External Memory Setting Dialog”**.)

ID Memory

Enter value into ID memory. (See **Section 14.30 “User ID Memory Dialog”**.)

Settings

Enter default setting for the workspace, debugger, program loading, hot keys and projects. (See **Section 14.28 “Settings Dialog”**.)

12.2.9 Window

Below are the menu items in the Window menu.

Close All

Close all open windows.

Cascade

Arrange open windows to overlap so that each title bar is visible.

Tile Horizontally

Arrange open windows in smaller sizes to fit next to each other horizontally.

Tile Vertically

Arrange open windows in smaller sizes to fit next to each other vertically.

Arrange Icons

Arrange all iconized windows on the bottom of the IDE.

Window Sets

Select a predefined window set. Create a set with “Create Window Set”. Delete a set with “Destroy Window Set”.

Create Window Set

Save the current positions of open windows and/or toolbars to a file. Use the new window set as a template to set up the layout of other projects.

MPLAB IDE windows and toolbars that are not open will be opened and positioned according to the selected window set. Tool-specific (including MPLAB Editor) windows and toolbars that are not open will not be opened. However, if the window or toolbar is already opened when the set is applied, it will be positioned.

Destroy Window Set

Remove a window set from the list.

Open windows

A list of all open windows is displayed. Click on a window name to make that window active.

More windows

If the list of all open windows is too long for the menu display, clicking this item will open a dialog that lists additional open windows. Click on a window name in the dialog to make that window active.

Note:	If the path name is too long to be shown in the scroll list and no horizontal scroll bar is visible, try resizing the Path column header to make the scroll bar appear.

12.2.10 Help

Below are the menu items in the Help menu.

Topics

Select a help file from the list on the dialog. (See **Section 14.16 "Help Topics Dialog"**.)

Release Notes

View a list of all Readme files available for Microchip tools. Select a file and then click on the button to see the full HTML file in a browser window.

Driver Installation

View a list of all device driver installation instruction files available for Microchip tools. Select a file and then click on the button to see the full HTML file in a browser window.

Check for Updates

Opens a dialog to check to manually or automatically check for updates to MPLAB IDE. See **Section 14.6 "Check for Updates Dialog"**.

Web Links

Find Microchip tools and support via the web.

About MPLAB IDE

Review MPLAB IDE trademarking and component version information. (See **Section 14.2 "About MPLAB IDE Dialog"**.)

12.3 TOOLBARS

MPLAB IDE displays different toolbars depending on which features or tools you are using. The icons in these toolbars provide shortcuts to routine tasks.

Toolbar buttons may be grayed out for various reasons. See **Section 12.5 “Grayed out or Missing Items and Buttons”**.

Toolbars Features

- Click and drag the toolbar to make it a floating toolbar.
- Click and drag the toolbar to the top or sides of MPLAB IDE desktop to dock it.
- Click and drag the toolbar off the MPLAB IDE desktop.
- Hover the mouse pointer over an icon to pop up the icon name.
- Right click on a toolbar to change the contents or show/hide toolbar.

Toolbars Available

- Standard Toolbar
- Project Manager Toolbar
- Debug Toolbar
- Device Debug Resource Toolbar
- Programmer Toolbar
- Checksum Toolbar

12.3.1 Standard Toolbar

The Standard (Edit) Toolbar currently contains button icons for the following functions:

- New File – Open a new file window
- Open File – Open an existing file in a window
- Save File – Save the current file window contents to a file
- Cut – Cut selected text to clipboard
- Copy – Copy selected text to clipboard
- Paste – Paste text from clipboard
- Print File – Print contents of active file window
- Find – Open the Find dialog for finding text in active file window
- Find in Files – Find text in multiple files
- Go Backward – Based on the history of the MPLAB Editor “GoTo” and “GoTo Locator”, move backward through this history.
- Go Forward – Based on the history of the MPLAB Editor “GoTo” and “GoTo Locator”, move forward through this history.
- Help – Displays MPLAB IDE Help selection dialog

12.3.2 Project Manager Toolbar

The Project Manager Toolbar currently contains button icons for the following functions:

- Build Configuration drop-down list – See **Section 8.3 "Build Configuration (Debug/Release)"** for more on this listbox.
- New Project – Set the name and location of a new project.
- Open Project – Open an existing project.
- Save Workspace – Save the current project and workspace to files.
- Build Options – View or change project settings.
- Package Project – Zip up files in project.
- Toolsuite Info – Pop up more information on your selected language toolsuite.

If a project is loaded, additional items may be found:

- Make – Build only the files in the active project that have changed.
- Build All – Build all files in the active project.
- Export Makefile – Export instructions to make project to a file.
- Locate Headers – Locate and display all included headers for addition to project.

12.3.3 Debug Toolbar

The Debug Toolbar currently contains button icons for the following functions:

- Run – Run program
- Halt – Halt program execution
- Animate – Continually step into instructions – Halt using *Debugger>Halt*
- Step Into – Step into next instruction
- Step Over – Step over next instruction
- Step Out – Step out of subroutine
- Reset – Perform $\overline{\text{MCLR}}$ Reset
- Breakpoints – Open the breakpoint dialog

Depending on the debug tool chosen, other icons may appear.

12.3.4 Device Debug Resource Toolbar

The Device Debug Resource Toolbar (DDRT) is available only when a debug tool is selected. The DDRT can be enabled/disabled from the *View>Toolbars* menu. Each toolbar element can be shown/hidden with the Customize Toolbar functionality. Also, toolbar labels can be shown/hidden to reduce toolbar size.

- HW BP – Hardware Breakpoints Supported
- Used – Hardware Breakpoints Used
-  – Breakpoint Details (see below)
- SW BP – SW BP Status

Breakpoint Details Window

This window will give details on breakpoint usage.

- Usage – The number of breakpoints of specified type currently in use.
- Breakpoint Type – The type of breakpoint, such as program memory, data memory, or data capture.

12.3.5 Programmer Toolbar

Depending on the programmer chosen, different button icons may appear on the Program Toolbar. Basic icons that will generally be available are:

- Blank Check All/Blank Check – Check that the device memory is blank.
- Read – Read device memory as specified in *Programmer>Settings*, **Program** tab.
- Program – Program device memory as specified in *Programmer>Settings*, **Program** tab.
- Verify – Verify that target memory has been correctly programmed.
- Erase Flash Device – If the device is Flash, erase the device.
- Program Statistics – Display programing statistics, such as how many times programming passed, failed and the total number of attempts to program.

12.3.6 Checksum Toolbar

This toolbar only displays the checksum value or N/A (see below). Checksum algorithms are described in the device's programming specification, found on our web site.

When a device is code-protected, you may use the unprotected checksum to determine the device checksum.

An “N/A” means that the checksum has not been specified. On some devices, the configuration bytes and ID locations are implemented as volatile, not persistent, memory, which means that configuration data must be programmed each time the device is powered up. Therefore, there is no checksum calculation due to the lack of readable locations on a code-protected device.

Previously, checksum had been on the status bar. Its place has been taken by banking information.

12.4 STATUS BAR

The status bar provides up-to-date information on the status of your MPLAB IDE session.

When an application is running, it displays "Running" and a progress bar. When an application is not running, the information provided includes that shown in Table 12-1.

Status bar items may not be shown for various reasons. See **Section 12.5 "Grayed out or Missing Items and Buttons"**.

TABLE 12-1: STATUS BAR ITEMS

Item	Title	Typical Entry	Description
1	Current debug tool	MPLAB® SIM	Displays currently selected debug tool. Use the Debugger menu to choose a different tool.
2	Current programmer	PICSTART® Plus	Displays currently selected programmer. Use the Programmer menu to choose a different tool.
3	Current processor	PIC18F452	Displays the currently selected processor. Use *Configure>Select Device* to change the device.
4	Current program counter	pc:0x5f	Displays the current program counter. Double click to enter a new PC value.
5	Current w register value	W:0x00	Displays current w register value.
6	Status bits	ov Z dc c	Upper Case = Set ('1') Lower Case = Reset ('0')
7	Global break enable	Bk On	Displays current status of Global Break Enable.
8	Processor frequency	4MHz	Displays current processor frequency.
9	Banking information	bank 0	Displays current bank in data memory.
10*	Line No., Column-Windows Open	Ln 1 Col 1	Displays current line number and column in file.
11*	Insert/strikeover	INS	Toggles typing mode between insert and strikeover: INS = Insert characters OVR = Type over characters Use <Insert> to change mode.
12*	Write/read only	WR	Displays Write/Read Only Status: WR = Editable File RO = Read Only File

* Only available when a file (editor) window has focus.

12.5 GRAYED OUT OR MISSING ITEMS AND BUTTONS

There are several reasons why a menu item, toolbar button or status bar item may be grayed out (unavailable) or missing:

- The item/button is related to a device feature that the selected device does not have, e.g., the PIC16F877A does not support external memory (*Configure>External Memory* is grayed out.)
- The item/button is related to a tool feature that the selected tool does not have, e.g., “Step Out” is not available on MPLAB ICD 2, so the item is grayed out on the Debugger menu and toolbar.
- The item/button is project-related and no project has been selected, e.g., *Project>Build Options* will not be available (No Active Project).
- The item/button is not supported for the selected device or tool.
- The item/button is performing its function and so cannot be selected again, e.g., “Run” is grayed out when the program is running.
- The item/button is mutually exclusive to another item, e.g., “Halt” is available when the program is running while “Run” is grayed out, and “Run” is available when the program is halted while “Halt” is grayed out.

NOTES:

Chapter 13. Windows

13.1 INTRODUCTION

MPLAB IDE windows behave as normal Windows applications. Other standard Windows features include window sizing buttons and vertical and horizontal scroll bars. Additional MPLAB IDE window features are:

- In-place editing of data
- Dockable windows
- Click-and-drag to change window column width and order
- Right click (context) menu to change window font and color

Most windows available in MPLAB IDE are listed under the View menu.

General Window Information
Changing Window Data and Properties
Code Display Window Symbols

Specific Windows*	**Related Menu**
Call Stack Window	View
Configuration Bits Window	Configure
CPU Registers Window (PIC32MX Devices Only)	View
Disassembly Listing Window	View
EEPROM Window	View
File (Editor) Window	File
File Registers Window	View
Flash Data Window	View
Hardware Stack Window	View
LCD Pixel Window	View
Locals Window	View
Logic Analyzer Window	View
Memory Window (PIC32MX Devices Only)	View
Memory Usage Gauge	View
Output Window	View
Program Memory Window	View
Project Window	View
RTOS Viewer Window	Tools
SFR/Peripherals Window (PIC32MX Devices Only)	View
Special Function Registers Window	View
Trace Memory Window	View
Watch Window	View
* Depending on the tools you have installed, tool-specific windows may be available for viewing. See the documentation for that tool for information about that window.	

13.2 CHANGING WINDOW DATA AND PROPERTIES

MPLAB IDE windows have some or all of the listed properties, depending on the type and use of the window.

13.2.1 Window Data Updates on Halt

Open windows are updated on a program halt. This includes a halt after a run and stepping. Halt updates can have the following effects:

- Speed – Updating takes time. To decrease update time, close any unused windows.
- Data overwrites – The value of a file register displayed in an open window is read on halt. See you device data sheet for register operation on read.

13.2.2 Changing Window Data – Edit-in-Place or Choose from List

MPLAB IDE window data may be edited as described below. If you cannot edit the data, then this information is not available for you to change.

- Data may be edited "in place", i.e., double click to select an item and then type in a new value.
- Date may be chosen from a drop-down list when only certain choices are possible.

13.2.3 Changing Window Location – Dock/Undock

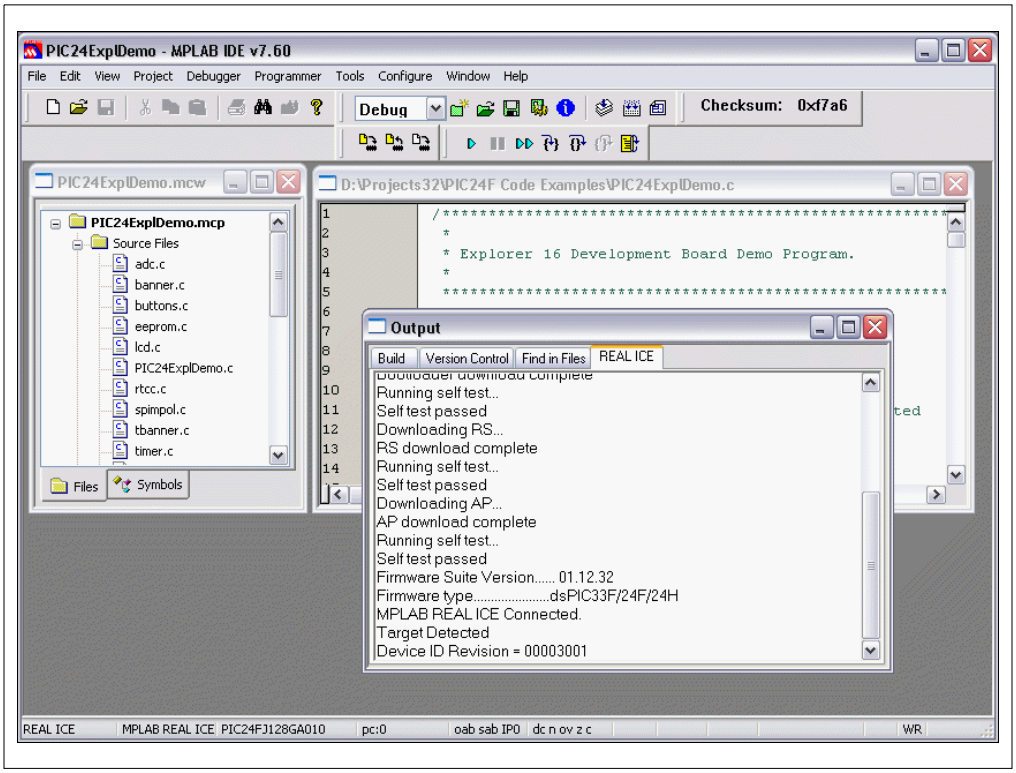
Windows may be docked to or undocked from the MPLAB IDE desktop.

Note:	File (editor) windows may not be made dockable.

13.2.3.1 UNDOCKED WINDOWS – DEFAULT CONFIGURATION

When you use MPLAB IDE, all windows are set to be undockable by default. This means you may move windows around only on the MPLAB IDE desktop, and the active window will be displayed over other inactive windows.

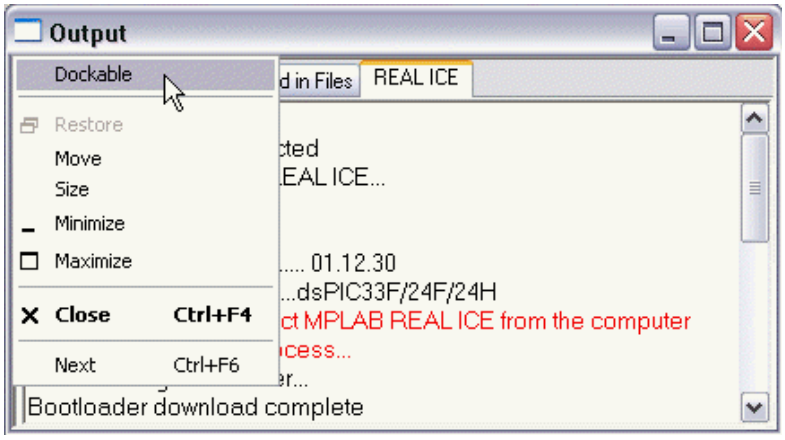
FIGURE 13-1: UNDOCKED WINDOWS – DEFAULT



13.2.3.2 MAKING WINDOWS DOCKABLE

Windows may be made dockable by clicking on the window system menu (located in the upper left-hand corner of the window) and selecting “Dockable” from the drop-down menu. (This will select the “Dockable” feature.)

FIGURE 13-2: MAKE WINDOW DOCKABLE



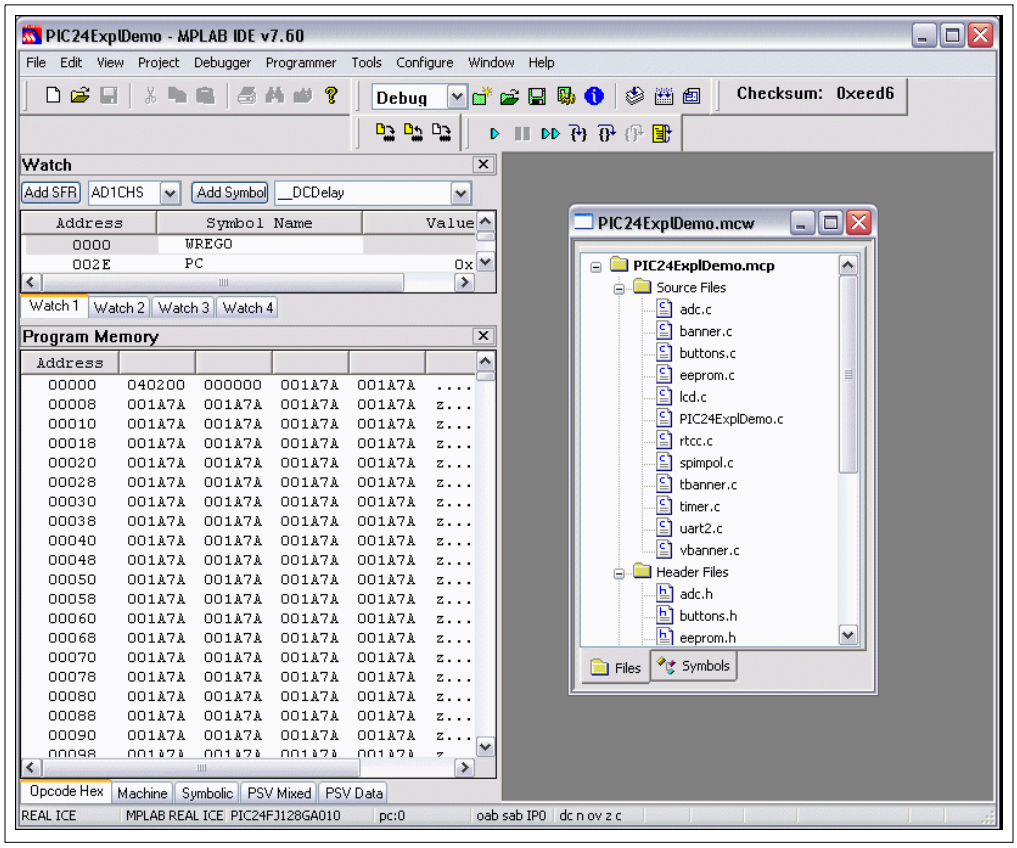
13.2.3.3 DOCKING AND UNDOCKING DOCKABLE WINDOWS

Dockable windows may be docked to the top, bottom or sides of the MPLAB IDE desktop by clicking on their title bar and dragging them to the desired docking location. They may be undocked by clicking on their title bar and dragging them to the middle of the desktop.

Dockable windows can share the same space as a window already docked by doing the following:

1. Grab the title bar of the window you want to add to another docked window.
2. Hover this window over the other. Gray lines will appear showing you where the window will be placed on the docked window.
3. Release the window when you have decided where you want it placed on the docked window.

FIGURE 13-3: DOCKED WINDOWS



The result will be two windows in the space the first previously occupied.

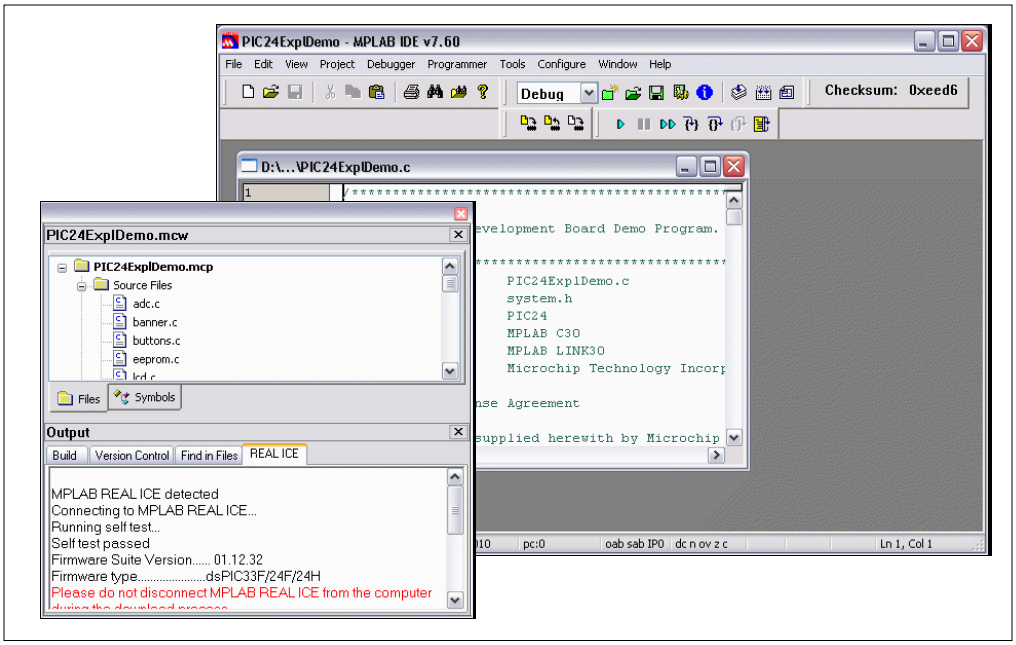
- To resize a window within this space, click and drag its edges when the resize arrows appear.
- To move a window from this space, grab its title bar and move it out of the floating window space.

13.2.3.4 FLOATING DOCKABLE WINDOWS

Dockable windows may be floated, i.e., dragged off the MPLAB IDE desktop. Floating windows can be independent or combined into the space of another floating window. To do this:

1. Grab the title bar of the window you want to add to another floating window.
2. Hover this window over the other. Gray lines will appear showing you where the window will be placed on the floating window.
3. Release the window when you have decided where you want it placed on the floating window.

FIGURE 13-4: FLOATING WINDOWS



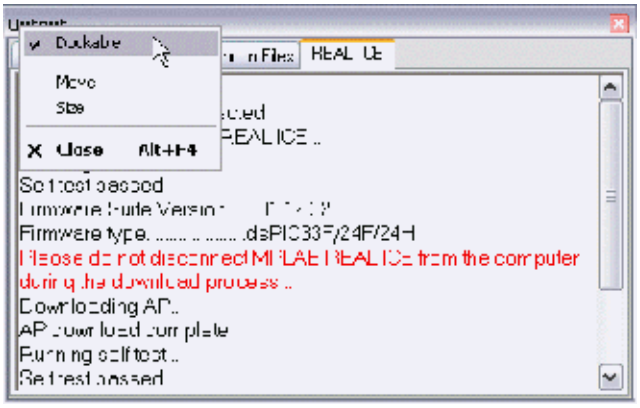
The resulting floating window will have a blank top title bar and title bar for each window within the floating window space.

- To resize a window within this space, click and drag its edges when the resize arrows appear.
- To move a window from this space, grab its title bar and move it out of the floating window space.

13.2.3.5 MAKING WINDOWS UNDOCKABLE

Windows may be made undockable by right clicking on the window's title bar and selecting "Dockable" from the drop-down menu. (This will uncheck "Dockable".)

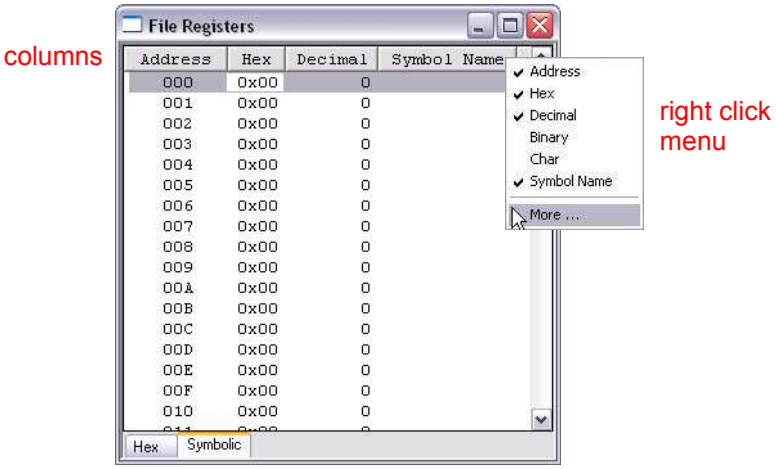
FIGURE 13-5: MAKE WINDOW UNDOCKABLE



13.2.4 Changing Window Columns – Size, Visibility, Order

Some windows contain data in columns.

FIGURE 13-6: FILE REGISTER WINDOW WITH COLUMNS



Columns in windows may be changed as follows:

- Resize Columns
- Make Columns Visible/Invisible
- Order Columns

13.2.4.1 RESIZE COLUMNS

Columns can be resized by moving the cursor over the line between columns until it changes to the resize cursor and then clicking and dragging to widen or narrow the column.

13.2.4.2 MAKE COLUMNS VISIBLE/INVISIBLE

Columns can be made visible/invisible as follows:

- Right click on any column head and check an item from the list to make it visible.
- Right click on any column head and select More to open the window Properties dialog, **Column Settings** tab, (see **Section 14.23 "Properties Dialog"**) where you may set column visibility.
- Right click in the window and select Properties. On the **Column Settings** tab, you may set column visibility.

13.2.4.3 ORDER COLUMNS

Columns can be ordered as follows:

- Click on any column head and drag it to the desired location.
- Right click on any column head and select More to open the window Properties dialog, **Column Settings** tab, (see **Section 14.23 "Properties Dialog"**) where you may set column location.
- Right click in the window and select Properties. On the **Column Settings** tab, you may set column location.

13.2.5 Changing Window Fonts and Colors

- Window fonts and change colors may be specified by right clicking in the window and selecting Properties, **General** tab (see **Section 14.23 "Properties Dialog"**).

13.3 CODE DISPLAY WINDOW SYMBOLS

In some windows, symbols may appear. Their meaning is shown in the table below.

TABLE 13-1: WINDOW SYMBOLS

Symbol	Description
	Current location (line of code) of the program counter. Program execution is halted (stopped).
	Location of the program counter before the program was run. Program is currently executing (running).
	Location specified by a Go To command.
	Breakpoint set.
	Breakpoint disabled.
	Breakpoint set on an address. This symbol is displayed only in the file (editor) window, since one line of code could correspond to several addresses. See **Section 5.18 "Using Breakpoints"** for more information.
	Program memory address. This symbol is displayed in the Address column of the Watch window for variables defined in program memory.

TABLE 13-1: WINDOW SYMBOLS (CONTINUED)

Symbol	Description
U	Large letters in "Add Files" window of Project Wizard. For meanings, see **Section 6.2.5 "Project Wizard – Add Files"**.
✔	Code coverage enabled for this location (Program Memory window, MPLAB® ICE 2000/4000 only).
	Complex trigger point set at this location (Program Memory window, MPLAB ICE 2000/4000 only).
	Filter in trace (File window). See **Section 16.6.4 "Filter Trace"** for more information.
	Filter out trace (File window). See **Section 16.6.4 "Filter Trace"** for more information.
	Set/remove data capture or runtime watches (Watch window.) See **Section 13.25 "Watch Window"**.
	Bookmarked line.

13.4 CALL STACK WINDOW

For 16- and 32-bit devices, a call stack window is available to view `call`s and `goto`s in executing C code. This window is not applicable for assembly code.

> **Note:** If the program stops on the declaration of the function, the display will not show the calling function.

It is recommended that code optimization be turned off when using the call stack.

13.5 CONFIGURATION BITS WINDOW

The Configuration Bits window displays information on Configuration bit settings. MPLAB IDE recognizes Configuration bits set in code as well as in the Configuration Bits window. Any Configuration bit whose value is not set in code or the window defaults to '`1`'.

Configuration bit settings made in code are reflected in the window upon loading a program (i.e., build, import or open a project). Configuration bit settings made in the window are reflected immediately. MPLAB IDE uses the current values of the window during debugger execution.

> **Note:** For PIC32MX MCUs, some configuration registers are too large (have too many bits) to be displayed in the configuration window. These registers can only be set in code.

If "Clear program memory upon loading a program" is checked in *Configure>Settings*, **Program Loading** tab, Configuration bits are cleared and then code changes are loaded upon loading a program. If no Configuration bit settings are specified in code, the bits are simply cleared.

When you close a workspace, MPLAB IDE saves the latest Configuration bit information in the workspace file. When you open a workspace, MPLAB IDE clears memory, loads code setups of Configuration bits and then loads the last Configuration Bits window information saved in the workspace.

For more information, see **Section 9.2 "Configuration Bits"**.

- Configuration Bits Window Display
- Configuration Bits Window Menu
- Configuration Bits Window FAQ

13.5.1 Configuration Bits Window Display

Selecting *Configure>Configuration Bits* opens the Configuration Bits window. Use this window to set Configuration bit values for your selected device.

> **Note:** These values are obtained when building a source file and are used to control the operation of the debug tool.

Configuration Bits set in code

When this checkbox is checked (default setting), configuration bit values are taken from code, e.g.,

```
;include file with config bit definitions
   #include p16f877a.inc
;Set oscillator to HS, watchdog timer off, low-volt prog. off
   __config _HS_OSC & _WDT_OFF & _LVP_OFF
```

When this checkbox is unchecked, code values will be overridden by values in the columnar display portion of this window until you rebuild the project.

TABLE 13-2: CONFIGURATION BITS COLUMNAR DISPLAY

Column Head	Definition
Address	The address of the configuration word/byte.
Value	The current value of the configuration word/byte.
Category	The name of the Configuration bit in the corresponding configuration word/byte.
Setting	The current setting of the Configuration bit. Use the drop-down list to change the setting. The Value of the configuration word/byte will change accordingly.

13.5.2 Configuration Bits Window Menu

Below are the menu items in the Configuration Bits right mouse button menu. Click on an item to expand or collapse the description.

Close

Close this window.

Reset to Defaults

Restore the window defaults. This would correspond to the power-up state of the Configuration bits on the actual device.

Output to File

Output the Configuration Bits for each address (i.e., configuration words) shown in the Configuration window to a text file at a selected location.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the Configuration Bits window.

Properties

Set up window properties. See **Section 14.23 "Properties Dialog"**.

13.5.3 Configuration Bits Window FAQ

How do I:

Clear Configuration bits upon loading a program?

Select *Configure>Settings* and click on the **Program Loading** tab. Check the box for "Clear Configuration bits upon program loading".

Develop with the Configuration Bits window?

To use only the Configuration Bits window to develop your application:

- Do not set any Configuration bits in code, i.e., do not use initialization data.
- *Configure>Settings*, **Program Loading** tab, "Clear program memory upon loading a program" should be unchecked.

  Once you have completed development, you will then have to copy your Configuration Bits window settings into initialization data.

Develop with Configuration bits set in code?

To use only Configuration bits in code, i.e., initialization data, to develop your application:

- Do not change any Configuration bit settings in the Configuration Bits window.
- *Configure>Settings*, **Program Loading** tab, "Clear program memory upon loading a program" should be checked.

13.6 CPU REGISTERS WINDOW (PIC32MX DEVICES ONLY)

The CPU Registers window displays the contents of the Special Function Registers (SFRs) that relate to the device CPU. To view only a few CPU SFR's, you may prefer to use a Watch window, which may help with speed issues when using hardware debug tools (i.e., faster window update rate.)

Whenever a break occurs, the contents of the CPU Registers are updated.

Visible Registers

If a data memory register is not physically implemented on a device, it may not appear in the SFR list. Some tools, such as simulators, may allow you to see registers that do not exist on the actual device, such as prescalers.

Single Stepping

If "Freeze Peripherals On Halt" is selected, the I/O port bits in the SFR or the Watch windows will not update when single stepping. The pin will be modified, but the read request to retrieve the new value is blocked by the freeze and cannot be updated until the next step or run command.

- CPU Register Window Display
- CPU Register Window Menu
- Changing Window Data and Properties
- CPU Register Window FAQ

13.6.1 CPU Register Window Display

Data is displayed in the following columns.

- Address – SFR physical hexadecimal address.
- Name – Symbolic name for the SFR.
- Hex – Hexadecimal value, shown in 4-byte blocks.
- Decimal – Value in decimal format.
- Binary – Value in binary format.
- Char – Value in character format.
- Virtual – SFR virtual hexadecimal address as defined by the Bus Matrix.

13.6.2 CPU Register Window Menu

Below are the menu items in the Special Function Registers right mouse button menu.

Close

Close this window.

Bitfield Mouseover

Enable/disable bitfield value mouseover. SFR window must be the active window (in focus) for this feature to work.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Goto

Go to a label, symbol or address in the CPU Register window.

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of an "Output Range" and tab-delimited option checkbox. Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the CPU Register window.

Properties

Set up window properties. See **Section 14.23 "Properties Dialog"**.

13.6.3 CPU Register Window FAQ

How do I:

Display different data types?

Right click on any header in the window and select or deselect the desired data types to display in columns. Alternately, right click on any header in the window, select More, and check or uncheck data types to display in columns. Then click **OK**.

13.7 DISASSEMBLY LISTING WINDOW

Select *View>Disassembly Listing* to view disassembled code in this window.

Breakpoints may be set in code. See **Section 8.4 "Breakpoints"**.

Code execution information may be displayed in the form of symbols in the gutter of the window.

If a word of Program Memory does not translate to an instruction or directive, that line will not be displayed in this window.

COD files (output by MPASM assembler v3.2x or before) containing EEData may not work properly. Both the start of the program and EEData appear as Line 0 in the line number table. COFF files (output by MPLINK linker) do not have this issue.

Below are the menu items in the Disassembly window right mouse button menu.

Set/Remove Breakpoint

Set or remove a breakpoint at the currently selected line.

Enable/Disable Break

Enable or disable a breakpoint at the currently selected line.

Breakpoints

Disable, enable or remove all breakpoints.

Run To Cursor

Run the program to the current cursor location. Formerly Run to Here.

See also **Section 8.4.4.2 "MPLAB IDE Features Impacted When All Breakpoints are Used"**.

Set PC at Cursor

Set the Program Counter (PC) to the cursor location.

Center Debug Location

Center the current PC line in the window.

Copy

Copy selected text to clipboard. Select text by (1) clicking and dragging mouse over text or (2) clicking at the beginning of text and shift-clicking at the end of text.

Select All

Select all text in the window.

Output to File

Write the displayed window contents to a text file using a Save As dialog (see **Section 14.11 "File Management Dialog"**).

Print

Print the contents of the window.

Help

Display help on the Disassembly window.

Properties

Open the Disassembly Options dialog. Set display and functional options. This dialog is similar to the Editor Options dialog (see **Section 16.2.1 "Editor Options Dialog"**).

13.8 EEPROM WINDOW

The EEPROM window displays EEPROM data for any microcontroller device that has EEPROM data memory (e.g., PIC16F84A). Data/opcode hex information of the selected device is shown. When an EEPROM register value changes or the processor is halted, the data in the EEPROM window is updated.

> **Note:** The start of EEPROM data memory needs to be specified for use with programmers. For most PIC MCUs, the start should be at 0x2100 (org H'2100'). For PIC18FXXXX devices, the start should be at 0xF00000 (org H'F00000'). Please check the programming specification for your selected device to determine the correct address.

- EEPROM Window Display
- EEPROM Window Menu
- EEPROM Window FAQ

13.8.1 EEPROM Window Display

This display format shows data in the following columns:

- Address – Hexadecimal address of the data in the next column.
- Data Blocks – Hexadecimal data, shown in 1- or 2-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

13.8.2 EEPROM Window Menu

Below are the menu items in the EEPROM right mouse button menu.

Close

Close this window.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Go to the specified item of the Go To dialog. (See **Section 14.15 "Go To Dialog"**.)

Import Table

Open the Import dialog (see **Section 14.11 "File Management Dialog"**).

Export Table

Open the Export dialog (see **Section 14.11 "File Management Dialog"**).

Fill Memory

Fill memory from Start Address to End Address with the value in Data. See **Section 14.12 "Fill Memory/Registers Dialog"**.

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of an "Output Range". Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the EEPROM window.

Properties

Set up window properties. See **Section 14.23.2 "General Tab"**.

13.8.3 EEPROM Window FAQ

How do I:

Fill EEPROM memory with a value?

Right click in the window and select "Fill Registers" to open a dialog where you may enter fill data.

If you are using a hardware debug tool, additional steps may be required to write to EEPROM on the part. See documentation for your tool.

13.9 FILE (EDITOR) WINDOW

A File window contains the source code for your application. You may open an existing source code text file in this window (*File>Open*) or open a blank window and type in your code (*File>New*). Once you have included a file in a project, and if you have it open when saving the project, it will open every time you open your project (*Project>Open*).

Text in a file window is edited using the MPLAB Editor. See documentation for the editor for more on its operation.

- File Window Display
- File Window Menu
- File Window FAQ

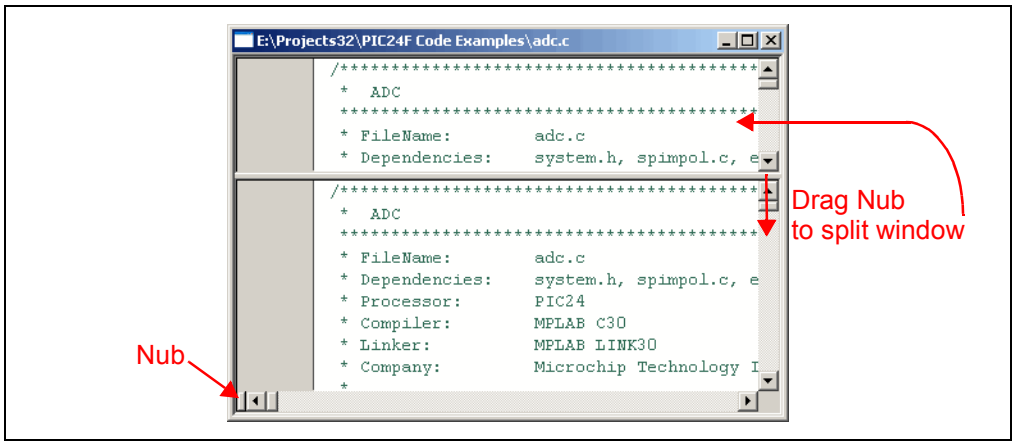
13.9.1 File Window Display

You open a File (Editor) window to display application source code by selecting *File>Open* (for existing code), by selecting *File>New* (for entering new code) or by double clicking on a file in the Project window.

13.9.1.1 SPLIT WINDOW

You may split the File window by clicking and dragging on the rectangular “nub” before the scroll arrow (see Figure 13-7). Splitting the window allows you to view different sections of long code.

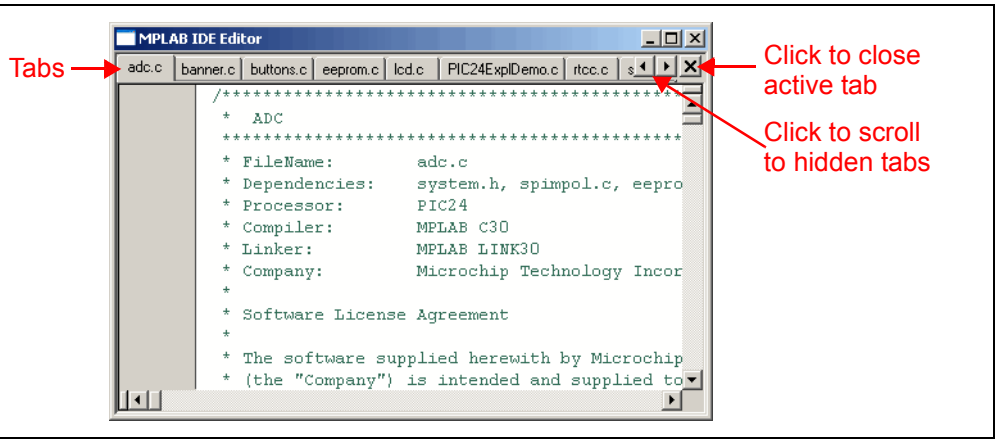
FIGURE 13-7: SPLIT FILE WINDOW



13.9.1.2 TABBED WINDOW

If you have selected Tabbed windows from *Edit>Properties*, **General** tab, “Use Tabbed Window”, you may close any active tab by clicking on the “X” button to the extreme right of all the tabs. Use the arrow buttons to scroll to hidden tabs if you have many source files open in the window.

FIGURE 13-8: TABBED FILE WINDOW



13.9.1.3 WINDOW MENUS

There is a right mouse button menu available in this window which contains several text editing and debugging options. Additional text editing options may be found on the Edit menu (see MPLAB Editor help for more information). Additional debugging options may be found on the Debugger menu (after you have selected a debug tool).

13.9.1.4 EXECUTION/DEBUG FEATURES

Code execution information may be displayed in the form of symbols in the gutter of the window. See

If you have enabled mouseovers (*Edit>Properties*, **Tooltips** tab) and you mouseover (move your cursor over) a variable, the variable address and value are displayed in a pop-up.

13.9.2 File Window Menu

Below are the menu items in the File window right mouse button menu.

Remove Filter Trace

Remove filter trace tags on selected code text. See **Section 16.6 "Working with Debug Features"**.

Remove All Filter Traces

Remove all filter trace tags on code text. See **Section 16.6 "Working with Debug Features"**.

Add Filter-out Trace

Add filter-out trace tags on selected code text. See **Section 16.6 "Working with Debug Features"**.

Add Filter-in Trace

Add filter-in trace tags on selected code text. See **Section 16.6 "Working with Debug Features"**.

Close

Close active window.

Set/Remove Breakpoint

Set or remove a breakpoint at the currently selected line.

Enable/Disable Break

Enable or disable a breakpoint at the currently selected line.

Breakpoints

Disable, enable or remove all breakpoints.

Run to Cursor

Run the program to the current cursor location. Formerly Run to Here.

See also **Section 8.4.4.2 “MPLAB IDE Features Impacted When All Breakpoints are Used”**.

Set PC at Cursor

Set the Program Counter (PC) to the cursor location.

Center Debug Location

Center the current PC line in the window.

GoTo

Open the “Go To” dialog (**Section 14.15 “Go To Dialog”**).

GoTo Locator

Go to where the selected C code symbol is defined. For example, right click on a function (in the Project window, Symbol tab, or in code in a File window) and select “GoTo Locator” to go to the line in code where this function is declared.

To use this option, “Enable Tag Locators” must be selected. Select “Properties” and in the Editor Options dialog, **General** tab, click on “Enable Tag Locators”.

Cut

Deletes the selected text in the current window and places it on the clipboard. After this operation you can paste the deleted text into another MPLAB Editor window, into a different location in the same MPLAB Editor window, or into another Windows application.

Copy

Copies the selected text in the current window onto the clipboard. After this operation, you can paste the copied text into another MPLAB Editor window, into another location in the same MPLAB Editor window, or into another Windows application.

Paste

Pastes the contents of the clipboard into the current window at the insertion point. You can only perform this operation if the clipboard contains data in text format. MPLAB Editor does not support pasting of bitmaps or other clipboard formats.

Delete

Deletes the selected text.

Add to Project

Insert file into the current project. Depending on the type of file, MPLAB IDE will sort the file into the correct type within the project window tree.

External DIFF

Opens the External DIFF dialog to perform a difference between Files/Folders via an External Diff Utility. See MPLAB Editor help for details.

Advanced

Set advanced text features. These include:

- Making selected text all uppercase or all lowercase.
- Making selected text a comment or not a comment.
- Formatting a block comment. See MPLAB Editor help for more information.
- Indenting or outdenting text.
- Matching if a brace, bracket or parenthesis. Go to the brace that is the matching brace for the brace at the cursor. This function works for curly braces, parentheses, angle brackets and square brackets.

Bookmark

Manage bookmarks. Toggle (enable/disable) a bookmark, move to the next or previous bookmarks or clear all bookmarks. See MPLAB Editor on-line help for more on bookmarks.

Code Folding

For Code Folding enabled (**Section 16.2.1 "Editor Options Dialog"**), control the amount of folding/unfolding using these subcommand items.

Text Mode

Customize text display based on development mode, i.e., device architecture and programming language.

Help

Display help on the File (Editor) window.

Properties

Set Editor Options, either display or functional options. See **Section 16.2.1 "Editor Options Dialog"**.

13.9.3 File Window FAQ

How do I:

Color my code based on its type?

MPLAB IDE will automatically color your code. To change this setting, right click in the window, select Text Mode and then select your desired type, either device-specific assembly, C, Basic or SCL (simulator control language).

Set the color-coding?

Right click in the window and select Properties. In the Editor Options dialog, click the **Text** tab and **Choose Colors**.

Set a breakpoint?

See **Section 8.4 "Breakpoints"**.

Use the Editor?

See *Help>Topics*, "System", "MPLAB Editor" for on-line editor help.

13.10 FILE REGISTERS WINDOW

The File Register window displays all the file registers of the selected device. When a file register value changes, or the processor is interrogated, the data in the File Register window is updated.

> **Note:** To speed up debugging with certain hardware tools, close this window. Use the SFR or Watch window instead.

- File Registers Window Display
- File Registers Window Menu
- File Registers Window FAQ

13.10.1 File Registers Window Display

You may change the way data is displayed in the file register window by clicking on one of the buttons on the bottom of the window.

- Hex
- Symbolic
- Dual Port (dsPIC33FJ DSC/PIC24HJ MCU devices only)
- XY Data (dsPIC DSC devices only)

13.10.1.1 HEX

This format displays file register information as hex data. The window will have the following columns:

- Address – Hexadecimal address of the data in the next column.
- *Data Blocks* – Hexadecimal data, shown in 1- or 2-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

13.10.1.2 SYMBOLIC

This format displays each file register symbolically with corresponding data in hex, decimal, binary and character formats. The window will have the following columns:

- Address – Data hexadecimal address.
- Hex – Hexadecimal data, shown in 1- or 2-byte blocks.
- Decimal – Data in decimal format.
- Binary – Data in binary format.
- Char – Data in character format.
- Symbol Name – Symbolic name for the data.

13.10.1.3 DUAL PORT (dsPIC33FJ DSC/PIC24HJ MCU DEVICES ONLY)

This format displays file register information as hex data. The window will have the following columns:

- Address – Data hexadecimal address.
- DMA Address – Offset from the silicon DMA address.
- *Data Blocks* – Hexadecimal data, shown in 1- or 2-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

For more information on dsPIC33F DSC and PIC24H MCU devices, see the Microchip website for device data sheets and dsPIC33F and PIC24H Reference Manual sections.

13.10.1.4 XY DATA (dsPIC DSC DEVICES ONLY)

This format displays file register information as hex data. The window will have the following columns:

- Address – X hexadecimal address of the data.
- Y Bus – Y hexadecimal address of data, if supported.
- *Data Blocks* – Hexadecimal data, shown in 2-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

For more information on dsPIC DSC devices, see "*dsPIC30F Family Reference Manual*" (DS70046).

13.10.2 File Registers Window Menu

Below are the menu items in the File Registers right mouse button menu.

Close

Close this window.

Full Memory Update

Updates the entire contents of this window after a halt or single step if enabled.

By default, this option is enabled. If the window is open, only the data visible is updated. If the window is closed, no data is updated.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Go to the specified item of the Go To dialog. (See **Section 14.15 "Go To Dialog"**.)

Import Table

Open the Import dialog (see **Section 14.11 "File Management Dialog"**).

Export Table

Open the Export dialog (see **Section 14.11 "File Management Dialog"**).

Fill Registers

Fill registers from Start Address to End Address with the value in Data. See **Section 14.12 "Fill Memory/Registers Dialog"**.

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of an "Output Range". Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the File Registers window.

Properties

Select background colors for SRFs and unallocated memory. Also, set up generic window properties (see **Section 14.23 “Properties Dialog”**).

13.10.3 File Registers Window FAQ

How do I:

Fill all registers with a value?

Right click in the window and select “Fill Registers” to open a dialog where you may enter fill data.

13.11 FLASH DATA WINDOW

The Flash Data window displays Flash data for any microcontroller device that has Flash data memory (e.g., PIC12F519). Data/opcode hex information of the selected device is shown. When a Flash data register value changes or the processor is halted, the data in the Flash Data window is updated.

Note:	The start of Flash data memory needs to be specified for use with programmers. The start should be at 0x400 (org H‘400’). However, please check the programming specification for your selected device to determine the correct address.

- Flash Data Window Display
- Flash Data Window Menu
- Flash Data Window FAQ

13.11.1 Flash Data Window Display

This display format shows data in the following columns:

- Address – Hexadecimal address of the data in the next column.
- Data Blocks – Hexadecimal data, shown in 1- or 2-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

13.11.2 Flash Data Window Menu

Below are the menu items in the Flash data right mouse button menu.

Close

Close this window.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Go to the specified item of the Go To dialog. (See **Section 14.15 “Go To Dialog”**.)

Import Table

Open the Import dialog (see **Section 14.11 "File Management Dialog"**).

Export Table

Open the Export As dialog (see **Section 14.11 "File Management Dialog"**).

Fill Memory

Fill memory from Start Address to End Address with the value in Data. See **Section 14.12 "Fill Memory/Registers Dialog"**.

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of an "Output Range". Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the Flash Data window.

Properties

Set up window properties. See **Section 14.23.2 "General Tab"**.

13.11.3 Flash Data Window FAQ

How do I:

Fill Flash data memory with a value?

Right click in the window and select "Fill Registers" to open a dialog where you may enter fill data.

If you are using a hardware debug tool, additional steps may be required to write to Flash data on the part. See documentation for your tool.

13.12 HARDWARE STACK WINDOW

The Hardware Stack window displays the contents of the hardware stack. The number of available levels depends on the selected device.

Note:	If Disable Stack Overflow Warning is cleared, MPLAB IDE will display stack overflow and underflow warnings when they occur. This is not supported on all processor modules.

- Hardware Stack Window Display
- Hardware Stack Window Menu
- Hardware Stack Window FAQ

13.12.1 Hardware Stack Window Display

Data is displayed in the following columns:

- TOS (if available) – A pointer showing the Top-of-Stack (TOS).
- Stack Level – Stack level number. The total number of levels is dependent on the device selected.
- Stack Return Address – Return addresses on the stack.
- Location – Function name + offset. Information on location in a function.

Stack Return Address

Displays the current contents of the hardware stack. Previously used values are still displayed, since those values are still in the device. The TOS indicator shows the current stack pointer location.

13.12.2 Hardware Stack Window Menu

Below are the menu items in the Hardware Stack right mouse button menu.

Close

Close this window.

Pop Stack

Move address at Top-of-Stack onto program counter.

Set Top-of-Stack

Set the Top-of-Stack to the current cursor location.

Center Stack Location

Center the currently selected line in the window.

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 “File Management Dialog”**), with the addition of an “Output Range”. Enter the “Start” and “End” lines to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the Hardware Stack window.

Properties

Set up fonts and colors. See **Section 14.23 “Properties Dialog”**.

13.12.3 Hardware Stack Window FAQ

How do I:

Understand the Return Address information?

See **Section 13.12.1 “Hardware Stack Window Display”**

Pop the stack?

Click on the right mouse button in the window to open a menu. Select Pop Stack.

Set Top-of-Stack (TOS)?

Click on the right mouse button in the window to open a menu. Select Set Top-of-Stack.

13.13 LCD PIXEL WINDOW

This window is visible for devices that support this feature, i.e., PIC16F913/914, PIC16F916/916, and PIC16F946.

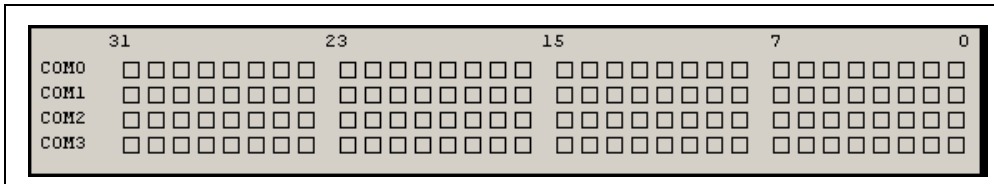
The LCD Pixel window displays LCD output (upper portion of window) when the device LCD function is enabled (SFRs in lower portion of window). All LCD pixels (squares) will be background gray when LCD functionality is disabled. When enabled, pixels will appear either white (off or '0') or dark gray (on or '1').

- LCD Pixel Window Display
- LCD Pixel Window Menu
- LCD Pixel Window FAQ

13.13.1 LCD Pixel Window Display

For devices that support LCD output (PIC16F913/914, PIC16F916/916, and PIC16F946), you can view results of this by selecting *View>LCD Pixel*.

FIGURE 13-9: LCD PIXEL DISPLAY



- LCD display area

  LCD disabled: all LCD pixels will be background gray color

  LCD enabled: off ('0') LCD pixels will appear white and on ('1') LCD pixels will appear dark gray

Note: You will have to enable LCD functionality through an LCD control register bit (e.g., for PIC16C924, set LCDCON register bit 7 (LCDEN) to '1'). Consult your device data sheet.

- Address – Address of LCD-related SFR. Click to alternate list order of address, i.e., high/low or low/high.
- SFR Name – Symbolic name of special function register related to LCD function.
- Hex – Hexidecimal value of SFR contents.
- Decimal – Decimal value of SFR contents.
- Binary – Binary value of SFR contents.
- Char – ANSI character value of SFR contents.

13.13.2 LCD Pixel Window Menu

Below are the menu items in the LCD Display window right mouse button menu.

Close

Close this window.

Bitfield Mouseover

Enable/disable bitfield mouseover. When enabled, mousing over a symbol in the window will pop up a display of the bitfield for that symbol.

Find

Find text in this window specified in the Find dialog. See **Section 14.14 “Find and Replace Dialogs”**.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Go to the specified item of the Go To dialog. (See **Section 14.15 “Go To Dialog”**.)

Output to File

Write the displayed window contents to a text file.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the LCD Pixel window.

Properties

Set up window properties. See **Section 14.23 “Properties Dialog”**.

13.13.3 LCD Pixel Window FAQ

How do I:

Enable LCD operation?

Consult your device data sheet to determine the LCD control registers needed to set up LCD functionality (e.g., for PIC16C924, set LCDCON register bit 7 (LCDEN) to ‘1’ to enable). Set up these registers for LCD operation in your code or find these SFR's in the lower portion of the window and click in a value field of the SFR (hex, decimal, binary or char) to enter an appropriate value.

Turn LCD pixels on or off in the display?

Consult your device data sheet to determine the LCD pixel control registers. Set up these registers for LCD operation in your code or find these SFRs in the lower portion of the window and click in a value field of the SFR (hex, decimal, binary or char) to enter an appropriate value. The corresponding value should appear in the LCD display in the upper portion of the window.

13.14 LOCALS WINDOW

The Locals window allows you to monitor automatic variables that are local in scope. This applies to projects using high-level languages (C, BASIC, etc.) and the linker.

- Locals Window Display
- Locals Window Menu
- Locals Window FAQ

13.14.1 Locals Window Display

Only automatic variables that are local in scope are displayed. If stepping from one function to another, the variables of one function will be replaced by the variables from the other function.

To view variables regardless of scope, use the Watch Window.

Data is displayed in the following columns:

- Address – Hexadecimal address of the variable.
- Symbol Name – Name of the variable.
- Value – Current value of the variable.

  To change a value, click in the Value column and type the new value. Then click outside the column to see the value update.
- Additional Columns – Hex, Decimal, Binary, Char

  You may add radix information to the display by right clicking on the column header bar.

Variables may be rearranged using drag-and-drop.

Window information may be sorted by column. Simply click on the column header to sort data according to that column.

13.14.2 Locals Window Menu

Below are the menu items in the Locals window right mouse button menu.

Close

Close this window.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Import Table

Save data to a table. See **Section 14.29 "Table Setup Dialog"**.

Export Table

Load data from a table. See **Section 14.29 "Table Setup Dialog"**.

Output to File

Write the displayed window contents to a text file.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the Locals window.

Properties

Opens the Locals dialog, so you can set up fonts and colors, as well as other Locals window properties. See **Section 14.33 "Watch/Locals Dialog"**.

13.14.3 Locals Window FAQ

How do I:

Save the window contents to a file?

Click the right mouse button in the window to open a menu. Select Export to save variable values to a file.

Load a previously saved window?

Click the right mouse button in the window to open a menu. Select Import to load variable values.

Change the order of variables?

Simply drag-and-drop items where you want them in the list.

13.15 LOGIC ANALYZER WINDOW

This window acts like a real logic analyzer, displaying the digital levels of all the selected pins during a selected time period.

The data is logged in conjunction with the Trace buffer to create synchronization between the two. See **Section 13.24 "Trace Memory Window"** for information on how to enable data capture to the Trace buffer.

All port data will be logged every cycle (or at the specified sample rate) as done in the Trace buffer. There is a 64K buffer for each port.

- Logic Analyzer Display
- Logic Analyzer Menu
- Logic Analyzer FAQ

13.15.1 Logic Analyzer Display

The Logic Analyzer display needs to be populated with signals/channels to observe. Click **Channels** to open the Configure Channel Dialog.

The display may have a maximum of 64 channels. The data shows a discrete level 0/1 change of state for each bit.

Window Controls

- "Trigger Position" – Three radio buttons allow you to specify where the trigger is to occur: at the start of capture, in the center of capture, or at the end of capture.
- "Trigger PC =" – The program counter (PC) value of the trigger.
  - Trigger Now – Trigger on clicking of this button. Current PC value is placed in the text box.
  - Trigger Clear – Clears the trigger value and data buffers for a new capture. The data buffers are also cleared if the display is closed.
- "Time Base" – The time base is in the form of instruction cycles.
- "Mode" – Displays the selected trigger mode, i.e., simple or complex.
- Channels – Click button to add or remove signals/channels from display.

Window Toolbar

- Scroll (Axes) – Click and drag on an axis to scroll the axis. Arrow keys and Page Up/Page Down keys may be used also.
- Zoom (Axes) – Click and drag on an axis to zoom the scale in or out. Arrow keys and Page Up/Page Down keys may be used also.
- Zoom Out All Axes – Click to zoom out entire display.
- Zoom In All Axes – Click to zoom in entire display.
- Zoom Box – Click and drag on display to select a region for zoom in.
- Cursor – Show a cursor on the display.
- Copy to Clipboard – Copy raw data to clipboard.
- Save to File – Save display as graphic.
- Print – Print display as graphic.
- Preview – Print preview of selected area of chart.

Signal/Channel Display

- Drag-and-drop the order of the display lines vertically after they have been defined (like Watch window variables).

13.15.2 Logic Analyzer Menu

Below are the menu items in the Logic Analyzer right click menu.

Close

Close dialog.

Zoom To Fit

Magnify (expand) or shrink to a time period that is viewable on the screen.

Go To

Enter a time to which to scroll the center of the display.

Refresh

Update the display with current trace information.

Clear

Clears data on the Logic Analyzer window.

Properties

Set up window properties. See **Section 14.19 "Logic Analyzer Properties Dialog"**.

Import/Export Table

Import from or export to a logic data file.

13.15.3 Logic Analyzer FAQ

How do I:

Add a signal to the logic analyzer display?

Click **Channels** and add the signal name, found under “Available Signals”. You may choose from individual pins or buses. Buses, or groups of pins, can be created by clicking **Configure Bus(es)**. Click **OK** to see the signal appear on the logic analyzer display as a channel.

For more information, see:

- **Section 14.7 “Configure Channel Dialog”**
- **Section 14.8 “Configure Bus Dialog”**

13.16 MEMORY WINDOW (PIC32MX DEVICES ONLY)

The Memory window displays locations in the range of program and/or data memory for the currently selected PIC32MX device.

- Memory Window Display
- Memory Window Menu
- Changing Window Data and Properties
- Code Display Window Symbols
- Memory Window FAQ

13.16.1 Memory Window Display

You may select the type of memory displayed in the window by clicking on one of the tabs on the bottom of the window:

- Code View - Program Memory
- Data View - Data Memory

13.16.1.1 CODE VIEW - PROGRAM MEMORY

This format displays program memory information as hex code. The window will have the following columns:

- Line – Reference line number corresponding to memory address.
- Address – Physical hexadecimal address of the opcode in the next column.
- Opcode – Hexadecimal opcode, shown in 4-byte blocks. The opcode that is highlighted represents the current location of the program counter.
- Label – Opcode label in symbolic format.
- Disassembly – A disassembled version of the opcode mnemonic.
- Virtual – Virtual hexadecimal address of the opcode as defined by the Bus Matrix.

13.16.1.2 DATA VIEW - DATA MEMORY

This format displays data memory information as hex code. The window will have the following columns:

- Address – Hexadecimal address of the data in the next column.
- Data Blocks – Hexadecimal data, shown in 4-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.
- Virtual – Virtual hexadecimal address.

13.16.2 Memory Window Menu

Below are the menu items in the Memory right mouse button menu.

Close

Close this window.

Set/Remove Breakpoint

Set or remove a breakpoint at the currently selected line.

Enable/Disable Break

Enable or disable a breakpoint at the currently selected line.

Breakpoints

Disable, enable or remove all breakpoints.

Run To Cursor

Run the program to the current cursor location. Formerly Run to Here.

See also **Section 8.4.4.2 "MPLAB IDE Features Impacted When All Breakpoints are Used"**.

Set PC at Cursor

Set the Program Counter (PC) to the cursor location.

Center/Update Debug Location

Center or update the current PC line in the window.

Cross Tab Tracking

Choose to track, via highlighting, the current PC or address line, across all window tabs/views. Choose "No Tracking" to disable this feature.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Go to the specified item of the Go To dialog. (See **Section 14.15 "Go To Dialog"**.)

Import Table

Open the Import dialog (see **Section 14.11 "File Management Dialog"**).

Export Table

Open the Export As dialog (see **Section 14.11 "File Management Dialog"**).

Fill Memory

Fill memory from Start Address to End Address with the value in Data. See **Section 14.12 "Fill Memory/Registers Dialog"**.

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 “File Management Dialog”**), with the addition of an “Output Range”. Select the type of range, either “Lines” or “Address”, and then enter the “Start” and “End” values to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the Memory window.

Properties

Set up window properties. See **Section 14.23 “Properties Dialog”**.

13.16.3 Memory Window FAQ

How do I:

Fill program memory with a value?

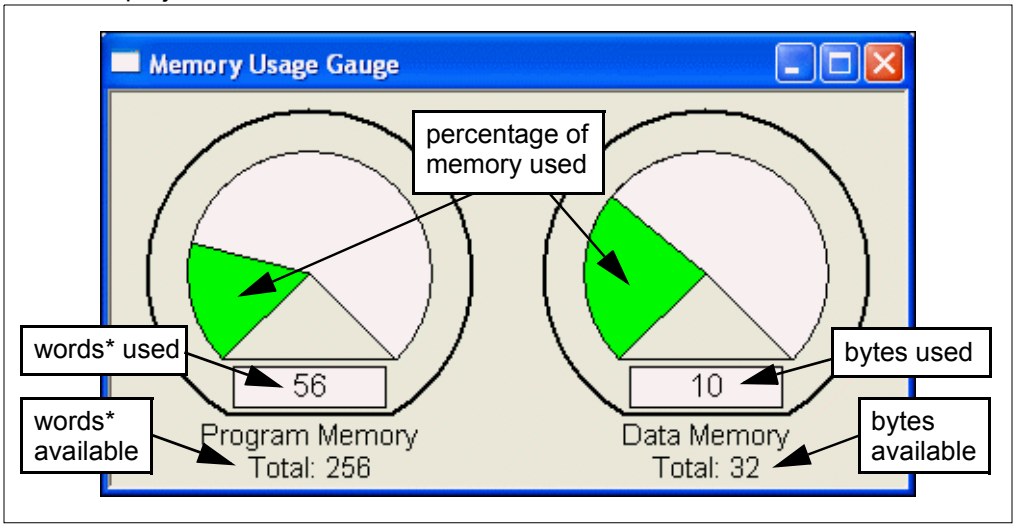
Right click in the window and select “Fill Memory” to open a dialog where you may enter fill data.

Set a breakpoint?

See **Section 8.4 “Breakpoints”**.

13.17 MEMORY USAGE GAUGE

This window displays the amount of program and data memory currently being used in the active project.



* Words are device-dependent. See **Section 9.3 "Program and Data Memory"**.

Considerations

You must use the linker to generate your code for this gauge to work, i.e., a `*.cof` or `*.elf` debug file must be generated.

If you are *not* using the full version of the MPLAB C Compiler for PIC32MX MCUs, the memory gauge will *not* show the full amount of device memory.

You must build your project to update the gauge. Therefore, if you close one project and open another, the gauge will not update from the first project until you build the second.

You will see that the memory usage gauge reading is different than that of memory usage in a generated map file. This is because the gauge shows only values in program memory, where as the map file includes all values, i.e., .config, .eedata, and .idlocs.

13.18 OUTPUT WINDOW

Selecting *View>Output* opens the Output window. This window contains tabbed information about program output.

- **Build** tab – Lists messages from a project build. Build messages are the result of language tools selected (*Project>Select Language Toolsuite*) and build options set (*Project>Build Options>Project*).
- **Version Control** tab – Displays version control information, if a version control system is used in the project (*Project>Version Control*).
- **Find in Files** tab – Lists the result of *Edit>Find in Files*.
- Depending on the functionality selected, other output tabs may be available.

Below are the menu items in the Output window right mouse button menu.

Select All

Selects all text and graphics in the Edit window.

Copy

Copies the selected text in the current window onto the clipboard. After this operation, you can paste the copied text into another MPLAB Editor window, into another location in the same MPLAB Editor window, or into another Windows application.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Clear Page

Remove all text in the selected output tab.

Help

Display help on the Output window.

Properties

Set up window properties. See **Section 14.23 "Properties Dialog"**.

13.19 PROGRAM MEMORY WINDOW

The Program Memory window displays locations in the range of program memory for the currently selected processor. If external program memory is supported by the selected device and enabled, it will also appear in the Program Memory window.

- Program Memory Window Display
- Program Memory Window Menu
- Program Memory Window FAQ

13.19.1 Program Memory Window Display

You may change the way opcodes are displayed in the program memory window by clicking on one of the buttons on the bottom of the window:

- Opcode Hex
- Machine or Symbolic
- PSV Mixed (dsPIC DSC/PIC24 devices only)
- PSV Data (dsPIC DSC/PIC24 devices only)

13.19.1.1 OPCODE HEX

This format displays program memory information as hex code. The window will have the following columns:

- Address – Hexadecimal address of the opcode in the next column.
- *Opcode Blocks* – Hexadecimal opcode, shown in 2- or 3-byte blocks. For most PIC MCUs these blocks represent words. For PIC18CXXX devices, the blocks represent 2 bytes. For dsPIC DSC devices, the blocks represent 3 bytes.

  The opcode block that is highlighted represents the current location of the program counter.
- ASCII – ASCII representation of the corresponding line of opcode.

13.19.1.2 MACHINE OR SYMBOLIC

Machine format displays disassembled hex code with no symbolic information. Symbolic format displays disassembled hex code with symbols. The window will have the following columns:

- Debug Info – Information useful for debugging. A pointer shows the current location of the program counter.
- Line – Reference line number
- Address – Opcode hexadecimal address.
- Opcode – Hexadecimal opcode, shown in 2- or 3-byte blocks. For most PIC MCUs these blocks represent words. For PIC18CXXX devices, the blocks represent 2 bytes. For dsPIC DSC devices, the blocks represent 3 bytes.
- Label (Symbolic Only) – Opcode label in symbolic format.
- Disassembly – A disassembled version of the opcode mnemonic.

13.19.1.3 PSV MIXED (dsPIC DSC/PIC24 DEVICES ONLY)

This format displays program memory as opcode and the PSV area (CORCON register, PSV bit set). The window will have the following columns:

- *Debug Info* – Information useful for debugging. A pointer shows the current location of the program counter.
- Line – Reference line number.
- Address – Opcode hexadecimal address.
- Opcode – Hexadecimal opcode, shown in 3-byte blocks.
- PSV Address – Data space hexadecimal address of the opcode.
- Data – Opcode formatted as data.
- Label – Opcode label in symbolic format.
- Disassembly – A disassembled version of the opcode mnemonic.

For more information, see the "*dsPIC30F Family Reference Manual"* (DS70046).

13.19.1.4 PSV DATA (dsPIC DSC/PIC24 DEVICES ONLY)

This format displays program memory as file registers, for when program space is visible in data space (CORCON register, PSV bit set). The window will have the following columns:

- Address – Program space hexadecimal address of the data.
- PSV Address – Data space hexadecimal address of the data.
- Data Blocks – Hexadecimal data, shown in 3-byte blocks.

The data block that is highlighted represents the current location of the program counter.

- ASCII – ASCII representation of the corresponding line of data.

For more information, see the *dsPIC30F Family Reference Manual* (DS70046).

13.19.2 Program Memory Window Menu

Below are the menu items in the Program Memory right mouse button menu.

Close

Close this window.

Set/Remove Breakpoint (Machine/Symbolic Only)

Set or remove a breakpoint at the currently selected line.

Enable/Disable Break (Machine/Symbolic Only)

Enable or disable a breakpoint at the currently selected line.

Breakpoints

Disable, enable or remove all breakpoints.

Run To Cursor

Run the program to the current cursor location. Formerly Run to Here.

See also **Section 8.4.4.2 “MPLAB IDE Features Impacted When All Breakpoints are Used”**.

Set PC at Cursor

Set the Program Counter (PC) to the cursor location.

Center Debug Location

Center the current PC line in the window.

Cross Tab Tracking

Choose to track, via highlighting, the current PC or address line, across all window tabs/views. Choose “No Tracking” to disable this feature.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Go to the specified item of the Go To dialog. (See **Section 14.15 “Go To Dialog”**.)

Import Table

Open the Import dialog (see **Section 14.11 “File Management Dialog”**). “Opcode Hex” must be selected for this item to be available.

Export Table

Open the Export As dialog (see **Section 14.11 “File Management Dialog”**). “Opcode Hex” must be selected for this item to be available.

Fill Memory

Fill memory from Start Address to End Address with the value in Data. See **Section 14.12 “Fill Memory/Registers Dialog”**.

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of an "Output Range". Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the Program Memory window.

Properties

Set up window properties. See **Section 14.23 "Properties Dialog"**.

13.19.3 Program Memory Window FAQ

How do I:

Enable external (off-chip) memory, for parts that support this?

Select *Configure>External Memory*. In the External Memory dialog, check "Use External Memory", enter a range and click **OK**. External Memory will now appear in the Program Memory window.

Note:	For some tools, you may actively have to upload external memory to MPLAB IDE before values will appear in the window.

Fill program memory with a value?

Right click in the window and select "Fill Memory" to open a dialog where you may enter fill data.

Set a breakpoint?

See **Section 8.4 "Breakpoints"**.

13.20 PROJECT WINDOW

The Project window contains a summary of information about the project. Select *View>Project* to alternately view or close the Project window.

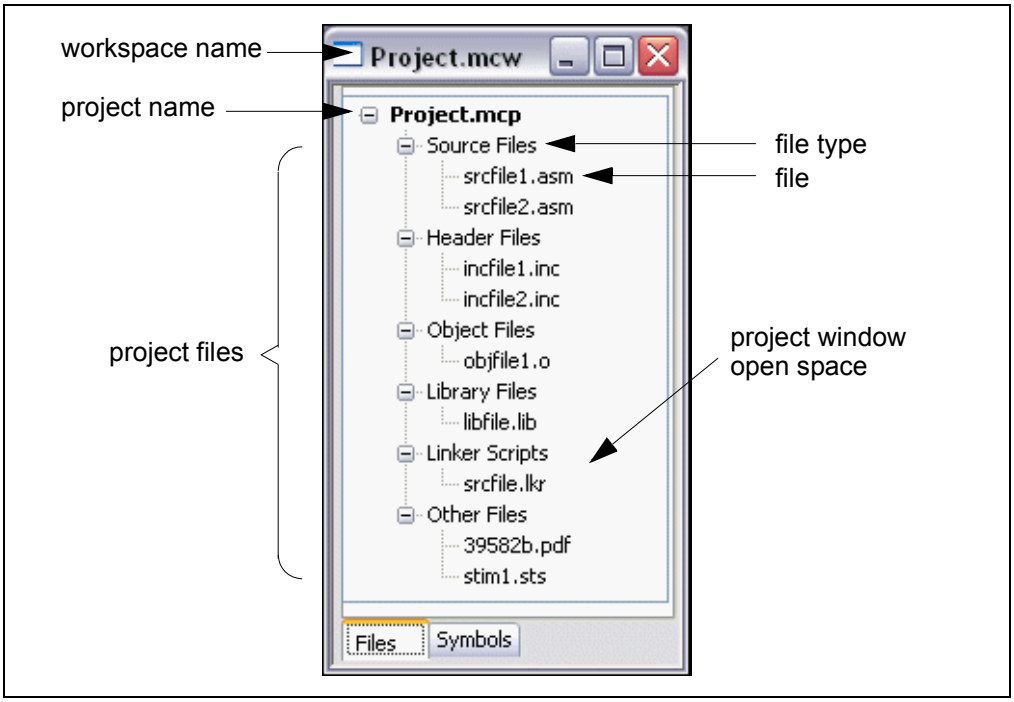
The Files tab displays project files in a tree structure. The Symbols tab displays C code symbols, i.e., functions, variables, etc.

- Project Window Display – Files Tab
- Project Window Menus – Files Tab
- Project Window Display – Symbols Tab
- Project Window Menus – Symbols Tab
- Project Window FAQ

13.20.1 Project Window Display – Files Tab

The title bar of the Project window contains the name of the project workspace. The window itself contains a list of projects and project files by file type in tree format. The general structure of the project tree for a single project is shown in Figure 13-10.

FIGURE 13-10: PROJECT TREE GENERAL STRUCTURE



If the project name is in bold, it is the active project. If the project name is followed by an asterisk (e.g., `c:\project1\project.mcp*`), either the project settings have changed or the project is new, and it needs to be saved.

For information on the types of files listed in this window, see **Section 6.5 "Project Folders and Files"**. To add files to the window, you may drag-and-drop the file from Windows Explorer or right click on the file type and select Add Files.

Window functions are as follows:

- Clicking the "+/-" in front of a file type will expand/collapse the file list for that type.
- Drag-and-drop a file to move it between folders.
- Double clicking on a file will open that file.
- Depending on the area of the window, right clicking will open various menus (see **Section 13.20.2 "Project Window Menus – Files Tab"**).
- Hitting <Tab> will move between the **Files** and **Symbols** tabs.

13.20.2 Project Window Menus – Files Tab

Depending on where you click in the project window, different menus are available.

- Display Menu
- Project Menu
- File Type Menu
- File Menu

13.20.2.1 DISPLAY MENU

If you right click on any open space in the project window, a menu with project display commands will be displayed.

Refresh

This command causes the IDE to go out and check the status of all the files in all projects.

For version control system files, the files' status is updated in the project window.

For missing files that you have relocated, "file not found" text will be removed.

Help

Open the help file to this section on the Project window.

Preferences

Opens a dialog to set project preferences. See **Section 14.21 "Project-Display Preferences Dialog"**.

13.20.2.2 PROJECT MENU

If you right click on a project file, a menu with project level commands will be displayed.

Set Active

Select a project as the active project in the workspace. For more on active projects, see **Section 6.9 "Using Multiple Projects in a Single Workspace"**. To enable Quickbuild, select None.

Package in .zip

Package the current project into a `.zip` file. While MPLAB IDE has the ability to zip files up, it cannot unzip them. So a separate program will be required to unzip the project.

Clean

Removes all intermediary project files, such as object, hex and debug files. These files are recreated from other files when a project is built.

Locate Headers

Locates all the header files (`.h`) called out in the project files and presents them in checklist window so they may be added to the project.

Although header files don't need to be added to your project in order for that project to build, there are several reasons why you might want to add them to the Project window under "Header Files":

- The files can be opened from the Project window for editing instead of hunting for them on your computer.
- The "Save Project As" feature works better
- The "Package Project as .zip" feature works better

The project must be a C code project.

Export Makefile

Under *Project>Build Options>Project*, **Directories** tab, you must have selected "Assemble/Compile/Link in the project directory" under "Build Directory Policy" for this feature to work.

Export to a file the instructions to Make the project. Use this file to Make your project outside of MPLAB IDE, i.e., with a `make`.

Build All

YOU MUST HAVE A PROJECT OPEN BEFORE THIS OPTION IS VISIBLE.

Build all files in the project, i.e., compile/assemble all files as specified in the project. Build All is available in the right mouse button menu of the project window as well.

Make

YOU MUST HAVE A PROJECT OPEN BEFORE THIS OPTION IS VISIBLE.

Build only the files in the project that have changed, i.e., compile/assemble these files as specified in the project. Make is available in the right mouse button menu of the project window as well.

Build Options

Set and view options for the project and individual files. See **Section 14.5 "Build Options Dialog"**.

Save

Save the active project.

Save As

Save the active project to a new location/name. See **Section 14.24 "Save Project As Dialog"**.

Close

Close/remove the selected project in the workspace.

Add Files

Insert files into the project. Depending on the type of file, MPLAB IDE will sort the files into the correct type within the project window tree.

Note:	Adding Header Files to the project window will not cause these files to be added to the project build. You must use a `#include` statement in your code.

Add New File

Requests a name for the new file in a dialog (see **Section 14.11 "File Management Dialog"**). On **Save**, the file is added to the project and a new file window is opened.

Depending on the type of file, MPLAB IDE will sort the files into the correct type within the project window tree.

Note:	Adding Header Files to the project window will not cause these files to be added to the project build. You must use a `#include` statement in your code.

Reload

This command causes MPLAB IDE to go out to disk and read the project file in again. It also does a Refresh immediately after. This functionality is useful if you modify your project and then decide you don't want the modifications. As long as you haven't saved and overwritten the old project, you can "Reload" and get back to your original project.

Refresh

This command causes MPLAB IDE to go out and check the status of all the files in the project.

For version control system files, the files' status is updated in the project window.

For missing files that you have relocated, "file not found" text will be removed.

Select Language Toolsuite

Select the toolsuite you will use for your project, e.g., Microchip MPASM Toolsuite. See **Section 14.26 "Select Language Toolsuite Dialog"**.

Version Control

Set up your project to use files from a version control system. See **Section 14.31 "Version Control Dialog"**.

13.20.2.3 FILE TYPE MENU

If you right click on a file type in the project tree, a menu with file type commands will be displayed.

Library Link Order (Library Files Only)

Opens the Library Link Order dialog, which shows you the true order of the library files and allows you to resequence them. This is important for 16-bit compiler users.

Add Files

Insert files into the project. Depending on the type of file, MPLAB IDE will sort the files into the correct type within the project window tree.

> **Note:** Adding Header Files to the project window will not cause these files to be added to the project build. You must use a `#include` statement in your code.

Create Subfolder

Create a "virtual" subfolder to help you further organize files in your project. These folders have no relation to the actual directory structure on your PC.

Filter

Change the way files are filtered to determine file type. For example, to be able to add an assembly file with extension .a under Source Files, add .a to the list of file extensions in the Filter dialog.

13.20.2.4 FILE MENU

If you right click on a file in the project tree, a menu with file commands will be displayed.

> **Note:** If you are using a version control system, additional commands that can be used on the selected file.

Assemble/Compile

Assemble/compile the selected file as appropriate.

Build Options

Set and view options for the project and individual files. See **Section 14.5 "Build Options Dialog"**.

Edit

Open the selected file in a window for editing, if appropriate.

Remove

Remove the selected file from the project. The file is not deleted from the directory.

External DIFF

Opens the External DIFF dialog to perform a difference between Files/Folders via an External Diff Utility. See MPLAB Editor help for details.

Is Generated

Mark a file as already generated. This is appropriate for files that are automatically generated by some process and will silence warnings about the file's absence when the project is first opened.

Locate Missing File

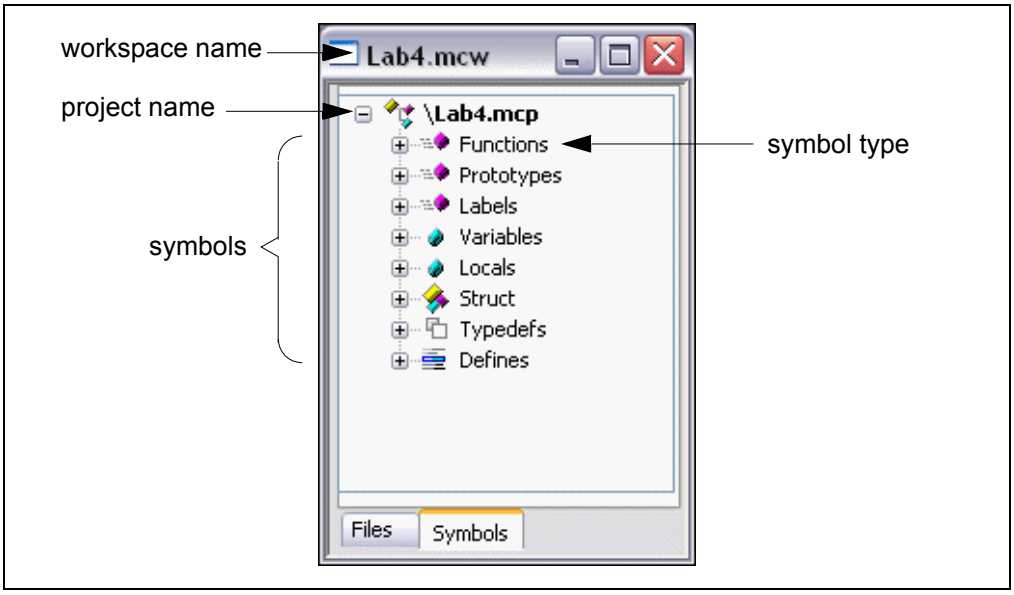
If a project file has the text "(file not found)" next to it, use this function to open a dialog to help you locate the missing file. See **Section 14.18 "Locate Missing File Dialog"**.

13.20.3 Project Window Display – Symbols Tab

To view symbols, select (check) the right mouse menu item "Enable Tag Locators".

The title bar of the Project window contains the name of the project workspace. The window itself contains a list of projects and project symbols by symbol type in tree format. The general structure of the project tree for a single project is shown in Figure 13-11.

FIGURE 13-11: SYMBOL TREE GENERAL STRUCTURE



If the project name is in bold, it is the active project. If the project name is followed by an asterisk (e.g., `c:\project1\project.mcp*`), either the project settings have changed or the project is new, and it needs to be saved.

The code used to scan the project files and create this view is an OpenSource application named CTags (http://ctags.sourceforge.net).

Window functions are as follows:

- Clicking the "+/-" in front of a symbol type will expand/collapse the symbol list for that type.
- Double clicking on a symbol will jump to that symbol in the source code file. Or clicking on a symbol to select it and then hitting <Enter> or <Shift>-8 (*) will work the same way.
- Right clicking in the window will open a menu (see **Section 13.20.4 "Project Window Menus – Symbols Tab"**).
- Hitting the right or left arrow keys will move between the **Files** and **Symbols** tabs.

13.20.4 Project Window Menus – Symbols Tab

If you right click anywhere in the project window, a menu with C code symbol commands will be displayed.

GoTo Locator

Right click on a symbol to go to where the symbol is declared in the C source code (in a File window).

Note: "Enable Tag Locators" must be enabled to use this feature.

Update Tags Now

Update C tags now, i.e., refresh the window.

Update Tags on Recompile

Update C tags when the project is built, i.e., refresh the window on a build.

Enable Tag Locators

To view the symbols on this tab of the Project window and to use "GoTo Locator", this item must be checked.

Help

Display help on the Project window.

13.20.5 Project Window FAQ

How do I:

Create/Update a project?

See **Section 6.3 "Creating/Updating any Project"**.

Build a project?

Right click on the project in the workspace that you wish to build. In the menu, select Build All to build the entire project or Make to recompile/reassemble any changed files and then build.

The results of this build will be loaded into program memory, even if the selected project is not the active project.

Use projects with workspaces?

See **Chapter 6. "Projects and Workspaces"**.

Set a project as active?

To set a project as the active project, select *Project>Set Active* *Project>project-name.mcp*, where `projectname.mcp` is the project name, or right click on the project in the Project window and select Set As Active Project.

Add a project-specific data sheet?

To add files to the window, you may drag-and-drop the file from Windows Explorer to Other Files, or right click on Other Files and select Add Files. Then double click the PDF to open it. (This requires that a PDF reader be installed.)

13.21 RTOS VIEWER WINDOW

MPLAB IDE provides a window for viewing the RTOS functions of supported RTOSs (Real-Time Operating Systems). The functions displayed are dependent on the specific RTOS. See “Readme for MPLAB IDE.txt” for a list of supported RTOSs.

The window is selectable from *Tools>RTOS Viewer* when (1) the RTOS is installed and (2) the RTOS is included in the active project. See your RTOS documentation for how to install the RTOS and what additional files you may need to add to your project.

13.22 SFR/PERIPHERALS WINDOW (PIC32MX DEVICES ONLY)

The SFR/Peripherals window displays the contents of the Special Function Registers (SFRs) that relate to the device peripherals. To view only a few SFR’s, you may prefer to use a Watch window, which may help with speed issues when using hardware debug tools (i.e., faster window update rate.)

Whenever a break occurs, the contents of the Special Function Registers are updated.

Visible Registers

If a data memory register is not physically implemented on a device, it may not appear in the SFR list. Some tools, such as simulators, may allow you to see registers that do not exist on the actual device, such as prescalers.

Single Stepping

If “Freeze Peripherals On Halt” is selected, the I/O port bits in the SFR or the Watch windows will not update when single stepping. The pin will be modified, but the read request to retrieve the new value is blocked by the freeze and cannot be updated until the next step or run command.

- SFR/Peripherals Window Display
- SFR/Peripherals Window Menu
- Changing Window Data and Properties
- SFR/Peripherals Window FAQ

13.22.1 SFR/Peripherals Window Display

Data is displayed in the following columns.

- Address – SFR physical hexadecimal address.
- Name – Symbolic name for the SFR.
- Hex – Hexadecimal value, shown in 4-byte blocks.
- Decimal – Value in decimal format.
- Binary – Value in binary format.
- Char – Value in character format.
- Virtual – SFR virtual hexadecimal address as defined by the Bus Matrix.

13.22.2 SFR/Peripherals Window Menu

Below are the menu items in the Special Function Registers right mouse button menu.

Close

Close this window.

Bitfield Mouseover

Enable/disable bitfield value mouseover. SFR window must be the active window (in focus) for this feature to work.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Go to the specified item of the Go To dialog. (See **Section 14.15 "Go To Dialog"**.)

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of an "Output Range" and tab-delimited option checkbox. Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the SFR/Peripherals window.

Properties

Set up window properties. See **Section 14.23 "Properties Dialog"**.

13.22.3 SFR/Peripherals Window FAQ

How do I:

Display different data types?

Right click on any header in the window and select or deselect the desired data types to display in columns. Alternately, right click on any header in the window, select More, and check or uncheck data types to display in columns. Then click OK.

13.23 SPECIAL FUNCTION REGISTERS WINDOW

The Special Function Registers window displays the contents of the Special Function Registers (SFRs) for the selected processor. The format provided by this window is more useful for viewing the SFRs than the normal file register window, since each SFR name is included and several number formats are presented. To view only a few SFR's, you may prefer to use a Watch window, which may help with speed issues when using hardware debug tools (i.e., faster window update rate.)

Whenever a break occurs, the contents of the Special Function Registers are updated.

Visible Registers

If a data memory register is not physically implemented on a device, it may not appear in the SFR list. Some tools, such as simulators, may allow you to see registers that do not exist on the actual device, such as prescalers.

Single Stepping

If "Freeze Peripherals On Halt" is selected, the I/O port bits in the SFR or the Watch windows will not update when single stepping. The pin will be modified, but the read request to retrieve the new value is blocked by the freeze and cannot be updated until the next step or run command.

- SFRs Window Display
- SFRs Window Menu
- SFRs Window FAQ

13.23.1 SFRs Window Display

Data is displayed in the following columns.

- Address – SFR hexadecimal address.
- SFR Name – Symbolic name for the SFR.
- Hex – Hexadecimal value, shown in 1-byte blocks.
- Decimal – Value in decimal format.
- Binary – Value in binary format.
- Char – Value in character format.

13.23.2 SFRs Window Menu

Below are the menu items in the Special Function Registers right mouse button menu.

Close

Close this window.

Bitfield Mouseover

Enable/disable bitfield value mouseover. SFR window must be the active window (in focus) for this feature to work.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Go to the specified item of the Go To dialog. (See **Section 14.15 "Go To Dialog"**.)

Output to File

Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of an "Output Range" and tab-delimited option checkbox. Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the SFRs window.

Properties

Set up window properties. See **Section 14.23 "Properties Dialog"**.

13.23.3 SFRs Window FAQ

How do I:

Fill all registers with a value?

Right click in the window and select "Fill Registers" to open a dialog where you may enter fill data.

13.24 TRACE MEMORY WINDOW

The Trace memory window (under the View menu) helps you to monitor much of the processor operation.

Note:	The Trace window is only available when using debuggers that support trace.

The Trace buffer is shared with the logic analyzer. To enable the tracing of all executing program lines to the Trace buffer, check the checkbox *Debugger>Settings*, **Trace** tab, Trace All. To enable the tracing of a select number of executing program lines to the Trace buffer, use the Filter-Out or Filter-In Trace options on right mouse button menus.

If the trace buffer is empty, the window will say "No items to display". Otherwise, the window will display the contents of the trace buffer. Up to 32767 instruction cycles can be displayed in the Trace memory window. Source code may be viewed in the lower portion of the window, if desired. This code is read-only.

- Trace Window Display
- Trace Window Menu
- Trace Window FAQ

13.24.1 Trace Window Display

The Trace memory window may be left open at all times, moved or resized. Data in the trace pane of the window may be edited “in place”.

Trace Pane

The Trace memory window contains trace information for each execution cycle. Depending on the tool selected, you may see some or all of the following:

- Line Number (**Line**) – Cycle's position relative to the trigger or halting point.
- Address (**Addr**) – Address of the instruction being fetched from program memory.
- Opcode (**Op**) – Instruction being fetched.
- Label (**Label**) – Label (if any) associated with the program memory address.
- Instruction (**Instruction**) – Disassembled instruction.
- Source Data Address (**SA**) – Address or symbol of the source data, if applicable.
- Source Data Value (**SD**) – Value of the source data, if applicable.
- Destination Data Address (**DA**) – Address or symbol of the destination data, if applicable.
- Destination Data Value (**DD**) – Value of the destination data, if applicable.
- Time Stamp (**Cycles/Time**) – Time stamp value in cycles or seconds
- External Inputs (**Probe #**) – Value of the external inputs (if any). Not available/applicable for simulator trace.

The approximate trigger cycle will be numbered as cycle 0. All other cycles will be numbered based on this cycle. Cycles that occurred before the trigger point will have a negative cycle number, and cycles that occurred after the trigger point will have a positive number.

Arrows on the side of the trace window allow movement through the trace data lines.

Source Code Pane

Corresponding source code may be viewed in the lower portion of the Trace memory window by selecting “Show Source” from the Trace memory menu. Drag the window bar dividing the trace and source code to resize each portion. The source code may not be edited in this window.

13.24.2 Trace Window Menu

Below are the menu items in the Trace memory right click menu.

Close

Close this window.

Find

Opens the Find dialog. In the Find What field, enter a string of text you want to find, or select text from the drop-down list. You can also select text in the edit window or place the cursor over a word you want to search for, before you open the Find dialog.

In the Find dialog you may select any of the available options and the direction you want to search. Up searches backward from the insertion point, Down searches forward.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Go To

Jump to the specified item:

- Trigger – Jump to the location of the trigger.
- Top – Jump to the top of the window.
- Bottom – Jump to the bottom of the window.
- Go To Trace Line – Go to the trace line specified in the dialog.
- Go To Source Line – Open a File window and go to the source code line corresponding to the selected trace line.

Show Source

Show/hide the source code listing on the bottom of the window. The window bar dividing the trace and source code may be dragged to resize each portion.

Reload

Reload the trace memory window with the contents of the trace buffer.

Reset Time Stamp

Reset the time stamp conditionally on processor Reset, on run or manually. Or, force an immediate reset by selecting Reset Now.

Display Time

Display the time stamp as a cycle count, in elapsed seconds or in engineering format.

Clear Trace Buffer

Clears the contents of the Trace buffer and window.

Symbolic Disassembly

Instead of numeric address for SFRs and symbols, display the names in listing.

Output to File

Export the contents of the trace memory window to a file. Uses a Save As dialog (see **Section 14.11 "File Management Dialog"**), with the addition of cycle and tab information. Enter a "Start" and "End" cycle to write to the file. Also specify if the text is to be tab-delimited.

Print

Print the contents of the trace memory window.

Refresh

Refresh the viewable contents of the window.

Properties

Set up window properties. See **Section 14.23 "Properties Dialog"**.

13.24.3 Trace Window FAQ

How do I:

Set up a simple trace?

In the file (editor) window containing application code, select either "Add Filter-in Trace" or "Add Filter-out Trace" from the right mouse menu. For more information on filter trace, see **Section 16.6 "Working with Debug Features"**.

Set up a more complex trace?

Many debug tools allow you to set up several conditions to define a trace. Consult the documentation for that tool to determine how to set up this type of trace.

13.25 WATCH WINDOW

The Watch window allows you to monitor program symbols while running your program. Up to four different watches may be set up on different tabs of this window.

For more information on using this window, see **Section 8.6 “Watch Window”**.

- Watch Window Display
- Watch Window Menus
- Watch Window FAQ

13.25.1 Watch Window Display

Select any one of four Watch views by clicking on one of the tabs on the bottom of the window, i.e., Watch 1, Watch 2, Watch 3 or Watch 4.

Data is displayed in the following columns:

- Update (Tool-dependent) – Set Data Capture and/or Runtime Watch

  Depending on the debug tool you have selected, you may see this column with the option to set a data capture and/or a runtime watch for the SFR or Symbol. See the debug tool documentation for more details.
- Address – Hexadecimal address of the SFR or Symbol.

  To add an absolute address to a Watch view, double click in the Address column of the first available row and enter a hex address. Click outside the column or hit return to enter the address. Addresses preceded by a “P” specify symbols defined in program memory.
- Symbol Name – Name of the SFR or Symbol.

  To add a Special Function Register to a Watch view, select the SFR from the drop-down list and then click **Add SFR**. To add a symbol to a Watch view, select the symbol from the drop-down list and then click **Add Symbol**.

  Alternatively, double click in the Symbol Name column of the first available row and enter an SFR or symbol name. Click outside the column or hit return to enter the SFR/symbol.

  Enter a colon in front of a symbol to delineate two variables from different files (see **Section 14.32 “Watch File Scope Dialog”**.)
- Value – Current value of the SFR or Symbol.

  To change a value in a Watch view, double click in the Value column and type the new value. Click outside the column or hit return to see the value update.
- Radix Information – Hex, Decimal, Binary, Char

  You may add radix information to the display by right clicking on the column header bar.
- Comment – Enter text comment

  For each symbol entered in the window, you may enter a comment. Useful for documenting bit fields, expressions, etc.

Watch variables may be rearranged using drag-and-drop.

Watch information may be sorted by column. Simply click on the column header to sort data according to that column.

13.25.2 Watch Window Menus

Right clicking in the Watch window will bring up one of the following menus: Standard Menu or Update Menu.

13.25.2.1 STANDARD MENU

Below are the menu items in the Watch window right mouse button menu.

Close

Close this window.

SFR Bit Field Mouseover

Enable/disable bitfield value mouseover. SFR window must be the active window (in focus) for this feature to work.

Find

Find text specified in the Find dialog in this window.

Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift>+<F3> reverses the direction of the last Find.

Add

Add a Watch item to the currently selected Watch tab. See **Section 14.3 "Add Watch Dialog"**.

Delete

Delete the selected Watch item from the currently selected Watch tab.

Save Watch Tab

Save the contents of the currently selected Watch tab to a file.

Load Watch Tab

Load the contents of a file containing previously saved Watch information into the currently selected tab.

Add Watch Tab

Add a Watch tab to the Watch window. You can have as many as 16 Watch tabs.

Rename Watch Tab

Rename the currently selected Watch tab.

Remove Watch Tab

Remove the currently selected Watch tab. You can have no fewer than four Watch tabs.

Import Table

Save Watch data to a table. See **Section 14.29 "Table Setup Dialog"**.

Export Table

Load Watch data from a table. See **Section 14.29 "Table Setup Dialog"**.

Output to File

Write the displayed window contents to a text file.

Print

Print the contents of the window.

Refresh

Refresh the data in this window.

Help

Display help on the Watch window.

Properties

Opens the Watch dialog, so you can set up fonts and colors, as well as other Watch window properties. See **Section 14.33 “Watch/Locals Dialog”**.

Data Capture/Realtime Watch Menus

Depending on your selected debug tool, you may see an “Update” column in the Watch window. Right clicking on a diamond in this field will pop up one of the following menus.

13.25.2.2 UPDATE MENU

This menu will pop up if you click in the region of the “Update” column. The presence of this column depends on which debug tool you have selected. Please see your debug tool documentation for details on the options on this menu.

13.25.3 Watch Window FAQ

How do I:

Add a watch (address, SFR or symbol) to a Watch window?

Select the Watch window you want by clicking on the appropriate button in the bottom of the window.

To add a watch at a specific **address**, click in the Address column of the first available row and enter a hex address.

To add a **special function register** to a Watch view, select the SFR from the list next to the Add SFR button and then click on the button. To add a **symbol** to a Watch view, select the symbol from the list next to the Add Symbol button and then click on the button.

Save the watch contents to a file?

Click the right mouse button in the window to open a menu. Select Save Watch to save the currently selected watch to a file. Or, select Export to save Watch values to a file.

Load a previously saved Watch window?

Click the right mouse button in the window to open a menu. Select Load Watch to load a previously saved watch into the currently selected watch. Or, select Import to load Watch values.

Change the order of watch items?

Simply drag-and-drop items where you want them in the watch list.

Watch a single bit from a register?

Set the variable format to binary. See **Section 14.33 “Watch/Locals Dialog”**.

Watch a 16-bit variable?

Set the variable size to 16 bit. See **Section 14.33 “Watch/Locals Dialog”**.

Watch C pointers and structure members?

See **Section 8.6.7 “C Language Usage – Watch Window”**.

Watch same-named variables in different files?

Add the variable name under the Symbol Name column. Place a ":" in front of the name and hit return to open the Watch File Scope dialog. Select the associated file.

Chapter 14. Dialogs

14.1 INTRODUCTION

MPLAB IDE dialog boxes behave as normal Windows applications. The basic dialogs available in MPLAB IDE are listed below.

> **Note:** Depending on the tools you have installed, tool-specific dialogs may be available. See the documentation for that tool for information about that dialog.

Dialogs covered in this chapter, listed alphabetically, are:

Dialogs	Related Object
About MPLAB IDE Dialog	Help menu
Add Watch Dialog	Watch window
Breakpoints Dialog	Debugger menu*
Build Options Dialog	Debugger menu*
Check for Updates Dialog	Help menu
Configure Channel Dialog	Logic Analyzer window
Configure Bus Dialog	Logic Analyzer window
Export Hex File Dialog	File menu
External Memory Setting Dialog	Configure menu
File Management Dialog (Open, Save As, Add Files)	File, Project menus
Fill Memory/Registers Dialog	Program Memory, EEPROM, File Registers, SFR windows
Find In Files Dialog	Edit menu
Find and Replace Dialogs	Edit menu
Go To Dialog	Edit menu
Help Topics Dialog	Help menu
Import Dialog	File menu
Locate Missing File Dialog	Project window menu
Logic Analyzer Properties Dialog	Logic Analyzer window
New Project Dialog	Project menu
Project-Display Preferences Dialog	Project menu
Project Wizard Dialogs	Project menu
Properties Dialog	most windows
Save Project As Dialog	Project menu
Select Device Dialog	Configure menu
Select Language Toolsuite Dialog	Project menu
Set Language Tool Location Dialog	Project menu
Settings Dialog	Configure menu Debugger, Programmer menu*

Dialogs	Related Object
Table Setup Dialog	Watch, Locals windows
User ID Memory Dialog	Configure menu
Version Control Dialog	Project menu
Watch File Scope Dialog	Watch, Locals windows
Watch/Locals Dialog	Watch, Locals windows
* A tool must be chosen before the related item will appear on the menu.	

14.2 ABOUT MPLAB IDE DIALOG

Select *Help>About MPLAB IDE* to open the About MPLAB IDE dialog. This is an informational dialog.

- Click the link "MPLAB Development Software Webpage" to go to the Microchip MPLAB IDE webpage.
- Click **Copy** to copy the Version Information to the PC clipboard.
- Click **OK** to close the dialog.

Version Information

A list box contains the following information:

- Filename – Name of the component/plug-in
- Version – Version number for that component/plug-in
- MPLAB Certified – Whether or not the component/plug-in is MPLAB Certified. For more information on certification, contact: joe.drzewiecki@microchip.com.

Clicking on an item in the list will reveal the path to that module below the list box.

Trademark Information

This section contains trademark information for Microchip products. In addition, references to other products are discussed.

14.3 ADD WATCH DIALOG

Right click in the Watch window and select **Add** to open the Add Watch dialog. Use this dialog to add a watch item to the currently selected Watch tab. When you have finished adding items, click **Close.**

Related windows/dialogs are:

- **Section 13.25 "Watch Window"**
- **Section 14.29 "Table Setup Dialog"**
- **Section 14.32 "Watch File Scope Dialog"**
- **Section 14.33 "Watch/Locals Dialog"**

14.3.1 Add an SFR

Select a Special Function Register (SFR) name from the pull-down list. Then select a format for the SFR value. Finally click **Add SFR**.

14.3.2 Add a Symbol

Select a symbol name from the pull-down list. Then select a format for the Symbol value. Finally click **Add Symbol**.

14.3.3 Add an Absolute Address or Address Range

First select a type of memory. The range of possible addresses will be displayed in Range. Second, enter a Start Address and either an End Address or the Word Count. (For Word Count, be sure to specify the Word Size.) Finally click **Add Address**.

14.4 BREAKPOINTS DIALOG

Select *Debugger>Breakpoints* to open the Breakpoints dialog. You must select a debug tool before this option is available on the Debugger menu.

Note: There are other ways to set breakpoints. See **Section 8.4 "Breakpoints"**.

The breakpoint dialog below applies to the following debug tools:

- MPLAB SIM simulator
- MPLAB ICE 2000/4000 in-circuit emulators
- MPLAB ICD 2 in-circuit debugger
- PICkit 2 and 3 Debug Express

Control	Description
Break At:	Enter the source code line number or address of a breakpoint. Hit <Enter> to place the breakpoint in the list below.
Program Memory Breakpoints	List of breakpoints set in program memory. These breakpoints may be selected and then acted on by the buttons to the right.
Active Breakpoint Limit	Total number of breakpoints that may be set for selected debug tool.
Available Breakpoints	How many breakpoints are left after subtracting the code breakpoints AND advanced breakpoints (if applicable). **Note:** Some MPLAB IDE commands use a breakpoint to run and stop. If none are available, these commands step-and-compare until they reach their stop location, which may cause them to execute slower. For more information, see **Section 8.4.4.2 "MPLAB IDE Features Impacted When All Breakpoints are Used"**.

Enter a breakpoint

Enter a breakpoint location in the "Break At" box.

- Enter a line number, e.g., 27. You must also enter the source file path and name for this to work, e.g., `c:\project1\program1.asm, 27`.

- Enter a line address (in hex), e.g., `002A`.

You will see the breakpoint you are typing also appear in the "Program Memory Breakpoints" box. Once you have entered a breakpoint, hit return to see it entered in the "Program Memory Breakpoints" box. A checkbox will appear in front of the breakpoint so that you may enable/disable this breakpoint.

Remove a breakpoint

Click on a breakpoint in the "Program Memory Breakpoints" box to highlight it. Then click the **Remove** button.

Remove all breakpoints

Click the **Remove All** button to delete all breakpoints from the "Program Memory Breakpoints" box.

Enable or disable a breakpoint

Click the checkbox in front of the breakpoint to alternately enable/disable it.

Enable or disable all breakpoints

Click either the **Enable All** or **Disable All** button to enable or disable all the breakpoints listing in the "Program Memory Breakpoints" box, respectively.

Fix unresolved breakpoints

If you have set a breakpoint on a line of high-level language code (e.g., C code) that cannot be resolved (into assembly code), you will get a warning message. Also, in the breakpoint dialog, a yellow question mark will appear next to the unresolved breakpoint.

Often optimized code will cause this problem. Rebuild your application without optimizing and try the breakpoint again.

Another solution is to simply move the breakpoint to another line of code that can be resolved.

Save/cancel changes

Click **OK** to save and implement your changes, and close the dialog. Click **Cancel** to close the dialog without saving or implementing your changes.

14.5 BUILD OPTIONS DIALOG

Select *Project>Build Options>Project* to open the Build Options dialog. You must have a project open before this option is selectable on the Project menu.

14.5.1 Directories Tab

Enter or browse for directories and search paths that MPLAB IDE will use when building a project. Directories are one specific path to files, whereas search paths may include several paths. Relative paths (e.g., ..\tmp) will be relative to the source file location.

Note:	These are MPLAB IDE search paths only, e.g., MPASM assembler does not use this information. Please consult other language tool documentation to determine if your tool uses this information.

14.5.1.1 DIRECTORIES AND SEARCH PATHS

Set up the directories and search paths to be used in a project build.

Selection	Description
Show Directories For	Select the directory type for which you wish to assign a path, e.g., select “Library Search Path”, click **New**, and type in or browse to (...) a path for MPLAB IDE to find the project’s library files. The directories available for selection in the list depend on the language toolsuite chosen.
Output Directory	Path to directory containing files that result only from a full build, specifically the link step. This is where the COD, COFF and hex files go, as well as list and map files. These files are affected by *Project>Clean*.
Intermediates Directory	Path to the directory containing intermediate files generated when a project is built. To delete these files, use *Project>Clean*.
Assembler Include Search Path	Path(s) to the directory(ies) containing assembler include files (`.inc`).
Include Search Path	Path(s) to the directory(ies) containing include files (`.h`).
Library Search Path	Path(s) to the directory(ies) containing library files (`.lib`, `.arc`).
Linker Script Search Path	Path(s) to the directory(ies) containing linker script files (`.lkr`, `.gld`).
Buttons	Click on a button to perform an action. Depending on path selection, some buttons may not be available (grayed out).
New	Click to add a new search path in the dialog window list. Type in the path or click on (...) to browse to a path.
Delete	Click on a search path in the dialog window list and then click this button to delete the path from the list.
Up/Down	Click on a search path in the dialog window list and then click either **Up** or **Down** to move the path either up or down in the list.
Suite Defaults	Replaces the current directory and path selection with default values. Defaults are set in *Project>Set Language Tool Locations*.

14.5.1.2 BUILD DIRECTORY POLICY

Select the policy to be used when building a project.

Selection	Description
Assemble/Compile in the source file directory, link in the output directory	Perform assembly/compilation in the directory where the source files are located but place the result of linking these files into the Output directory. Note that these directories may or may not be the same as the project directory.
Assemble/Compile Link in the project directory	Perform all functions in the project directory. This selection is required to “Export Makefile”.

14.5.2 Custom Build Tab

Use this tab to run a command-line batch script before or after a build.

Selection	Description
Pre-Build Setup	Enter a command line to be executed before a build begins.
Post-Build Setup	Enter a command line to be executed after the successful completion of a build.

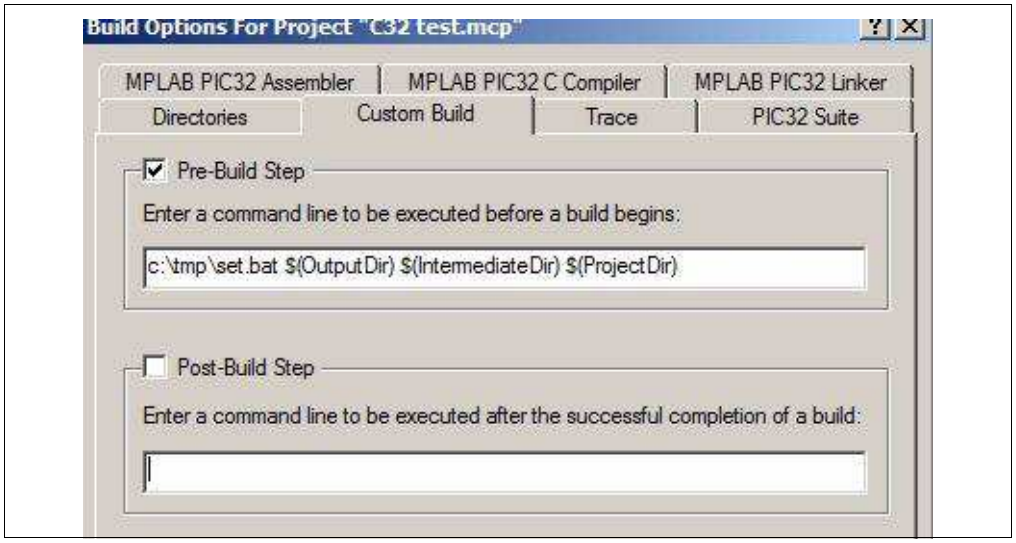
Starting with MPLAB IDE v8.10, there are some variables that you can add to your custom-build command:

- $(OutputDir)
- $(IntermediateDir)
- $(ProjectPath)
- $(ProjectDir)
- $(ProjectName)
- $(TargetPath)
- $(TargetDir)
- $(TargetName)
- $(Device)

In the example below, three variables are used in the Pre-Build Step:

c:\tmp \set bat $(Output Dir) $(Intermediate Dir) $(Project Dir)

FIGURE 14-1: EXAMPLE CUSTOM BUILD INPUT



14.5.3 Trace Tab

This tab is *only* used for the MPLAB REAL ICE in-circuit emulator. Enable and set up trace for many supported devices. However, PIC32MX Instruction trace is enabled under *Debugger>Settings*, **Trace** tab.

14.5.4 Language Tool Setup Tabs

Depending on the language toolsuite selected, there may be other tabs displayed. To select a language toolsuite, use *Project>Select Language Toolsuite*.

Use these tabs to set up language tools in the selected suite. See the language tool documentation (user's guide or on-line help) for more information.

14.6 CHECK FOR UPDATES DIALOG

Selecting *Help>Check for Updates* opens this dialog.

Section	Description
Current Versions	This section lists version information: • Active version of MPLAB® IDE • Installed version(s) of MPLAB IDE • Available update version(s) of MPLAB IDE A summary statement shows whether any updates are viable.
MPLAB Development Software Webpage	Click this link to go to the webpage where the updates are located. If you do choose to update, make sure you close this dialog and any open versions of MPLAB IDE before performing the update.
Update Method	Select a method for installing updates: • Manual – you choose when to update • Weekly – every week the webpage is checked for updates • Monthly – every 4 weeks the webpage is checked for updates
On startup, only show dialog when new Update is available.	To display the dialog on startup only when an update is available, check this checkbox.

14.7 CONFIGURE CHANNEL DIALOG

Select pin and/or bus signals to be used by the Logic Analyzer window (see **Section 13.15 "Logic Analyzer Window"**). Pin signals show changes on individual pins. Bus signals show changes (in Hex) for multiple pins, as for a port. Each signal chosen will be given its own channel in the Logic Analyzer display.

To add signals to a channel:

- Select the type of signal to add from the pull down list: pin, bus or both. Buses can be created by clicking **Configure Bus(es)**.
- Click on a signal in the "Available Signals" column to select it. To select more than one signal at a time, use shift-click (for a continuous group) or <Ctrl> + Click (for a non-continuous group) to select additional signals.
- Click **Add=>** to add the signals to the "Selected Signals" column. The signals are added as one per channel.

To delete signals from a channel:

- Click on a signal in the "Selected Signals" column to select it. To select more than one signal at a time, use shift-click (for a continuous group) or <Ctrl> + Click (for a non-continuous group) to select additional signals.
- Click **Remove<=** to remove the signals to the "Selected Signals" column.

To change the list order of channels:

- Click on a signal/channel in "Selected Signals" to select it.
- Use **Move Up** or **Move Down** to move the selection in the list.

14.8 CONFIGURE BUS DIALOG

Create/remove buses to be used as channels (see **Section 14.7 "Configure Channel Dialog"**) by the Logic Analyzer window (see **Section 13.15 "Logic Analyzer Window"**).

To create/remove a bus:

- Click on **New Bus** to create a new bus. You will be asked to supply a name. The new bus will appear under "Available Bus(es)".
- Click on a bus under "Available Bus(es)" and then click **Delete Bus** to remove the bus from the available list.

To add signals to a bus:

- Click on the bus under "Available Bus(es)" to which you want to add signals.
- Click on a signal in the "Available Signals" column to select it. To select more than one signal at a time, use shift-click (for a continuous group) or <Ctrl> + Click (for a non-continuous group) to select additional signals.
- Click **Add=>** to add the signals to the "Selected Signals" column. The signals are added from the Most Significant bit (MSb) at the top of the list to the Least Significant bit (LSb) at the bottom of the list.

To delete signals from a channel:

- Click on the bus under "Available Bus(es)" from which you want to delete signals.
- Click on a signal in the "Selected Signals" column to select it. To select more than one signal at a time, use shift-click (for a continuous group) or <Ctrl> + Click (for a non-continuous group) to select additional signals.
- Click **Remove<=** to remove the signals to the "Selected Signals" column.

To change the list order of channels:

- Click on a signal/channel in "Selected Signals" to select it.
- Use **Move Up** or **Move Down** to move the selection in the list.

14.9 EXPORT HEX FILE DIALOG

The Export Hex File dialog is available from *File>Export*.

Use Export to save program memory, a section of program memory, EEPROM memory, Configuration bits or user ID to a `.hex file`. There are two formats allowed. The INHX32 format should be used for most purposes. The INHX8S format saves the memory area as two files: one for the even bytes and one for the odd bytes. This is useful where program memory is stored in pairs of external 8-bit wide memory chips.

To export a hex file:

1. Click on the **File Format** tab. Select the Hex file format.
2. Click on the **Memory Areas** tab. Select which memory areas to export and, if applicable, their range.
3. Click **OK**.
4. In the Save As dialog, enter the name and directory location of the file and click **Save**.

14.10 EXTERNAL MEMORY SETTING DIALOG

Some PIC MCU and dsPIC DSC devices support extended program memory through the use of off-chip or external memory. These devices have modes you may select, such as Extended Microcontroller (some on-chip and some off-chip program memory) and Microprocessor (all off-chip program memory).

To enable external memory use in a debug tool, a device that supports this feature must be selected (*Configure>Select Device*). Then you may select *Configure>External Memory*. This will open the External Memory Setting dialog.

- **Use External Memory** – Enable/disable external memory use
- **Address Range** – The **Start** address is always set at the end of internal memory. Enter the **End** address for external memory

Once enabled, the external program memory will appear in the program memory window.

Click **OK** to save your changes and close the dialog. Click **Cancel** to close the dialog without saving your changes.

14.11 FILE MANAGEMENT DIALOG

A file management dialog allows you to manage source and project files. The file management dialogs in MPLAB IDE are:

- **Open** dialog – Open an existing file, project or workspace or import an existing debug file.
- **Save As** dialog – Save a file, project or workspace in a different folder or with a different name.
- **Import** dialog – Import tabular data from a file into a memory window.
- **Export As** dialog – Export tabular data from a memory window into a file.
- **Add New Files to Project** dialog – Insert new source files into your project.
- **Add Files to Project** dialog – Insert existing source files into your project.

Depending on the actual dialog, you may see some of the controls listed in **Section 14.11.1 “File Management Controls”**. The top of the dialog will contain the basic controls shown in **Section 14.11.2 “Navigation Buttons”**.

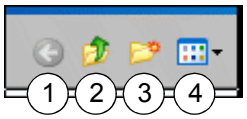
14.11.1 File Management Controls

Control	Type	Description
Look In	Input	Select the folder containing the file/project from the drop-down list. Default is previously viewed directory.
Save In	Output	
Menu of Files	All	A list of files/projects found in the selected folder. The types of files displayed is selected in the “Files of Type”/ “Save As Type” list box. Click on a file here to select it or enter a name in the “File Name” text box. This text box has the following edit features: delete file, rename file and drag-and-drop copy file.
File Name	All	Enter a name for the file/project or select one from the Menu of Files text box.
Files of Type	Input	Select from the list the types of files you wish to view in the Menu of Files text box.
Save As Type	Output	
Jump To	All	Jump to project directories in workspace or recently-accessed directories.
Start Address	Import/Export	Specify the address at which to start the import/export of data.

Control	Type	Description
End Address, Single Column Output	Export	Specify the address at which to stop the export of data. Also check the checkbox for "Single Column Output" or uncheck for multicolumn output for the export file.
Encoding	Save File As	Specify the encoding of the saved file, either ANSI or unicode.
Add File to Project	Save File As	Check to add the saved file to the active project.
Auto, User, System	Add Files to Project	Select the desired radio button to determine which files are added to the active project. **Auto:** MPLAB IDE will determine which files to add. These will usually be all known files in the selected directory that are associated with MPLAB IDE. **User:** You select the files to add. The files are associated with the project via relative (to the project directory) paths. **System:** You select the files to add. The files are associated with the project via absolute (root) paths.

14.11.2 Navigation Buttons

The following navigation buttons are located to the right of the drop-down Look In/Save In box.



1. Go to last folder visited
2. Up one level
3. Create new folder
4. View menu of files as Thumbnails, Tiles, Icons, List or Details.

14.12 FILL MEMORY/REGISTERS DIALOG

To fill memory with a value, right click in one of the windows below and select "Fill Memory".

- **Section 13.19 "Program Memory Window"**
- **Section 13.8 "EEPROM Window"**

To fill registers with a value, right click in one of the windows below and select "Fill Registers".

- **Section 13.10 "File Registers Window"**
- **Section 13.23 "Special Function Registers Window"**

Enter a "Start Address" and an "End Address" as the fill range. If you want to save this range so it always appears in this dialog, check "Retain Address Range".

Enter a value in "Data" to be filled in each address in the specified range or check "Randomize Data". Check "Sequence Start" to fill memory with sequential data.

Select the "Data Radix", either Hexadecimal or Decimal.

Click **Write** to fill memory or register range. Click **Close** to abort.

14.13 FIND IN FILES DIALOG

Selecting *Edit>Find In Files* opens a dialog that allows you to search for information through multiple files, in a directory or in a project. The search results appear in the Output window.

Control	Description
Find what:	Enter a text string of or expression for what you want to find.
Interpret as:	Literal Text - Enter exact text you wish to find under “Find what”.
	Wildcard Expression - Use a wildcard expression in “Find what”. * - Specifies zero or more characters of any value ? - Specifies one character of any value Example: movl? to find all “move literal” instructions.
	POSIX Regular Expression - Use a POSIX regular expression in “Find what”. See below for more information.
Match whole word only	Text in the file window must match the “Find what” text as a whole word, not part of a larger word.
Match case	Text in the file window must match the case of the “Find what” text. E.g., “prog1” would not be a match for “Prog1”.
Look in:	Enter or browse to the directory in which the files are located. To include subdirectories in the search, check “Look in subfolders”.
File types:	Select the type of files to search.

POSIX Regular Expressions

POSIX stands for “Portable Operating System Interface”. Regular Expressions (REs) provide a mechanism to select specific strings from a set of character strings. See *IEEE Std 1003.1-2001*. This standard is heavily influenced by UNIX.

For more information, go to:

- http://en.wikipedia.org/wiki/Regular_expression
- http://www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html

Examples

POSIX	RE	Description
`[[:upper:]]TEMP`	`[A-Z]TEMP`	Find and note all occurrences of strings beginning with letters `A` through `Z` and ending in `TEMP`, such as `ATEMP`, `MTEMP` or `ZTEMP`.
`register[^[:digit:]]`	`register[^0-9]`	Find and note all occurrences of the string `register` that **does not** end in numbers `0` through `9`, such as `registerA`.
`\(`	`\(`	Find and note all occurrences of the left parenthesis.

14.14 FIND AND REPLACE DIALOGS

Use the Find dialog (*Edit>Find*) to find an occurrence of a text string in a file (editor) window. Use the Replace (*Edit>Replace*) dialog to substitute one text string for another in a file (editor) window.

Control	Dialog	Description
Find what:	Find, Replace	Enter a string of text you want to find, or select text from the drop-down list. You can also select text in the file window or place the cursor over a word you want to search for, before you open the dialog.
Replace with:	Replace	Enter a string of text with which you want to replace the find text, or select text from the drop-down list.
Match whole word only	Find, Replace	Text in the file window must match the "Find what" text as a whole word, not part of a larger word.
Match case	Find, Replace	Text in the file window must match the case of the "Find what" text. E.g., "prog1" would not be a match for "Prog1".
Direction	Find, Replace	Up searches backward from the cursor position; Down searches forward.
Find Next	Find, Replace	Highlight the next occurrence of the "Find what" text.
Replace	Replace	Replace the highlighted text with the "Replace with" text.
Replace All	Replace	Replace all occurrences of "Find what" text with "Replace with" text.

14.15 GO TO DIALOG

Select "Go To" from a menu to open this dialog. "Go to" options may vary depending on the type of window with which it is used.

Go to what:	Selection
Line	**Enter a Line Number** – Enter a line number as the "go to" location. To view line numbers in the editor window, set up editor preferences (**Section 16.2.1 "Editor Options Dialog"**).
Address	**Enter a Hex Address** – Enter a hexadecimal address as the "go to" location. The allowable memory address range is specified below the text box.
Label	**Select a Label** – Select a label from the drop-down list as the "go to" location. The list contains all labels used in the current program.
Function	**Select a Function** – Select a C-language function from the drop-down list as the "go to" location. The list contains all functions used in the current program.

14.16 HELP TOPICS DIALOG

Select *Help>Topics* to open the MPLAB IDE Help Topics dialog. Use this dialog to select a help file to display. Help files are arranged according to the type of development tool they describe, i.e., System, Language Tools, Debuggers, Programmers and (Other) Tools. You may only select one help file to open.

Double click on your selection or click to select and then click **OK** to open the help file. Click **Cancel** to close the dialog without opening a help file.

14.17 IMPORT DIALOG

The Import dialog is available from *File>Import*. It is basically an Open dialog (see **Section 14.11 "File Management Dialog"**).

Use Import to load a `.hex` file into memory. The hex file may contain data for program memory, data EEPROM, Configuration bits and ID locations. This is useful if you want to program a device using a previously assembled or compiled project. When you do this, if the associated debug file is in the same directory as the `.hex` file, the symbols and line number information will also be loaded into the current project.

You can also import debug files: `.cof`, `.elf` or `.dwarf`. These contain both the memory information from the associated `.hex` file and the symbol information from when that project was last built. These will be no notification that the import has completed, but the data in the IDE windows will change.

14.18 LOCATE MISSING FILE DIALOG

Right click on a Project window file with the text "(file not found)" and select "Locate Missing File" to open this dialog. See **Section 13.20 "Project Window"**.

Use this dialog to help locate/manage missing project files.

Control	Description
Project Directory	The path to the project that contains the missing file.
Missing file	The name of the missing file.
Use this suggested file	Click the radio button and select a file from the list of suggested files.
Use a file of my own choosing	Click the radio button and type in a file name or browse to a file.
This is a generated file - I expect it to be missing	Label the missing file as generated so MPLAB IDE will not warn you that it is missing.
I don't need this file - please remove it	Remove the file from the project.

14.19 LOGIC ANALYZER PROPERTIES DIALOG

Right clicking in the Logic Analyzer window (**Section 13.15 "Logic Analyzer Window"**) and selecting Properties from the pop-up menu will display a dialog where you may set window properties. The tabs of this dialog, and available options on each tab, are described below. This dialog is similar to the standard windows properties dialog (**Section 14.23 "Properties Dialog"**).

Colors/Fonts

- Colors – Select colors for the display background, grid lines, signal lines, trigger line and delta cursor.
- Fonts – For the axes, select a color and type of font.

General

- Grid Lines – Set the type of grid lines you want, i.e., vertical only, horizontal only, both vertical and horizontal or none.

14.20 NEW PROJECT DIALOG

Enter the name and location of a new project (*Project>New*).

- Name – Enter a name for the new project.
- Directory – Enter a directory for the new project. You may do this in one of two ways:
  - Type in the path to an existing directory or type in the path to a new directory, in which case you will be prompted to create the directory when you click **OK**.
  - Click Browse to browse to an existing directory or to one level above where you wish to place a new directory. Click **OK** in the Browse for Folder dialog. Complete the path if you are creating a new directory and then click **OK**. You will be prompted to create the directory if it does not exist.

14.21 PROJECT-DISPLAY PREFERENCES DIALOG

Right click in an empty area of the Project Window and select Preferences from the pop-up menu to display the Project-Display Preferences dialog. Use this dialog to set preferences for the Project window.

Control	Description
Display Project Nodes As	Select how you want project file types displayed on the project tree, either as simply names, or names with paths.
Display File Nodes As	Select how you want project files displayed on the project tree, either as simply file names, file names with paths or as specified internally in the file.
Refresh Version Controlled Files	If your project is set up to use version controlled files (*Project>Select Version Control System*), select when you want the project window refreshed to reflect the version control status. Refresh means asking the version control system, "Who has this file checked out?" for each and every file. This can be very slow, especially if you have many files in your project. If you disable both "Refresh on application focus" and "Auto refresh", you will still get refreshes after performing version control operations, and in response to the "Refresh" command on the Project window context menus.

14.22 PROJECT WIZARD DIALOGS

Select *Project>Project Wizard* to launch a sequence of dialogs for setting up your project. For more information, refer to **Section 6.2 "Using the Project Wizard"**.

14.23 PROPERTIES DIALOG

Right clicking in a window and selecting Properties from the pop-up menu will display a dialog where you may set window properties such as font type and color, background color and column settings. Depending on the window, this dialog will may have different tabs. The generic tabs used are described below.

14.23.1 Column Settings Tab

Column heading used in the window display will be listed in a list box.

To Show/Hide a Column

- Check/uncheck the checkbox next to an item to display/hide the column.
- Click on an item to select it. Then use the **Show** or **Hide** buttons to display or hide the column.

To Reorder Columns

- Click on an item to select it. Then use the **Move Up** or **Move Down** buttons to display or hide the column.

To View Column Width

- Click on an item to select it. The column width will be shown in “Selected column width in pixels”.

To Change Column Width

- Make the window active.
- Move the cursor over the line between columns until it changes to the resize cursor
- Click and drag to widen or narrow the column.

To Restore Default Settings

- Click **Defaults**.

14.23.2 General Tab

Set up the font and change color for all debug windows, i.e., what you set up here will globally determine settings for all debug (View menu) windows.

Fonts

Click **Select Font** to open the standard Windows Font dialog where, for a selected script (e.g., Western), you may set up font type, font style and font size. To choose only non-proportional fonts in the Font dialog, check the checkbox for “Show Fixed Pitch Only”.

Colors

Click **Change Color** to open the standard Windows Color dialog where you may set the change color. Change color is the color the text is displayed in when a change to that text occurs, e.g., when a `movwf PORTB` changes the value of PORTB from 00 to 55 in the SFR window, the 55 will be displayed in the change color.

14.24 SAVE PROJECT AS DIALOG

Select *Project>Save Project As* to open the Save Project As dialog.

When a project is saved to a new directory, all files in the project directory and its subdirectories will be copied to the new directory along with the new project and workspace files. Any files which exist outside of the project tree are considered to be shared files and are not copied. The context menu on the Project window will change to reflect the new location of the project.

To make projects (and related workspaces) portable, MPLAB IDE stores files in the project directory and its subdirectories with relative paths and files outside of the project directory with full paths.

The Save Project As dialog is functionally the same as a standard Save As dialog (see **Section 14.11 "File Management Dialog"**).

14.25 SELECT DEVICE DIALOG

To choose a device for development, select *Configure>Select Device* to open the Select Device dialog.

CAUTION

Some devices operate at 5V and some devices operate at 3.3V. Selecting a 5V device in MPLAB IDE and then using an actual 3.3V device could result in device damage when using some hardware tools.

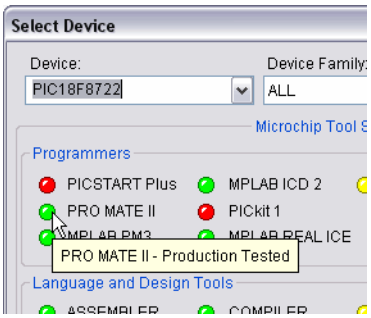
Control	Description
Device	All devices currently supported by this version of MPLAB® IDE will be listed here. Select from the list or start typing in a device name.
Device Family	To narrow the list of devices under "Device", select a grouping from "Device Family". Selecting "ALL" will cause all supported devices to be listed.
Microchip Tool Support Programmers Language and Design Tools Debuggers	Once you have selected a device, these sections will reflect the available tool support for that device from Microchip Technology. The level of support is shown via buttons (see below). For MPLAB ICE 2000 and 4000, supported processor modules are shown. For MPLAB ICD 2 and 3, MPLAB REAL ICE emulator, and PICkit 2 and 3, header support is also shown.
OK	Click **OK** to save your changes and close the dialog.
Cancel	Click **Cancel** to close the dialog without saving your changes.

Tool Support Level

The level of tool support for the selected device is shown as colored “lights”.

Light	Color	Support
	Green	Production tested. For assemblers/compilers, numbers for the first supported production version are shown also.
	Yellow	Not production tested (Beta support available).
	Red	No support currently available.

You can also mouse over a light to pop up its support level.



Tool Selection

To choose a programmer, select from the list of *Programmer>Select Programmer*.

To choose a language tool, select *Project>Select Language Toolsuite*. To set up a language toolsuite, select *Project>Set Language Tool Locations*.

To choose a design tool, select from the Tools menu.

To choose a debug tool, select from the list of *Debugger>Select Tool*.

14.26 SELECT LANGUAGE TOOLSUITE DIALOG

Select *Project>Select Language Toolsuite* to open the Select Language Toolsuite dialog. You must have a project open before this option is selectable on the Project menu.

Use this dialog to select the suite of language tools you will use in your project. See documentation for the language tools to ensure that they support the device you will be using.

- **Active Toolsuite** – Select the toolsuite you will use.
- **Toolsuite Contents** – View the language tools associated with the toolsuite selected above. If these are not the tools you wanted, choose another toolsuite. Click on a language tool to see its location.
- **Location** – Change the path or file, enter new path or file information or **Browse** for the executable file of the language tool highlighted in the above list box. Check the “Store tool locations in project” checkbox to use these tool locations, instead of the defaults (**Section 14.27 “Set Language Tool Location Dialog”**), in this project.

- **Store tool locations in project** – Store toolsuite information with the project. Therefore the project will always use this toolsuite. This is checked by default for starter kit installations, as they are set up to work with the language toolsuite they come with.

A red "X" opposite a tool indicates that it has not been installed or that MPLAB IDE does not know where to find the tool.

14.27 SET LANGUAGE TOOL LOCATION DIALOG

Select *Project>Set Language Tool Locations* to open the Set Language Tool Location dialog.

Use this dialog to set the path to individual language tool executables within a toolsuite.

- **Registered Tools** – Find the toolsuite you will be using (e.g., Microchip MPASM Toolsuite). Click on the "+" to expand.
  - Click on the "+" next to "Executables" to expand. Click on a tool to see its currently assigned path in "Location".
  - Click on the "+" next to "Default Search Paths and Directories" to expand. Click on a path to see its current assigned in "Location".
- **Location** – Change the path or file, enter new path or file information or **Browse** for the path or file.

14.28 SETTINGS DIALOG

Select *Configure>Settings* to open the Settings dialog and set up general MPLAB IDE functions. Click **OK** to save your changes and close the dialog. Click **Cancel** to close the dialog without saving your changes.

Note:	Once you have selected a debugger or programmer, a tool-specific Settings dialog may be selected from the respective menu. See tool documentation for more on this dialog.

- Workspace Tab – set up workspace options
- Debugger Tab – set up debug options
- Program Loading Tab – set up clear/don't clear memory on program load
- Hot Keys Tab – set up hot keys for various IDE operations
- Other Tab – set up miscellaneous options
- Projects Tab – set up project options
- External Editor Tab – set up an external editor, i.e., a non-MPLAB Editor.

14.28.1 Workspace Tab

Select *Configure>Settings* and click the **Workspace** tab to set up Workspace options.

A **workspace** contains information on the selected device, debug tool and/or programmer, open windows and their location and other IDE configuration settings.

Control	Description
Automatically save workspace upon closing	To automatically save the contents of an active workspace when closing MPLAB® IDE, select Yes. To not save the contents, select No. To be prompted to save the contents, select Prompt. Saving a workspace also saves project information.
Reload last workspace at startup	If checked, reload the last workspace setting on startup of MPLAB IDE. This will also reload the last open project.
Recent files list contains	Specify the number of recent files to include on the list found under *File>Recent Files*.
Recent workspaces list contains	Specify the number of recent workspaces to include on the list found under *File>Recent Workspaces*.
Always show full path in recent file and workspace lists	If checked, show complete path to recent project files.

14.28.2 Debugger Tab

Select *Configure>Settings* and click the **Debugger** tab to set up debug options.

Control	Description
Automatically save files before running	Make sure all project files are saved before running the selected debugger.
Remove breakpoints upon importing a file	Clear all breakpoints upon importing a file, i.e., adding a debug/hex file using *File>Import*.
Reset device to the beginning of main function	Change the function of Reset from processor reset to reset then run to the `main()` function. **Note:** This does not change the function of reset in the build-then-reset operation. This function uses breakpoint resources. If you have used all available breakpoints, you will have to check the “Auto step” checkbox below for this option to function as expected.
Auto step to the target address when no breakpoints are available	Use auto step, instead of breakpoint resources, to enable several MPLAB® IDE features. For more information, see **Section 8.4.4.2 “MPLAB IDE Features Impacted When All Breakpoints are Used”**.
Stepping Behavior	
Track debugger location in the source code	If checked, forces focus on source window while stepping. If unchecked, another window (e.g., Watch) will stay in focus while stepping.
Browse for source if file is not found	If source file cannot be found, search for it using paths known to MPLAB IDE.
Show disassembly if source is unavailable	If source file cannot be found, open disassembly window.

14.28.3 Program Loading Tab

Select *Configure>Settings* and click the **Program Loading** tab to set up whether memory is cleared or uncleared on program loading. A program load means a build, make or import.

When to Clear Memory

Control	Description
Clear memory before building a project	Clear all memory before a project is built.
Clear memory after successfully building a project	Clear all memory only after a project has built successfully (no errors).

What Memory is to be Cleared

Control	Description
Clear program memory upon loading a program	Clear/Don't clear program memory on program load.Uncheck when using concurrent projects.
Clear configuration bits upon loading a program	Clear/Don't clear Configuration bit values on program load.
Clear EE data upon loading a program	Clear/Don't clear EEPROM data memory on program load.
Clear user ID upon loading a program	Clear/Don't clear the user ID value on program load.

14.28.4 Hot Keys Tab

Select *Configure>Settings* and click the **Hot Keys** tab to set up hot key functions for various MPLAB IDE operations. Hot keys or shortcut keys, are keyboard shortcuts to MPLAB IDE menu or toolbar commands.

Control	Description
Hot Key mapping scheme	Use the default settings, or save your own using **Save As**. A file with a `.hot` extension will be created with your custom settings.
Command	Select the command for which you wish to assign a hot key, or to view the currently selected hot key for that command.
Hot Key combination	Enter the hot key for the selected command or view the currently selected hot key for the selected command.If you hit **Delete** or **Backspace**, this field will be cleared and display "None". When you enter a hot key, the key value will then be displayed here. **Note:** You must hit the actual key(s) you wish to use, not type in the name/function. E.g., To set a hot key combination of **Ctrl + F8** for a command, hold down the **Ctrl** key and then hit the **F8** key.
Hot Key currently in use by	If the hot key you have chosen is already in use, its corresponding command will be displayed here.

14.28.4.1 VALID HOT KEYS

The following keys are not valid hot keys:

- Enter
- Backspace
- Delete
- Tab
- Any alphanumeric keys or shift of above keys

If you select a key that is not valid, Hot Key combination will reflect the invalid selection by displaying None.

14.28.4.2 MULTIPLE ASSIGNMENTS OF HOT KEYS

It is possible for hot keys to be assigned to multiple commands. If a conflict does exist, the hot keys will be applied to the first 'loaded' command encountered in the table.

E.g., Hot keys are assigned to an MPLAB ICD 2 debugger command and an MPLAB PM3 programmer command. However, MPLAB ICD 2 is not enabled when the keys are assigned, but MPLAB PM3 is. Therefore, only the MPLAB PM3 command would be applied. If both MPLAB ICD 2 and MPLAB PM3 are enabled, only the command first encountered in the hot key table will be applied.

Note: Assigning a hot key to a command does not remove any prior command assignments of that hot key.

14.28.5 Other Tab

Select *Configure>Settings* and click the **Other** tab to set up miscellaneous options.

Control	Description
Clear configuration bits when clearing memory	If checked, clear all Configuration bit settings when device memory is cleared.
Automatically reload files that were modified outside of the IDE	Currently used for MPLAB REAL ICE in-circuit emulator trace. Disables the warning message produced when `_LOG` or `_TRACE` IDs are generated and placed in code. See the emulator help file for more information.

14.28.6 Projects Tab

Select *Configure>Settings* and click the **Projects** tab to set up project options.

A **project** contains the files needed to build an application (source code, linker script files, etc.) along with their associations to various build tools and build options.

Control	Description
Close open source files on project close	When a project is closed, close all related open source files.
Clear output window before build	Clear the contents of the output window and then build.
Save project before build	Save the active project setup before building.
Save files before build	To save all active project files before building, select Yes. To not save the files, select No. To be prompted to save the files, select Prompt.
Halt build on first failure	Halt build when the first error occurs. (Many errors may be reported for one source file that fails.)
Use one-to-one project-workspace model	Allow only one project in a workspace. For more on single and multiple project workspaces, see **Chapter 6. "Projects and Workspaces"**.

14.28.7 External Editor Tab

Select *Configure>Settings* and click the **External Editor** tab to set up a editor other than MPLAB Editor.

Control	Description
Use External Editor	If checked, use the external editor specified in the path below.
External Editor	Enter or Browse to the external editor executable.

The External Editor will be used for the "Edit" command when you right click on a file in the Project window, **Files** tab.

However, the MPLAB Editor will still be used for the following: *File>Open*, *File>New*, Recent File, all debugging that opens source files and all "GoTo" and "Enable Tag Locators" commands (Project window, **Symbols** tab.)

14.29 TABLE SETUP DIALOG

Right click in a window and select "Import Table" or "Export Table" to open the Table Setup dialog. Import or Export data from or to a table file. Export SFR/Symbol/Variable values to a file and then import them back as needed.

Control	Function	Description
Start Address	Import/Export	Enter start address for type of memory selected below. Defaults to address of selected item in Watch window.
End Address	Export	Enter end address for type of memory selected below. Defaults to address of selected item in Watch window.
Memory	Import/Export	Choose type of memory to import/export. Defaults to full range when clicked.
Symbol Start Address	Import/Export	The symbolic representation of selected start address, if applicable.
Single Column Output	Export	Output written in a single column.
Type Formatted	Import/Export	Import use: Take a spreadsheet column and import it to an array of floats. Export use: Move floating point/decimal data into spreadsheet or graphing programs.

Related windows/dialogs are:

- **Section 13.14 "Locals Window"**
- **Section 13.25 "Watch Window"**
- **Section 14.3 "Add Watch Dialog"**
- **Section 14.33 "Watch/Locals Dialog"**
- **Section 14.32 "Watch File Scope Dialog"**

14.30 USER ID MEMORY DIALOG

Some devices have memory locations where you can store checksum or other code identification (ID) numbers. These locations are readable and writable during program/verify. Depending on the device, they also may be accessible during normal execution through the `TBLRD` and `TBLWT` instructions.

Select *Configure>ID Memory* to open the User ID Memory dialog.

Control	Description
User ID	Enter a user ID. Consult your device programming specification to determine what values may be entered here. For most devices, this sets the low nibble of the device ID word; the high nibble is set to '0'. The high nibble can only be written to programmatically, such as by using Table Writes in the case of PIC18 MCU devices.
Use Unprotected Checksum	Use the unprotected (i.e., code-protect off) checksum as the user ID for devices that support unprotected checksum. See the programming specification for your device to see how this checksum is calculated. **Note:** You must click **OK** and then reopen this dialog to see the unprotected checksum as the user ID. Also, this value may change after any make or build.

14.31 VERSION CONTROL DIALOG

Select *Project>Version Control* to open the Version Control dialog. Set up your MPLAB IDE Project to use a version control system (VCS).

- Version Control System – Select an available version control system from the drop-down list.
- System – Set up the necessary parameters to use your selected VCS, if required.
- For Project "*project*.mcp" – Set up the necessary project information for project *project*.mcp.

For more on version control systems, see **Section 6.6 "Using A Version Control System (VCS)"**.

CVS

System	Description
CVS Executable	Enter or browse to the directory containing the CVS executable files, e.g., `C:\cvsnt\cvs.exe`.
Remember user name and password*	Check to automatically access the CVS system after entering your user name and password the first time.
For Project "*project*.mcp"	**Description**
Host	Enter the host name or IP address of the machine that is the CVS server. This should match the host name or IP address given to the `cvs login` command.
Port	Enter the port number for the CVS service on the CVS server. You can leave this blank if the server is using the default port value of 2401.
Root	Since a CVS server can host multiple roots, provide the name of the root with a leading forward slash.
Module	Enter either a real CVS module or a top-level directory within the given CVS root.

* The first time you use CVS with MPLAB IDE, you must log into CVS outside of MPLAB IDE, i.e.;

```
$ cvs -d :pserver:MyUserName@ServerHostName:/CVSRoot login
Logging in to :pserver:MyUserName@ServerHostName:2401:/CVSRoot
CVS password: ********
```

Once you have done this, CVS will work with MPLAB IDE.

PVCS

System	Description
EXE directory	Enter or browse to the directory containing the PVSC executable files.
Java directory	Enter or browse to the directory containing PVSC java scripts.
Remember user name and password	Check to automatically access the PVSC system after entering your user name and password the first time.
For Project "*project*.mcp"	**Description**
Database directory	Enter or browse to the directory containing the PVSC database, from which the project files will come.
Archive directory	Enter or browse to the directory containing the PVSC archive, to which project files may be saved.

Microsoft Visual Source Safe

For Project "*project*.mcp"	Description
SRCSAFE.INI file	Specify the path to and name of the Visual Source Safe initialization file for the database you will be using, e.g., `C:\VSS\VSS_6.0\win32\scrsafe.ini`. Click **Browse** to locate the file.
VSS project path	Specify the path within Visual Source Safe to the project files, e.g., `$/Project Files/Project2`.

Subversion

System	Description
SVN Executable	Enter or browse to the directory containing the Subversion executable files, e.g., `C:\svn-win32\bin\svn.exe`.
Remember user name and password	Check to automatically access the SVN system after entering your user name and password the first time.

14.32 WATCH FILE SCOPE DIALOG

Select the file that relates to a variable in a Watch window.

A variable with the same name may exist in more than one file. MPLAB IDE will recognize 2 static variables with same name in different files.

Add the variable to the Watch window by selecting it from the "Add Symbol" list or typing it in to the "Symbol Name" column. Once the variable is in the Watch window, double click on it and add a colon (:) in front of the variable name and hit return. This will bring up the Watch File Scope dialog.

Select the file that is associated with the variable you want to watch. A **Browse** button is provided for selecting files not built.

- The filename information comes from either CTags or Project Manager.
- The Debug .COF/.ELF file must be built to resolve the *filename:variable name* to an address.

Click **OK** when you have selected the file.

The Symbol Name will be displayed as *filename:variable name* with no path. The path may be seen in a tooltip.

Related windows/dialogs are:

- **Section 13.14 "Locals Window"**
- **Section 13.25 "Watch Window"**
- **Section 14.3 "Add Watch Dialog"**
- **Section 14.29 "Table Setup Dialog"**
- **Section 14.33 "Watch/Locals Dialog"**

14.33 WATCH/LOCALS DIALOG

Set up the Watch/Locals window by right clicking in the window and selecting Properties. The Watch/Locals dialog has the following tabs:

- Watch Properties Tab
- Preferences Tab
- General Tab (See **Section 14.23.2 "General Tab"**.)

Related windows/dialogs are:

- **Section 13.14 "Locals Window"**
- **Section 13.25 "Watch Window"**
- **Section 14.3 "Add Watch Dialog"**
- **Section 14.29 "Table Setup Dialog"**
- **Section 14.32 "Watch File Scope Dialog"**

14.33.1 Watch Properties Tab

This tab is used to set the properties for the specific symbols in the window.

Control	Description
Symbol	Select the watch symbol for which to set properties. Choose from the currently selected and displayed symbols in the window.
Size	Select the bit size of the selected watch symbol value. Choose from 8, 16, 24, 32, 40 or 64 bits, depending on device.
Format	Select the format of the selected watch symbol value. Choose from: - Hex(adecimal) - IEEE Float - Binary - Single Bit - Decimal - dsPIC Integer - ASCII - dsPIC Fractional - MCHP Float - Double **Note:** With single bit, no information about which bit is being displayed will be shown in the Symbol Name.
Byte Order	View or change the byte order of the selected watch symbol.
Memory	View the type of memory for the selected watch symbol.

14.33.2 Preferences Tab

This tab is used to select default settings for new symbols added to the window.

Control	Description
Float Format	Select a floating-point format. • Use COFF float type – Use the information in the COFF file. **This is the default.** • IEEE 754 32-bit – Use with the PIC18 MCU compilers (v2.40 and later), the 16-bit compiler and HI-TECH 'C' with -D32x. • IEEE Modified 24-bit – Use with HI-TECH 'C' compilers with -D24x. • Microchip High:Low – Use with the PIC18 MCU compiler before v2.40. • Microchip Low:High – Use with CCS compilers. The float value selected will also be used to format the editor mouse-overs.
Default Type Format	Select a default type format. This format is used when no type can be specified. Choose from Hex(adecimal), Binary, Decimal or Char. Also choose the byte order (High:Low or Low:High). The format selected will be used for SFRs, absolute address and ASM symbols.
Expand SFR Bitfields	To see special function register bitfields, check this checkbox.

Chapter 15. Operational Reference

15.1 INTRODUCTION

Reference information about the operation of MPLAB IDE is discussed here.

- Command-Line Options
- Keyboard Shortcuts
- Files Used by MPLAB IDE
- Saved Information
- File Locations

15.2 COMMAND-LINE OPTIONS

MPLAB IDE can be invoked through the command-line interface as follows:

`mplab [`*`file`*`] [/`*`option`*`]`

`file` = `workspace.mcw`

Open the workspace workspace.mcw in MPLAB IDE. Any projects contained in the workspace will be opened also.

`option` = `nosplash`

Do not display splash screen on program launch.

15.3 KEYBOARD SHORTCUTS

The following keyboard shortcuts are available for MPLAB IDE. To change or set shortcut key assignments, use *Configure>Settings*, **Hot Keys** tab.

15.3.1 File and Edit Menus

Information on shortcut keys for these menus may be found in the MPLAB Editor documentation (MPLAB Editor section of the MPLAB IDE User's Guide or MPLAB Editor Help).

15.3.2 Debugger Menu

Keystroke	Result	Keystroke	Result
F2	Manage breakpoints	F9	Run
Ctrl + 1	Toggle breakpoint	F10	Make
F5	Halt	Ctrl + F10	Build All
F6	Processor Reset	Alt + F10	Quickbuild
F7	Step Into	Ctrl + 2	Run to Cursor
F8	Step Over	Ctrl + Shift + 2	Set PC to Cursor

15.4 FILES USED BY MPLAB IDE

MPLAB IDE generates several files and uses the generated files of its supported tools. The default extensions of these files are listed in Table 15-1.

TABLE 15-1: MPLAB® IDE DEFAULT EXTENSIONS

Extension	Definition
a	Archive (library) file – Microchip 16-bit archiver/librarian
asm	Assembly language source file – MPASM™ assembler
c	C source file – MPLAB® C Compilers
chm	Compiled HTML Help file
cod	File containing symbolic information and object code – MPASM assembler
cof	File containing symbolic information and object code – MPLINK™ linker
elf	File containing symbolic information and object code – Microchip 16-bit linker
err	Error file – assembler/compiler
evt	Event file – MPLAB ICE 2000
exe	Program/utility file
fsti	File stimulus file – MPLAB SIM for PIC17 MCUs
gld	Linker script file – Microchip 16-bit linker
h	C include file – MPLAB C Compilers
hex	Machine code in hex(adecimal) format file **Note:** Not all hex files are the same. Depending on how they are generated, one hex file's content may differ from another, i.e., a hex file generated from a project may differ from a hex file generated from a *File>Export*.
inc	Assembly language include file – Microchip assemblers
lib	Library file – MPLIB™ librarian
lkr	Linker script file – MPLINK linker
lst	Absolute listing file – assembler/compiler
map	Map file – linker
mch	Exported data file
mcp	Project information file
mps	Build state file for Make
mcw	Workspace information file
o	Object file – assembler/compiler
psti	Pin stimulus file – MPLAB SIM for PIC17 MCUs
rsti	Register stimulus file – MPLAB SIM for PIC17 MCUs
ssti	Synchronous stimulus file – MPLAB SIM for PIC17 MCUs
s	Assembly language source file – Microchip 16-bit assembler
sbs	SCL generator workbook file – MPLAB SIM
scl	SCL file from SCL generator – MPLAB SIM
stc	Stimulus scenario file – MPLAB SIM
trc	Trace save file
trg	Trigger file – MPLAB ICE 2000
xrf	Cross-reference file – MPASM assembler

15.5 SAVED INFORMATION

Information concerning your setup of MPLAB IDE is saved as follows.

Workspace (MCW File)

A workspace contains the following information:

- Selected device, debug tool and/or programmer.
- Debug tool/programmer settings information.
- *Configure>Settings*, **Program Loading** tab information.
- Configuration bits settings.
- Open IDE windows and their location.
- Other IDE system settings.

Project (MCP File)

A project contains the following information:

- The group of files needed to build an application.
- File associations to various build tools.
- Build options.

Registry

The following information is saved in the registry file of the Windows OS:

- Language tool names and installation locations.
- Most items on the *Configure>Settings*, **Workspace** tab.
- All items on the *Configure>Settings*, **Projects** tab.
- Information on items that may be made visible/invisible in windows, e.g., columns, split windows, etc.

INI Files

The following information is saved in initialization (`.ini`) files:

- Individual window docking information in the `mplab.ini` file.
- Editor settings in the `mpeditor.ini` file.

Window Sets

The following information is saved as a window set:

- Position information of all open windows and toolbars on the desktop.

> **Note:** Tool-specific windows and/or toolbars that are not open when a Window Set is applied will not automatically be opened. However, when they are manually opened, the position information will be applied.

For more on Window Sets, see **Section 12.2.9 "Window"**.

Hex Files

The following information is saved in a Hex file, depending on selections made under *File>Export*:

- Hex format type INHX32 or INHX8S.
- Program memory, calibration memory, EEPROM, configuration bits, user ID.

15.6 FILE LOCATIONS

This section describes the locations of files associated with MPLAB IDE operation.

- Installed Files
- Output Data Files

15.6.1 Installed Files

MPLAB IDE, and any support files selected during installation, will be installed as follows:

Default main directory: **C:\Program Files\Microchip**

Subdirectory structure:

MpAM

Application Maestro.

MPASM Suite

MPASM Assembler, MPLINK Object Linker and MPLIB Object Librarian executables and help files, support executables, assembler include (INC) files.

- Example – Example files.
- LKR – Linker scripts.
- Template – Template files to aid in code development.
  - Code – Absolute code templates (MPASM assembler used only.)
  - Object – Relocatable code templates (MPASM assembler used with MPLINK object linker.)

MPLAB ASM30 Suite

MPLAB ASM30 and MPLAB LINK30/LIB30 help files.

- bin – MPLAB ASM30, MPLAB LINK30, MPLAB LIB30 and other 16-bit support executables.
- lib – Start-up libraries (crt0, crt1).
- Support – Linker scripts (gld), include files (inc), and templates.

MPLAB IDE

Main MPLAB IDE directory. Contents dependent on selections made when installing.

- Core – MPLAB IDE and MPLAB Editor executables and help files, support files, and legacy third-party MTC files.
- Device – Device files, i.e., files containing coded information about each device supported by MPLAB IDE.
- *Tool-Specific Directories* – Several tool-specific directories containing combinations of tool help files, support files, firmware, and drivers.
- Programmer Utilities – Command-line programmer support, includes Visual PROCMD, PROCMD and PM3CMD.
- Readmes – Readme files for all tools.
- Tools – Plug-in tools for use with MPLAB IDE, such as the Data Monitor and Control Interface (DMCI).
- Utilities – Utilities for MPLAB IDE, including the USB driver preinstaller and cleaner, MPLAB ICE 4000 firmware updater (automatically run by MPLAB IDE if detected out-of-date), and the documentation viewer (*Help>Version Notes*).

MPLAB IDE Common

Initialization files, MPLAB IDE version switching program (MPSwitch.exe) and Microchip language toolsuite information (HTML).

Third Party

Third party files.

15.6.2 Output Data Files

MPLAB IDE and Tool output data files will be placed in the following locations:

For Windows XP and Vista:

`C:\Documents and Settings\`*`username`*`\Application Data\Microchip`

For older Windows operating systems:

`C:\Program Files\Microchip\MPLAB IDE Common`

NOTES:

Part 5 – MPLAB Editor

NOTES:

Chapter 16. Using the Editor

16.1 INTRODUCTION

The MPLAB Editor is an integrated part of the MPLAB IDE Integrated Development Environment. The editor is always available when MPLAB IDE is running.

The MPLAB IDE and MPLAB Editor are designed to provide developers with an easy and quick method to develop and debug firmware for Microchip Technology's PIC microcontroller (MCU) and dsPIC digital signal controller (DSC) product families. For more information on these devices, visit the Microchip web site.

The editor functions in a File window of MPLAB IDE. Menu items (on File and Edit menus and a right mouse menu) are available for editor control.

The MPLAB Editor allows you to use the following features for general text editing.

- **Inserting, Selecting and Deleting Text**

  You can enter text in either insert or strike-over mode. MPLAB Editor shows the mode as either "INS" or "OVR" on the status bar. The text you enter may contain any combination of the standard alpha-numeric and punctuation characters. You can enter additional characters with special keys and\or key combinations.

  Text selection features allow you to select a character, word, entire line or other blocks of text. You can copy or delete any selected text. MPLAB Editor's built-in find and replace feature allows you to search for and replace text.

- **Indenting Text and Removing Indentation**

  You can change the indentation level of one or more lines of text.

  You can specify whether new lines are automatically indented.

- **Delimiting Source Code Blocks**

  You can find matching brackets, such as braces and parentheses, which often delimit sections of text or program sources. This also works on C preprocessor # blocks.

- **Undoing Edit Actions**

  You can reverse edit actions with the Undo command, and do them over again with the Redo command.

16.2 CONFIGURING THE EDITOR

The editor may be configured using two dialogs:

- Editor Options Dialog
- Editor Color Customization Dialog

16.2.1 Editor Options Dialog

You can set editor properties in the Editor Options dialog. Select *Edit>Properties* or right click in a file (editor) window and selecting Properties.

- General Tab
- ASM/C/BAS File Type Tab
- ToolTips Tab
- Text Tab
- Other Tab

16.2.1.1 GENERAL TAB

Set up general properties using the **General** tab on the Editor Options dialog.

TABLE 16-1: GENERAL TAB CONTROLS

Option	Description
Use Tabbed Window	When checked, displays files within a single window, instead of separate windows for each file. **Note:** You must restart MPLAB IDE for any change to this option to take effect.
Enable Tag Locators	For C code, enable "GoTo Locator" functionality. See **Section 13.20.4 "Project Window Menus – Symbols Tab"**. Also needed to enable Autocomplete feature (**Section 16.2.1.3 "ToolTips Tab"**).
Find Wrap MessageBox	When checked, a message box is displayed when the find wraps top or bottom, in addition to the current message of "Find Wrapped" in the status bar position panel.
Protect Read Only Files	When checked, read-only files cannot be edited. When unchecked, you can edit the file and save the changes to another file.
Enable Color Printing	When checked, print-out will be in color. This is useful for preserving syntax information.
Alternates <Home> key action	When checked, alternates between the beginning of text and beginning of line. When unchecked, always goes to the beginning of line.

16.2.1.2 *ASM/C/BAS* FILE TYPE TAB

ASM/C/BAS signifies the specific programming language used:

- ASM - assembly
- C - C code
- BAS - basic

Set up file properties using the **File Type** tab on the Editor Options dialog.

TABLE 16-2: FILE TYPE TAB CONTROLS

Option	Description
Auto Indent	When checked, a new line is automatically indented by the same number of spaces and/or tabs as the line above it.
Auto Indent with Brace Placement	When checked, increase line indent when an opening brace is encountered and decrease line indent after a closing brace is encountered. Exception: If opening/closing braces are on the same line. Only valid for C code.
Line Numbers	When checked, line numbers are visible in the display.
Double click Toggles Breakpoint	When checked, double clicking on a line will alternately enable/disable the breakpoint. When unchecked, double clicking selects the text under the cursor.
Repair Mismatched CR/LF	If you cut-and-paste or import text from another editor and experience problems (disappearing code, etc.), you may want to check this checkbox. Then the MPLAB Editor will check, on save, for the correct line endings to work with Microchip language tools.
Line Wrap	When checked, line wrap is enabled.
Print Line Numbers	When checked, line numbers are printed out.
Enable Code Folding	When checked, provides a mechanism to collapse/expand nested code (similar to file folders). **C code:** Folding uses brackets - { } **ASM code:** Folding uses semicolon-brackets - ;{ to start a code block and ;} to end a code block.
Highlight Match	**C code only.** When cursor is on a match char, highlight both matched chars, with the color selected in the Text tab, Chose Colors, Match Token Highlight.
Close Matched	**C code only.** When any of { [(is typed, automatically place closing }]).
Tabs	
Tab Size	Enter a value from 1 to 16 to specify the width of tabs. **Insert spaces** – Select this to insert spaces up to the next tab position when you type a tab into text. **Keep tabs** – Select this to insert a hard tab character when you type a tab into text.

16.2.1.3 TOOLTIPS TAB

Set up tool tip properties using the **Tooltips** tab on the Editor Options dialog.

TABLE 16-3: TOOLTIPS TAB CONTROLS

Option	Description
Mouseover	
Enable Variable Mouseover Values	When checked, hovering mouse over a variable will pop up its value. Select its format from the list to the right.
Default Type Format	Select the format of the mouseover value. To use the same format as specified in a Watch window, select "Use Watch Preferences". Otherwise, select a specific mouseover format.
Show Address in Mouseover	When checked, prepends the address to the mouseover, e.g., 'Addr = 0x0003 Name = 0x00'
Mouseover Active on Debug	When checked, mouseovers will only be active when running or stepping code. On halt, the first edit keystroke will disable mouseovers. This helps avoid collisions with the Autocomplete feature.
Autocomplete	
Enable Autocomplete	When "Enable Autocomplete" and "Enable Tag Locators" (**Section 16.2.1.1 "General Tab"**) are checked, autocomplete is enabled. This means that if you hit <Ctrl>-<Space>, a list of selectable symbols for the project will pop up. This is the same list as in the Project window on the Symbols tab. The list will stay open as you type a symbol name, and match what you type with a selection from the list.
Autolist Struct members	When checked, typing a period (.) or right arrow (->) will bring up a list of selectable structure members for the project.
Function parameter information	When checked, typing a left parenthesis - (- will display the function prototype, if it has been previously defined in the project.

16.2.1.4 TEXT TAB

Set up text properties using the Text tab on the Editor Options dialog.

TABLE 16-4: TEXT TAB CONTROLS

Option	Description
Fonts	
Select Fonts	Click to select the editor font.
National Language Code Page:	For Unicode-enabled operating systems only. Select the value to be used to translate from Unicode text to ANSI.
Colors	
Choose Colors	Click to select the context-sensitive colors in the Editor Color Customization Dialog. Determine the context under Text Mode.
Default Colors	Click to revert to default context-sensitive colors.

TABLE 16-4: TEXT TAB CONTROLS (CONTINUED)

Option	Description
User Defined Color File	Color may also be applied to keywords you define in a file. Create the file as specified below, then enter or browse to the file to select it in this tab of the dialog. Click on **Choose Colors** to assign colors to your selected file keywords. • The format of the file should be one keyword per line with a carriage return after each keyword. • Keywords in the file must be alphanumeric with an alpha character as the first character. Keywords that use other characters will be ignored. • Keywords should be unique. If you choose an already defined directive or reserved keyword, then the color displayed is indeterminate.

16.2.1.5 OTHER TAB

Set up other properties using the **Other** tab on the Editor Options dialog.

TABLE 16-5: OTHER TAB CONTROLS

Option	Description
Default Window Width	Range: 10 to 80 characters. Specify the size you would like the editor window to use as a default.
Editor and Disassembly Gutter Width	Range: 3 to 12 characters. Specify the gutter width on windows that have gutters.
Line End Ruler Position	Disabled: 0, Range: 1 to 200 characters wide. Specify a position for the end-of-line ruler. The “ruler” will appear as a vertical line at the specified position running the length of the window. **Ruler Color**: Click to select a color for the ruler.
Debugger PC Location - Highlight Full Line	Check to highlight the line at the PC location. **Full Line Color**: Click to select a highlight color.
Format Comment Block - see Section 16.5.7 “Format Comment Block”.	
Block Comment Length	Range: 10 to 200 characters. Format the selected comment block to the desired length.
Block Comment Prefix String	Add the prefix to each line of the comment block. For C this could be “* ”. For assembly, this could be “; “.

16.2.2 Editor Color Customization Dialog

You can set color options to be used by the editor in the Editor Options dialog, **Text** tab. The colors will be used for all files which will be opened subsequently having the same syntax type as the currently active source file.

The color change dialog contains a list of specific symbol types which have their own coloring. The meaning of each kind of symbol type depends on the kind of file being colored. As each symbol type is selected, the "Foreground Color" and "Background Color" buttons show the current colors for that type. To change a color, click on the button and the Colors dialog will come up to allow you to design a new color. You may also specify that a symbol type appear in bold and/or italic face by checking the appropriate box. If you want to see the effect of your current changes without closing the dialog, click **Apply**. Otherwise your changes will take effect when you click **OK**, which closes the dialog.

Syntax coloring identifies text in each of the following categories. The syntax type of the source file determines which category a piece of text belong in.

- ALL Settings – global color setting for all categories
- Whitespace – "invisible" characters such as spaces and tabs
- Special Symbols – punctuation and operators
- Floating-Point Numbers
- Implicit-Base Numbers – numbers in the default number base for the file
- Binary Integers
- Octal Integers
- Decimal Integers
- Hexadecimal Integers
- String Constants
- Character Constants
- Preprocessor Directives
- Trigraphs and Digraphs – special key combinations used to represent Special Symbols which do not appear on the keyboard (obsolescent C feature)
- Comments
- Labels
- Reserved Words
- Bad – invalid characters
- Editor Window – text not belonging in any of the above categories
- User Defined File – text defined by the user file specified on Text tab
- Match Token Highlight – matching tokens (brackets, etc.) are highlighted

In addition to color, you can designate **Bold** or *Italic* as well with checkboxes.

16.3 WORKING WITH FILES

The following editor features are available for working with files:

- Creating a New File
- Opening Files
- Viewing Files
- Printing Files
- Saving Files
- Closing Files
- Using Bookmarks
- Using a Compare/Difference Utility

16.3.1 Creating a New File

To create a new file:

1. Click the New File icon, select *File>New* or press <CTRL> + <N>. A new window will open named “Untitled”. This is your new file.
2. To give the file the name you want it to have, select either *File>Save* or *File>Save As*.

16.3.2 Opening Files

To open an existing file:

1. There are two ways to select an existing file to open:
   - Click the Open File icon, select *File>Open* or press <CTRL> + <O>. The Select Input File dialog opens. In the dialog, browse to the location of the file you want to open and select it. Click the **Open** button.
   - Select the file from the Project window by double clicking on it.
2. The selected file is displayed in an editor window. If the selected file is already open, its current editor window will become the active window.

To protect a read-only file:

1. Select *Edit>Properties* or right click in a file (editor) window and select “Properties”.
2. Click on the **General** tab and check “Protect Read Only Files”.

When checked, read-only files cannot be edited. When unchecked, you can edit the file and save the changes to another file.

16.3.3 Viewing Files

To see line numbers:

1. Select *Edit>Properties* or right click in a file (editor) window and select “Properties”.
2. Click on the ***ASM/C/BAS* File Type** tab and check “Line Numbers”.

To wrap lines of text:

1. Select *Edit>Properties* or right click in a file (editor) window and select “Properties”.
2. Click on the ***ASM/C/BAS* File Type** tab and check “Line Wrap”.

To set up tab widths:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the ***ASM/C/BAS* File Type** tab and under "Tab Size" enter a value from 1 to 16.
3. Select either "Insert spaces" or "Keep tabs". Select "Insert spaces" to insert spaces up to the next tab position when you type a tab into text. Select "Keep tabs" to insert a hard tab character when you type a tab into text.

To see files in a tabbed window:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the **General** tab and check "Use Tabbed Window".

This will result in files being displayed on tabs in a single window, instead of in individual windows.

To set up other display features:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the **Other** tab and set up the default window width, gutter width and line end ruler position.

16.3.4 Printing Files

To print an open file:

1. Make sure the window that contains the file you want to print is the active window.
2. Click the Print icon, select *File>Print* or press <CTRL> + <P>. The Print dialog box is displayed.
3. Select the print options you want and click **OK**. The file is printed on the selected printer.

To print in color:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the **General** tab and check "Enable Color Printing".

To print line numbers:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the ***ASM/C/BAS* File Type** tab and check "Print Line Numbers".

16.3.5 Saving Files

To save a file:

1. Make sure the window that contains the file you want to save is the active window.
2. Click the Save icon, select *File>Save* or press <CTRL> + <S>. The file is saved with the name on its window.

To save a file with a different name:

1. Make sure the window that contains the file you want to save is the active window.
2. Select *File>Save As*. The New File Name dialog displays.
3. In the dialog, browse to the folder where you want to save the file.
4. In the “File Name” field, modify the name of the file if you want to save the file with a different name.
5. Click **Save**.

16.3.6 Closing Files

There are several ways of closing a file, as shown below:

- From the File Menu:
  - Make sure the window containing the file you want to close is the active window.
  - Select *File>Close*. If the file has changed since it was saved last, you will be prompted to save your changes.
- From the System Button menu on the file window, select “Close”.
- Click the close button (X) on the file window.
- Type <CTRL> + <F4> when the file you want to close is the active window.

16.3.7 Using Bookmarks

From the context (right mouse) menu, select “Bookmarks”. Use bookmarks to easily locate specific lines in a file.

- Toggle Bookmark – Sets/removes an unnamed bookmark at the currently selected line.
- Next/Previous – Go to the next/previous bookmark.
- Clear All – Clear all bookmarks.
- Bookmarks – Opens the Bookmarks window

Bookmarks Window

This window displays a list of all bookmarks. Manage your bookmarks here.

16.3.8 Using a Compare/Difference Utility

MPLAB IDE does not provide a file compare or difference (DIFF) utility. However, you may install a utility of your choosing and configure it to work with MPLAB IDE.

From the Edit menu or right click menu in an editor window, select "External DIFF". The External DIFF Arguments dialog will open.

TABLE 16-6: EXTERNAL DIFF ARGUMENTS DIALOG

Option	Description
Files/Folders	
File/Folder 1 File/Folder 2	Specify the files to compare. Include their paths, if necessary. Folder names may be edited from the selected filename. Use the **Browse** button to browse to a file.
Double Quote Long Paths	Some utilities may not be able to handle spaces properly, so check to place quotes around path/file names.
DIFF Command Arguments	Enter any command-line arguments to control the operation of the DIFF utility.

On the External DIFF Arguments dialog, you can click **Configure** to open the External DIFF Configuration dialog.

TABLE 16-7: EXTERNAL DIFF CONFIGURATION DIALOG

Option	Description
External DIFF	
DIFF Executable Path	Enter the path to the compare/difference utility. Use the **Browse** button to browse to the executable location.
Executable Type	*Windows GUI DIFF* - select if the utility is a Windows utility. *Command Line DIFF pipes to Output Window* - select if the utility is used on the command line. In this case, its output will be displayed in the MPLAB IDE Output window.
DIFF Command Arguments	Enter any command-line arguments to control the operation of the DIFF utility.

16.4 WORKING WITH TEXT

The following editor features are available for working with text:

- Selecting Text
- Moving to a Specific Line or Label
- Cutting/Deleting Text
- Copying Text
- Pasting Text
- Finding Text
- Replacing Text
- Matching Braces
- Undoing Editor Actions
- Redoing Editor Actions
- Indent and Outdent
- Formatting Text

16.4.1 Selecting Text

To select any block of characters:

- With the mouse:
  - Click the mouse at one end of the text you wish to select.
  - Hold down the left mouse button and move the mouse cursor to the other end of the text.
  - Release the left mouse button. The text you selected is now highlighted.
- With the keyboard:
  - Using the navigation keys (see **Section 16.7.2 "Movement and Selection"**), move the text cursor to one end of the text you wish to select.
  - Hold down a shift key and move the text cursor to the other end of the text.
  - Release the shift key. The text you selected is now highlighted.

To select a whole word or line:

Note:	This function is available if you do not have "Double click Toggles Breakpoint" checked in the Editor Options dialog, ***ASM/C/BAS* File Type** tab.

- Word: Double click the word you want to select. The word is now highlighted.
- Line: Double click a word in the line you want to select. Then click again. The line is now highlighted.

To select the entire contents of the file:

Press <CTRL> + <A> or select *Edit>Select All*. The entire buffer is now highlighted.

Note:	If you move the cursor after selecting text, the text will no longer be selected.

16.4.2 Moving to a Specific Line or Label

No matter where the cursor is in a file, you can quickly move it to any line or label that you specify.

To move to a specific location in the active window:

1. Select *Edit>Goto* or press <CTRL> + <G>. A Goto dialog displays.
2. Enter the line number or label in the dialog and click the **OK** button. The editor will move the cursor to the specified line/label and scroll to display the line/label, if necessary.

Note:	If you enter a number larger than the number of lines in the file, a warning dialog will open.

To move to the beginning of a line or text:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the **General** tab and select the desired action for the <Home> key under "Alternate <Home> key action".
3. Hit the <Home> key to perform this action.

16.4.3 Cutting/Deleting Text

To cut text:

1. Select the text to cut.
2. Click the Cut icon, select *Edit>Cut*, press <CTRL> + <X> or press <Shift> + <Delete>.

The selected text is deleted from the source document but moved to the Windows clipboard so that it may be pasted elsewhere.

To delete text:

1. Select the text to delete.
2. Select *Edit>Delete* menu or press <Delete>.

The selected text is deleted from the source document. If you deleted in error, immediately perform an Undo.

To delete single characters:

1. To delete the character to the left of the cursor, press <Backspace>.
2. To delete the character under the cursor, press <Delete>.

To delete a whole word (left):

1. Click in the line to the left of the word.
2. Press <CTRL> + Backspace.

To delete a line of text:

1. Click in the line to delete.
2. Press <CTRL> + <Shift> + Y.

16.4.4 Copying Text

To copy text:

1. Select the text you want to copy.
2. Click the Copy icon, select *Edit>Copy*, press <CTRL> + <C> or press <CTRL> + <Insert>.

To copy a column of text:

1. Place the cursor at a corner of the text column you want to copy.
2. Hold down <Alt> and then click-and-drag with the mouse to select the text column.
3. Click the Copy icon, *Edit>Copy*, right click on the selection and choose Copy, press <CTRL> + <C> or press <CTRL> + <Insert>.

Note:	If the wrong column is highlighted, make sure your cursor is on the same line as your mouse pointer for proper selection.

The selected text is copied to the Windows clipboard so it can be pasted elsewhere.

16.4.5 Pasting Text

You can paste any text contained in the Windows clipboard into a file in the MPLAB Editor. You can paste text that you have copied from the same file or from another application. Text must exist in the Windows clipboard before it can be pasted.

To paste text from the Windows clipboard:

1. Move the cursor to the point where you want to insert the text.
2. Click the Paste icon, select *Edit>Paste*, press <CTRL> + <V> or press <Shift> + <Insert>.

Note:	Pasting text does not remove it from the clipboard; you may paste it multiple times.

To repair pasted text that causes errors:

If you cut-and-paste or import text from another editor and experience problems (disappearing code, etc.):

1. Select *Edit>Properties* or right click in a file (editor) window and select “Properties”.
2. Click on the ***ASM/C/BAS* File Type** tab and check “Repair Mismatched CR/LF”.

The MPLAB Editor will now check, on save, for the correct line endings to work with Microchip language tools.

16.4.6 Finding Text

To find text in a file:

1. Make sure the file you want to search is open in the active window.
2. Select *Edit>Find* or press <CTRL> + <F>. The Find dialog displays.
3. If you selected text before opening the Find dialog, that text is displayed in the “Find What” field. If the text you want to search for is not displayed in the “Find What” field, enter the text you want to find or select a previously found text string from the drop-down list.
4. Select any of the Find options in the dialog that you want to use for this search. These options are:
   - Match whole word only
   - Match case
   - Direction – Up
   - Direction – Down
5. Click **Find Next**. The editor will move the cursor to the next (previous) matching text in the file. If there is no such text, the editor will display a message saying so.

To repeat the previous find:

Press <F3>, or, if the Find dialog is still up, click **Find Next**. To repeat the previous find in the opposite direction, press <Shift> + <F3>.

To be notified if the find wraps:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the **General** tab and check "Find Wrap MessageBox".

A message box is displayed when the find wraps top or bottom, in addition to the current message of "Find Wrapped" in the status bar position panel.

16.4.7 Replacing Text

To find and replace text in a file:

1. Make sure the file you want to edit is open in the active window.
2. Select *Edit>Replace* or press <CTRL> + <H>.
3. If you selected text in the file before opening the Replace dialog, that text is displayed in the "Find What" field. If the text you want to replace is not displayed in the "Find What" field, enter the text you want replace.
4. In the "Replace With" field, enter the replacement text.
5. Select any of the Find/Replace options in the dialog that you want to use for this search. These are:
   - Match whole word only
   - Match case
6. Click:
   - **Find Next** to do a simple find
   - **Replace** to replace the closest matching text
   - **Replace All** to replace all matching text
   - Cancel to close the dialog with no further action.

To repeat a Replace operation:

1. Press <F3>.
2. If the Find/Replace dialog is still open, click **Replace**.

The most recently performed Find/Replace operation is executed, including all Find/Replace options that were selected.

16.4.8 Matching Braces

To recognize a section of bracketed text:

1. Move the cursor to an opening/closing symbol.
2. Press <CTRL> + <M>, select *Edit>Advanced>Match* or right click in the edit window and select *Advanced>Match*. The cursor will move to the matching symbol.

Opening/closing symbols are: [ ] { } ( ) < >

See also **Section 16.5.3 "Matching C Code Braces"**.

16.4.9 Undoing Editor Actions

If you have just made a change to a file, you can reverse the effect of the last change you just made.

- Select *Edit>Undo* or press <CTRL> + <Z>.

You can repeat the Undo action multiple times in succession. Each will undo the action prior to the last undo.

16.4.10 Redoing Editor Actions

If you have just reversed a change with the Undo function, you can redo the change.

- Select *Edit>Redo* or press <CTRL> + <Y>.

You can repeat the Redo action multiple times in succession.

16.4.11 Indent and Outdent

You may indent or outdent text by the tab width specified in the Editor Options dialog (*Edit>Properties*), ***ASM/C/BAS* File Type** tab. Then use <Tab>/<Shift>+<Tab> or Indent Block/Outdent Block commands (Right click menu, Advanced).

To use Auto Indent, you must enable this feature in the Editor Options dialog (*Edit>Properties*), ***ASM/C/BAS* File Type** tab.

16.4.11.1 INDENTING

Single line

1. Move the cursor to the start of the line you want to indent.
2. Make sure the editor is in Insert mode
3. Press <Tab> once for each level you want to indent the line.

Auto Indent

New line automatically indented like the previous line.

Block of lines

1. Select the lines in the block you want to indent.
2. Press <Tab> once for each level you want to indent the lines in the block.

16.4.11.2 OUTDENTING

Text must of been indented before outdenting will work.

You cannot outdent a line backwards through column 1.

Single line

1. Move the cursor to the start of the line you want to outdent.
2. Press <Shift> + <Tab> once for each level you want to outdent the line.

Block of lines

1. Select the lines in the block you want to outdent.
2. Press <Shift>+<Tab> key once for each level you want to outdent the lines in the block.

16.4.12 Formatting Text

To setup fonts and colors:

1. Select *Edit>Properties* or right click in a file (editor) window and select “Properties”.
2. Click on the **Text** tab and select the editor font and context-sensitive colors.

To see color-coded syntax:

Please refer to **Section 16.5.1 “Syntax Type”**.

To set the case of a selection of text:

1. Select the text.
2. Right click on the selection and then choose either *Advanced>Uppercase* or *Advanced>Lowercase* to make the selected text either all uppercase or all lowercase, respectively.

16.5 WORKING WITH PROGRAMMING LANGUAGES

The MPLAB Editor has several features that help you identify and work with various programming languages.

- Syntax Type
- Matching C Preprocessor Directives
- Matching C Code Braces
- Code Folding
- GoTo Locator
- Commenting Out Text
- Format Comment Block
- Additional Programming Resources

16.5.1 Syntax Type

For the purposes of syntax coloring, the MPLAB Editor recognizes the syntax of various assemblers and compilers. You may specify the language type a source file contains.

1. Right click in the file for which you wish to set the syntax type.
2. Select Text Mode from the right mouse button menu in an editor window. You will get a menu containing the syntax types which the editor currently recognizes.
   - Disabled – No text formatting.
   - PIC16C5x Asm – 12-bit core MCU devices
   - PIC16Cxxx Asm – 14-bit core MCU devices
   - PIC17Cxxx Asm – 16-bit core MCU devices (End of life)
   - PIC18Cxxx Asm – Enhanced 16-bit core MCU devices
   - ASM30 – 24-bit core MCU/DSC devices
   - C – C language listing
   - Basic – Basic listing
   - SCL – Simulator Control Language listing
3. Select the closest matching language type for your file.

16.5.2 Matching C Preprocessor Directives

To recognize a section of bracketed text:

1. Move the cursor to an opening/closing symbol.
2. Press <CTRL> + <M>, select *Edit>Match* or right click in the edit window and select *Advanced>Match*. The cursor will move to the matching symbol.

Opening/closing symbols are: `#if`/`#ifdef`, `#endif`.

16.5.3 Matching C Code Braces

In addition to the basic Match Braces (**Section 16.4.8 "Matching Braces"**), there are other options for C code.

To place braces with auto indent:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the ***ASM/C/BAS* File Type** tab and check "Auto Indent with Brace Placement".

To highlight matching braces:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the ***ASM/C/BAS* File Type** tab and check "Highlight Match".
3. To pick the highlight color, click the Text tab, "Choose Colors", and then select the Match Token Highlight.

To automatically place the closing brace:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the ***ASM/C/BAS* File Type** tab and check "Close Matched".

16.5.4 Code Folding

Select *Edit>Properties* or right click in a file (editor) window and select "Properties". Click on the ***ASM/C/BAS* File Type** tab and check "Enable Code Folding". When checked, code folding provides a mechanism to collapse/expand nested code (similar to file folders):

- C code: Folding uses double-forward slash-brackets - `//{` to start a code block and `//}` to end a code block. Or you could use matching preprocessor directives - `#if`/`#ifdef`, `#endif`.
- ASM code: Folding uses semicolon-brackets - `;{` to start a code block and `;}` to end a code block.

To use code folding, right click on the desired text and select one of the following from the "Code Folding" submenu:

- Expand All
- Collapse All
- Expand Block
- Collapse Block

16.5.5 GoTo Locator

Go to where the selected C code symbol is defined. For example, right click on a function (in the Project window, Symbol tab, or in code in a File window) and select "GoTo Locator" to go to the line in code where this function is declared.

To enable "GoTo Locator" functionality:

1. Select *Edit>Properties* or right click in a file (editor) window and select "Properties".
2. Click on the **General** tab and check "Enable Tag Locators".

For more information, see **Section 13.20.3 "Project Window Display – Symbols Tab"**.

16.5.6 Commenting Out Text

To comment out a block of code:

1. Select the code to be commented out.
2. Right click on the selection and choose *Advanced>Comment Block*.

To uncomment a block of code:

1. Select the code to be uncommented.
2. Right click on the selection and choose *Advanced>Uncomment Block*.

16.5.7 Format Comment Block

A comment block is a long comment broken up into multiple lines. For C, it would be of the form:

```
/************************************************
This function does this. It takes in these inputs
with these expected preconditions. The output x
will be generated.
************************************************/
```

When you want to edit this type of comment, it can be difficult as you may have to manually reformat the old text to make the new text fit. For example:

```
/************************************************
This function does this. It can also do this. It takes in these inputs
with these expected preconditions. The output x
will be generated.
************************************************/
```

To automatically reformat the comment block:

1. Select *Edit>Properties* and click the **Other** tab. Enter the desired length of each line in the comment block under "Block Comment Length". Click **OK**.
2. Highlight the comment block and select *Edit>Advanced*, "Format Comment".

For some comment blocks, you may wish to precede each line with a prefix, as for this assembly comment:

```
; This function does this. It takes in these inputs
; with these expected preconditions. The output x
; will be generated.
```

To automatically add a prefix to the comment block text:

1. Select *Edit>Properties* and click the **Other** tab. Enter the desired prefix for each line in the comment block under "Block Comment Prefix String". Click **OK**.
2. Highlight the comment block and select *Edit>Advanced*, "Format Comment".

16.5.8 Additional Programming Resources

For help on assembly code, select *Help>Topics>Lanugage Tools* and then select an assembler that supports your device. Assembler help contains assembler directive information (e.g., `udata`, `pagesel`, `banksel`.) Also, you can find PDF versions of these documents on our website (www.microchip.com).

For help on C code, consult our website for PDF documentation.

For help on instruction sets (e.g., `movlw`, `btfss`, `goto`), consult our website for PDF versions of data sheets.

16.6 WORKING WITH DEBUG FEATURES

When a debugger is selected in MPLAB IDE, several code debugging features are available in the editor window.

- Mouseovers
- Autocomplete
- Breakpoints
- Filter Trace

16.6.1 Mouseovers

When mouseovers are enabled, hovering the mouse pointer over a variable will pop up its value. To enable and set up mouseovers:

1. Do one of the following:

a) In the Project window (*View>Project*, **Symbols** tab), right click in an empty area of the window and select “Enable Tag Locators”. Then right click again and select “Update Tags Now”.

b) Select *Edit>Properties* or right click in a file (editor) window and select “Properties”. Click on the **General** tab and check “Enable Tag Locators“.

2. Select *Edit>Properties* or right click in a file (editor) window and select “Properties”. Click on the **ToolTips** tab and enable/set up mouseovers.

If you only enable mouseovers on the **ToolTips** tab, you may see some simple variable values. However, equations will not be evaluated.

An additional way to evaluate equations is to drag them to a Watch window to see their value.

16.6.2 Autocomplete

When autocomplete is enabled, if you hit <Ctrl> <Space>, a list of selectable symbols for the project will pop up. This is the same list as in the Project window on the Symbols tab. To enable and set up the autocomplete function:

1. Do one of the following:

a) In the Project window (*View>Project*), right click in an empty area of the window and select “Enable Tag Locators”. Then right click again and select “Update Tags Now”.

b) Select *Edit>Properties* or right click in a file (editor) window and select “Properties”. Click on the **General** tab and check “Enable Tag Locators“.

2. Select *Edit>Properties* or right click in a file (editor) window and select “Properties”. Click on the **ToolTips** tab and enable/set up autocomplete features.

16.6.3 Breakpoints

Breakpoints stop executing code at a designated line so that you may observe register, bit and variable values for debugging.

Breakpoints are a standard feature of MPLAB IDE Debug tools. For more information, see **Section 5.18 “Using Breakpoints”**.

To enable breakpoint functionality on a double click:

1. Select *Edit>Properties* or right click in a file (editor) window and select “Properties”.
2. Click on the ***ASM/C/BAS* File Type** tab and check “Double click Toggles Breakpoint”.

When checked, double clicking on a line will alternately enable/disable the breakpoint. When unchecked, double clicking selects the text under the cursor.

16.6.4 Filter Trace

Some debug tools support setting up a simple filter trace in the editor window. See your debug tool documentation for details. If the debug tool trace feature is already enabled, you will be warned that using this filter trace may overwrite your previous settings.

Filter-in and Filter-out trace are mutually exclusive, i.e., setting one will clear the other.

Filter-In Trace

You may select (filter) the code that is traced into the trace buffer (for display in a Trace window) by:

- selecting the code to add to the trace in the editor window
- right clicking in the window and selecting "Add Filter-in Trace" from the menu

Filter-Out Trace

You may select (filter) the code that is to be excluded from the trace buffer and thereby trace only the unselected code into the trace buffer by:

- selecting the code to exclude from the trace in the editor window
- right clicking in the window and selecting "Add Filter-out Trace" from the menu

Removing Traces

To remove the code from the trace, right click in the filter code and select "Remove Filter Trace".

To remove multiple sections that have be used for trace, right click in the window and select "Remove All Filter Traces".

16.7 KEYBOARD FEATURES

The following keys are specified for the editor:

- Shortcuts
- Movement and Selection
- Special Characters

16.7.1 Shortcuts

The following keyboard shortcuts are available for the editor. To change shortcut key assignments, use *Configure>Settings*, **Hot Keys** tab.

16.7.1.1 EDITING

Keystroke		Result
CTRL + Z	Alt + Backspace	Undo last edit
CTRL + Y	Shift + Alt + Backspace	Redo last edit
CTRL + X	Shift + Delete	Cut selected text
CTRL + C	CTRL + Insert	Copy selected text
CTRL + V	Shift + Insert	Paste text
Delete		Delete selected text
CTRL + Backspace		Delete whole word (left)
CTRL + Shift + Y		Delete line
CTRL + A		Select all text in the window
CTRL + F		Find
CTRL + H		Find/Replace
F3		Repeat find/replace down
SHIFT + F3		Repeat find/replace up
CTRL + G		Goto line
CTRL + M		Match brace, C preprocessor directive

16.7.1.2 FILE MANAGEMENT

Keystroke	Result
CTRL + N	Create new text file
CTRL + O	Open existing file
CTRL + F4	Close existing file
CTRL + S	Save active file
CTRL + P	Print active file

16.7.1.3 NAVIGATION

Keystroke	Result
CTRL + F6	Next open window
SHIFT + CTRL + F6	Previous open window
CTRL + K	Toggle Bookmark
CTRL + L	Next Bookmark
F1	Help

16.7.2 Movement and Selection

The keyboard keystrokes shown in Table 16-8 may be used when editing text.

TABLE 16-8: KEYSTROKES FOR EDITING TEXT

Keystroke	Result
CTRL + A	Selects all the text in the file
UP	Moves the cursor up by one line
SHIFT + UP	Moves the cursor up by one line, extending the selection
DOWN	Moves the cursor down by one line
SHIFT + DOWN	Moves the cursor down by one line, extending the selection
LEFT	Moves the cursor left by one character
SHIFT + LEFT	Moves the cursor left by one character, extending the selection
CTRL + LEFT	Moves the cursor left by one word
CTRL + SHIFT + LEFT	Moves the cursor left by one word, extending the selection
RIGHT	Moves the cursor right by one character
SHIFT + RIGHT	Moves the cursor right by one character, extending the selection
CTRL + RIGHT	Moves the cursor right by one word
CTRL + SHIFT + RIGHT	Moves the cursor right by one word, extending the selection
PGDN	Moves the cursor down by one page
SHIFT + PGDN	Moves the cursor down by one page, extending the selection
PGUP	Moves the cursor up by one page
SHIFT + PGUP	Moves the cursor up by one page, extending the selection
HOME	Moves the cursor to the start of the line or alternates between the start of the text and the start of the line. See **Section 16.2.1.1 "General Tab"**.
SHIFT + HOME	Moves the cursor to the start of the line, extending the selection
CTRL + HOME	Moves the cursor to the start of the file
CTRL + SHIFT + HOME	Moves the cursor to the start of the file, extending the selection
END	Moves the cursor to the end of the line
SHIFT + END	Moves the cursor to the end of the line, extending the selection
CTRL + END	Moves the cursor to the end of the file
CTRL + SHIFT + END	Moves the cursor to the end of the file, extending the selection

16.7.3 Special Characters

To enter any single character:

1. Put the keyboard keypad into numeric mode with <NumLock>.
2. Hold down <Alt>.
3. Type the decimal number of the character you want (between 0 and 256).
4. Release <ALT>.

If the character you typed has a visible form, it will appear at the cursor in your file window.

Insert/Overtype Modes:

Use the Insert key to toggle between INS and OVR mode, shown on the status bar.

Key(s)	Insert Mode	Overtype Mode
Enter	Insert NewLine	Move to first character of next line, if any
TAB or CTRL + I	Insert Tab	Move to next tab position, if any

16.8 EDITOR TROUBLESHOOTING

This section is designed to help you troubleshoot any problems, errors or issues you encounter while using MPLAB Editor.

16.8.1 Limitations

- **Find in Files with a split window**

  When using “Find in Files” with a split window (see **Section 13.9.1 “File Window Display”**), both splits will reflect the location of the Find, not just the active one.
- **Can’t see Autocomplete bitfields for MPLAB C18 SFR extern structs**

  In MPLAB C18, all SFR names are defined in an ASM library, so they are `extern` in the C header file. Autocomplete relies on the translator, which requires the `stuct` to be in scope.
- **Language Tools don’t support UNICODE**

  MPLAB Editor allows files to be saved in UNICODE format on UNICODE enabled operating systems. Those files stored in UNICODE format will NOT compile with Microchip language tools, and you cannot use them to debug.
- **Symbol Fonts**

  Several of the MPLAB Editor fonts are Symbol based and are not suitable for use with the editor. The Symbol fonts are included in the selection list to allow you to select the Terminal font.
- **Large Fonts**

  When using a system display font size of Large Fonts (125% normal), and selecting an Italic font, some characters may display truncated on the top-right border of the editor.
- **Font/Color changes not made for open windows**

  When setting the properties for text font and colors, the Editor and the Debug displays do not broadcast the changes to already open windows. The changes take effect when the debug display is next opened, or the editor file is re-opened.

16.8.2 Common Problems/FAQ

- **Some items in the context menu are disabled.**

  Depending on your selection of debug and/or programmer tools, some items may not be applicable and so are grayed.
- **Nothing happens when I press <CTRL> + <M>.**

  The text cursor must be on an opening/closing symbol when you press <CTRL> + <M>. When it is, Match will move it to the matching symbol. If there is no match, the cursor does not move.

  Opening/closing symbols are: [ ] { } ( ) < >
- **When I insert a carriage return, the new line always starts in column 1, regardless of the current indentation level.**

  Auto Indent must be checked in the Tab Settings dialog for auto indent to work.

- **I imported some code into a file window, and now it's missing. What happened?**

  Text editors end their lines in different ways. Therefore imported or cut-and-pasted text may not be compatible with Microchip tools. In the Editor Options dialog, on the ***ASM/C/BAS* File Types** tab, check the checkbox for "Repair Mismatched CR/LF" to fix any improper carriage returns or line feeds.
- **I've finished writing my code, and now it won't assemble/compile. What's wrong?**

  Some things to check are:

  - Did you cut-and-paste some example code from a document? The text may not be of a supported type, i.e., UNICODE. Microchip language tools don't support this format. Select *Edit>Save As*, choose "ANSI" as the "Encoding" and then click **Save**.

  - Did you save the source code as UNICODE? Again, Microchip language tools don't support this format.Select *Edit>Save As*, choose "ANSI" as the "Encoding" and then click **Save**.

  - Do you have the latest version of the language tool? Go to our website to try free student versions or purchase full versions.
- **I switched devices in my open project and now mouseovers/autocomplete won't work**

  You need to rebuild the project.

Part 6 – MPLAB SIM Simulator

NOTES:

Chapter 17. Simulator Overview

17.1 INTRODUCTION

MPLAB SIM is a discrete-event simulator for:

- PIC microcontroller (MCU) families
- dsPIC Digital Signal Controller (DSC) families

The simulator is integrated into MPLAB IDE (integrated development environment). The MPLAB SIM debugging tool is designed to model the operation of Microchip Technology's MCU and DSC devices to assist in debugging software for these devices.

If you are new to MPLAB SIM, you may want to begin with the MPLAB IDE tutorials in the MPLAB IDE online help.

17.2 SIMULATOR FEATURES

MPLAB SIM allows you to:

- Modify object code and immediately re-execute it
- Inject external stimuli to the simulated processor - stimulus is the simulation of hardware signals
- Set pin and register values at pre-specified intervals
- Trace the execution of the object code
- Code coverage to detect “dead” code areas
- Extract data for validation

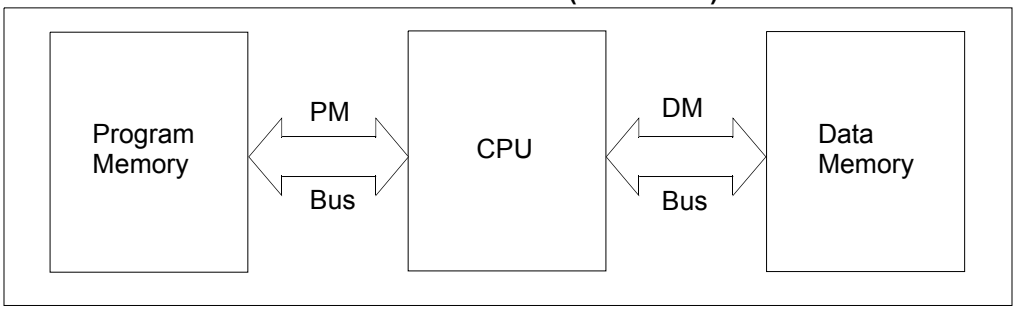
17.3 SIMULATOR MODEL

The simulator model is dependent on device architecture. The model is controlled through the MPLAB IDE.

17.3.1 Architecture - 8-bit devices

Microchip 8-bit devices use Harvard architecture, with separate program and data memory buses (Figure 17-1). Therefore, program memory instructions are not limited to data memory lengths.

FIGURE 17-1: DEVICE ARCHITECTURE (HARVARD)



Program memory, or core, instruction length is used to group 8-bit PIC MCUs. Data memory is 8-bit for these devices.

The 12-bit, 14-bit and 16-bit (PIC17) core devices have word-addressed program memory space.

The 16-bit (PIC18) core devices have a byte-organized program memory space. There are some restrictions in the silicon on how the program memory space can be accessed, especially when using long writes to program memory. The simulator may not show the same restrictions in all situations. Consult the data sheet for proper operation.

At the time this documentation was produced, the following ISA (instruction set architecture) cores are associated with the listed devices:

- 12-bit ISA Core – PIC12C5XX, PIC12CE5XX, PIC16X5X, PIC16C505
- 14-bit ISA Core – PIC12C67X, PIC12CE67X, PIC12F629/675, PIC16
- 16-bit ISA Core – PIC17
- 16-bit ISA Core – PIC18

17.3.2 Architecture - 16-bit devices

Microchip 16-bit devices use a modified Harvard architecture, with separate program and data memory buses (Figure 17-1). Therefore, program memory instructions are not limited to data memory lengths.

The dsPIC digital signal controller (DSC) is a combination of a digital signal processing (DSP) core and PIC microcontroller (MCU) peripheral features. PIC24 MCUs are basically dsPIC DSC devices without DSP capability. Both devices provide separate 24-bit program memory and 16-bit data memory spaces.

At the time this documentation was produced, the following were available devices of this type:

- dsPIC30F, dsPIC33F
- PIC24F, PIC24H

17.3.3 Architecture - 32-bit devices

Microchip 32-bit devices use Von Neumann architecture (a linear address space for program and data memory.) Both program and data memory are 32-bits long.

The 32-bit (PIC32) core devices have a half word organized program memory space. There are some restrictions in the silicon on how the program memory space can be accessed, especially when using long writes to program memory. The simulator may not show the same restrictions in all situations. Consult the data sheet for proper operation.

At the time this documentation was produced, the following were available devices of this type:

- PIC32MX

17.3.4 Simulator Operation and Displays (Except PIC17 Devices)

Stimulus is used for controlling the synchronous stimulation of hardware signals to the application under test. A stimulus component (dialog) is provided for defining and generating stimulus code, which can then be attached to the simulation session. The stimulus component (dialog) also supports generation of asynchronous stimulus, which may be used independently of, or in conjunction with, the synchronous stimulus.

The Oscillator setting can be changed to allow accurate timing of instructions through the Stopwatch display. It also allows computational delays required within peripherals during simulation.

The MIPS rating (Instructions Count and Execution Cycles) of the last executed code segment (not stepped code) can be displayed in the output window from *Debugger>Profile>Display Profile*. This gives the execution speed of code, based on the PC it is executed on. This may differ from the execution speed on target (target speed is dependent on Oscillator Frequency). MIPS rating for the same code may be different on different PC's.

When trace is enabled, there may be a reduction of speed with the simulator performance. Trace records all instructions as they occur with source and destination data and address. It also captures all I/O port values on each instruction for displaying under the logic analyzer.

There is `printf()` support in the output window for MPLAB C compilers. This uses file names placed in the text fields on UART1, taking priority over any stimulus, register injection or register trace files attached to UART1.

17.3.4.1 dsPIC30F/33F DSCS, PIC24F/H MCUS

The simulator of 16-bit families (SIM30) supports Code Guard for flow control, data memory access, EEPROM access, Flash program/erase and security configuration bit erase.

The simulator has no separate hardware stack. The stack is mapped into the 16-bit data memory space and can be viewed by setting a watch or opening the file register window at the appropriate location.

> **Note:** MPLAB IDE has a software (call) stack window.

The File Register window has additional features for supporting dsPIC devices. The File Register Window has an additional “X/Y Memory” tab that allows the inspection and/or modification of this memory space. X and Y memory space is differentiated in the display, and the basic operation is similar to the File Register Window.

The Program Memory window has two additional tabs for supporting dsPIC devices: "PSV" and "Mixed." When PSV memory is enabled on the device, the PSV window will show 16-bit data and labels at the addresses mapped. The Mixed display shows memory in both 24-bit and 16-bit, and tags data with labels and disassembles instructions. This memory can be viewed as both program memory and PSV data memory in the case where code and data share this area.

dsPIC33F/PIC24H devices have a dual port tab on the File Registers window to show DMA accessible memory more easily. This is the RAM used by the DMA peripheral.

17.3.4.2 PIC32MX MCUS

The memory in PIC32MX is implemented as one unified memory (from the MPLAB IDE main menu, select *View>Memory*). There is no separate window to show RAM memory. The unified memory window displays RAM memory, program memory and Boot Flash. This supports breakpoints, stepping, trace and other execution features associated with program memory, boot flash and RAM memory. You can navigate to different memory regions using the Go To option (select *Edit>Go To*). Tabs at the bottom of the display, as used on current displays, let you switch views between Machine code and Symbolic code.

There are two other types of memory window for display. They are CPU Registers and SFR/Peripherals. Select *View>CPU Registers* to display all CPU Registers. Select *View>SFR/Peripherals* to display registers associated with different peripherals.

The simulator has no separate hardware stack. The stack is mapped into the 32-bit data memory space and can be viewed by setting a watch or opening the file register window at the appropriate location.

17.3.5 Simulator Operation and Displays (PIC17 Devices)

Stimulus is used for controlling the stimulation of hardware signals to the application under test. Refer to **Chapter 20. "Using Stimulus – PIC17 Devices"** for details on using stimulus for PIC17 MCUs and the related dialogs.

17.4 SIMULATION DESCRIPTION

MPLAB SIM simulation depends on the device being simulated, i.e., the device selected in MPLAB IDE will determine how MPLAB SIM operates.

17.4.1 Pin Simulation – All Devices

MPLAB SIM only simulates to the register level, not the pin level, e.g., RB0 represents the value in bit0 of the PORTB register, not the value on the pin named RB0. This makes sense as the simulator is a software model, and not actual device hardware.

In general, bit/pin binary values are interchangeable. However, A/D operation requires that the corresponding port register read '0', which may be different from what would be on an actual pin. This can impact the operation of ADC and comparators. Please see the Peripheral section for more on how these are simulated.

Device I/O pins may be multiplexed with other peripherals (and therefore referred to by more than one name). The simulator recognizes only the pin names specified in the standard device headers as valid I/O pins. Therefore, you should refer to the header file for your device (p*DeviceNumber*.inc) to determine the correct bit/pin names.

Most multiplexed bit/pin names may be used interchangeably, e.g., for the PIC18F8720 pin RF2/AN7/C1OUT, you may use RF2 or AN7 or C1OUT (although it is recommended that you use the appropriate name for peripheral function you are using). Basically, AN7 and C1OUT are aliases for RF2. The only exception to this is when there is a multiplexed peripheral function across multiple ports. In this case, the function name is not selectable; you must choose the corresponding port bit/pin name. E.g., Configuration bits are used on the same PIC MCU to switch the CCP2 pin between RC1, RB3 and RB7; CCP2 will not be available for selection.

17.4.2 12-Bit Core Device Simulation

The following topics discuss the 12-bit core device features modeled in the simulator.

- 12-bit Core CPU
- 12-bit Core Peripherals

17.4.2.1 12-BIT CORE CPU

Reset Conditions

All Reset conditions are supported by the MPLAB SIM simulator. The types of resets available by selecting *Debugger>Reset* are:

- MCLR Reset
- Watchdog Timer Reset
- Brown Out Reset
- Processor Reset F6

The Time-out (TO) and Power-down (PD) bits in the STATUS register reflect appropriate Reset condition. This feature is useful for simulating various power-up and time-out forks in your code.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the pre/postscaler settings in the OPTION_REG register. The minimum period (with pre/postscaler at 1:1) may be set on the **Break Options** tab of the Settings dialog (*Debugger>Settings*).

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or Reset, depending on the selection in the **Break Options** tab of the Settings dialog.

17.4.2.2 12-BIT CORE PERIPHERALS

Along with core support, MPLAB SIM fully supports:

- TIMER0 timer/counter module in both internal and external clock modes.

> **Note:** It is important to remember that, because MPLAB SIM executes on instruction cycle boundaries, resolutions below 1 TCY cannot be simulated.

- A/D Converter
- Comparators
- EEPROM Data Memory
- OSC Control of IO
- IO Ports

TIMER0

Timer0 and the interrupt it can generate on overflow is fully supported, and will increment by the internal or external clock. Delay from external clock edge to timer increment has also been simulated, as well as the interrupt latency period. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to the stimulus file requirements.

A/D Converter

All the registers, timing function and interrupt generation are implemented. The simulator, however, does not load any meaningful value into the A/D result register (ADRES) at the end of a conversion unless an injection file has been attached to ADRES (L). To load meaningful data, use an injection file (see **Section 19.4.6 "Register Injection Tab"**) The simulator will produce a warning in the Output window if a conversion is initiated but no input data is available.

Note:	If you have trouble with I/O pins on processors that have A/D, make certain that the A/D pin control registers (ADCON registers) are configuring those pins for digital I/O rather than for analog input. For most processors, these default to analog inputs and the associated pins cannot be used for I/O until the ADCON (or ADCON1) register is set properly.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

Comparators

Only comparator modes that do not use Vref are simulated in MPLAB SIM.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

Toggling a comparator pin will not work since toggling involves reading a value and inverting it. Since the value always reads '0', the bit/pin never toggles. Instead, use two statements to toggle a comparator pin, e.g.,

- RA1 set low
- RA1 set high

EEPROM Data Memory

The EEPROM data memory is fully simulated. The registers and the read/write cycles are fully implemented. The write cycle time is device dependent (to nearest instruction cycle multiple).

The simulator simulates the functions of WRERR and WREN control bits in the EECON1 register. WRERR can be set using stimulus for testing purposes.

OSC Control of IO

The I/O pins are controlled for I/O depending on oscillator settings and whether the oscillator uses the pins.

IO Ports

All I/O ports are supported for input/output, interrupt and change events. The I/O is supported to a register level and not a pin level. See **Chapter 19. "Using Stimulus"**.

17.4.3 14-Bit Core Device Simulation

The following topics discuss the 14-bit core device features modeled in the simulator.

- 14-bit Core Interrupts
- 14-bit Core CPU
- 14-bit Core Peripherals

17.4.3.1 14-BIT CORE INTERRUPTS

The following interrupts are supported:

- Timers
- CCP/ECCP
- UARTS
- Change on Port RB <7:4>
- External interrupt from RB0/INT pin
- Comparators
- A/D converter
- EEPROM write complete

17.4.3.2 14-BIT CORE CPU

Reset Conditions

All Reset conditions are supported by the MPLAB SIM simulator.

The Time-out (TO) and Power-down (PD) bits in the STATUS register reflect appropriate Reset condition. This feature is useful for simulating various power-up and time-out forks in the user code.

A $\overline{\text{MCLR}}$ Reset during normal operation or during Sleep can easily be simulated by driving the $\overline{\text{MCLR}}$ pin low (and then high) via stimulus or by selecting *Debugger> Reset>MCLR Reset*.

Sleep

When executing a Sleep instruction, MPLAB SIM will appear “asleep” until a wake-up from sleep condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the pre/post-scaler setting).

An example of a wake-up-from-sleep condition would be Timer1 wake up from sleep. In this case, when the processor is asleep, Timer1 would continue to increment until it overflows. If the interrupt is enabled, the timer will wake the processor on overflow and branch to the interrupt vector.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the pre/postscaler settings in the OPTION_REG register. The minimum period (with pre/postscaler at 1:1) may be set on the **Break Options** tab of the Settings dialog (*Debugger>Settings*).

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or Reset, depending on the selection in the **Break Options** tab of the Settings dialog.

17.4.3.3 14-BIT CORE PERIPHERALS

Along with core support, MPLAB SIM supports the following peripheral modules, in addition to general purpose I/O. Consult the data sheet for the particular device you are using for information on which symbols are implemented.

> **Note:** Even if peripheral functionality is not supported in the simulator, the peripheral registers will be available as read/write.

- Timers
- CCP/ECCP
- Comparators
- A/D Converter
- USART
- EEPROM Data Memory
- OSC Control of IO
- IO Ports

The delays and interrupt latency are implemented on all peripherals.

Timers

Timer0 (and the interrupt it can generate on overflow) is fully supported, and will increment by the internal or external clock. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to stimulus requirements.

Timer1 in its various modes is supported, except when running in counter mode by an external crystal. MPLAB SIM supports Timer1 interrupts generated on overflow, and interrupts generated by wake-up from sleep.The external oscillator on RC0/RC1 is not simulated, but a clock stimulus can be assigned to those pins.

Timer2 and the interrupt that can be generated on overflow are fully supported.

CCP/ECCP

Capture

MPLAB SIM fully supports capture and the interrupt generated.

Compare

MPLAB SIM supports compare mode, its interrupt and the special event trigger (resetting Timer1 by CCP1).

PWM

PWM output is fully supported (resolution greater than 1 TCY only) except for Dead Band delay.

Comparators

Only comparator modes that do not use Vref are simulated in MPLAB SIM.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

Toggling a comparator pin will not work since toggling involves reading a value and inverting it. Since the value always reads '0', the bit/pin never toggles. Instead, use two statements to toggle a comparator pin, e.g.,

- RA1 set low
- RA1 set high

A/D Converter

All the registers, timing function and interrupt generation are implemented. The simulator, however, does not load any meaningful value into the A/D result register (ADRES) at the end of a conversion unless an injection file has been attached to ADRES (L). To load meaningful data, use an injection file (see **Section 19.4.6 "Register Injection Tab"**).

Note:	If you have trouble with I/O pins on processors that have A/D, make certain that the A/D pin control registers (ADCON registers) are configuring those pins for digital I/O rather than for analog input. For most processors, these default to analog inputs and the associated pins cannot be used for I/O until the ADCON (or ADCON1) register is set properly.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

USART

USART functionality is supported. For more information, see **Section 18.7 "Using a USART/UART"**.

The following limitations apply to USART operations:

- The receiver register must be stimulated using a **message-based** stimulus file. The characters are clocked into the receiver register at the default or current baud rate, starting at the time defined within the message-based stimulus file. If the receiver is not enabled when the data starts arriving, the data is lost.

EEPROM Data Memory

The EEPROM data memory is fully simulated. The registers and the read/write cycles are fully implemented. The write cycle time is device dependent (to nearest instruction cycle multiple).

The simulator simulates the functions of WRERR and WREN control bits in the EECON1 register. WRERR can be set using stimulus for testing purposes.

OSC Control of IO

The I/O pins are controlled for I/O depending on oscillator settings and whether the oscillator uses the pins.

IO Ports

All I/O ports are supported for input/output, interrupt and change events. The I/O is supported to a register level and not a pin level. See **Chapter 19. "Using Stimulus"**.

17.4.4 16-Bit Core Device Simulation – PIC17

The following topics discuss the 16-bit core device features modeled in the simulator.

- 16-bit Core (PIC17) Interrupts
- 16-bit Core (PIC17) CPU
- 16-bit Core (PIC17) Processor Modes
- 16-bit Core (PIC17) Peripherals

17.4.4.1 16-BIT CORE (PIC17) INTERRUPTS

The following interrupts are supported:

- External interrupt on INT pin
- TMR0 overflow interrupt
- External interrupt on RA0 pin
- Port B input change interrupt
- Timer/Counter1 interrupt
- Timer/Counter2 interrupt
- Timer/Counter3 interrupt
- Capture1 interrupt
- Capture2 Interrupt

17.4.4.2 16-BIT CORE (PIC17) CPU

Reset Conditions

All Reset conditions are supported by the MPLAB SIM simulator.

The Time out (TO) and Power-Down (PD) bits in the CPUSTA register reflect appropriate Reset condition. This feature is useful for simulating various power-up and time-out forks in the user code.

A $\overline{\text{MCLR}}$ Reset during normal operation or during Sleep can easily be simulated by driving the $\overline{\text{MCLR}}$ pin low (and then high) via stimulus.

Sleep

When executing a Sleep instruction, MPLAB SIM will appear "asleep" until a wake-up from sleep condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the postscaler setting).

An example of a wake-up-from-sleep condition would be an input change on Port B. If the interrupt is enabled and the GLINTD bit is set, the processor will wake up and will resume executing from the instruction following the Sleep command. If the GLINTD = `0`, the normal interrupt response will take place.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the postscaler Configuration bits WDTPS0:1. The minimum period (with postscaler at 1:1) is approximated, to the closest instruction cycle multiple, to the value in the device data sheet. Setting the Configuration bits WDTPS0:'`1`' to '`00`' will disable the WDT.

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the postscaler.

A WDT time-out is simulated when WDT is enabled, proper postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or Reset, depending on the selection in the Break Options tab of the Settings dialog.

17.4.4.3 16-BIT CORE (PIC17) PROCESSOR MODES

The following processor modes are supported by MPLAB SIM for devices which allow them:

- Microcontroller (Default)
- Extended Microcontroller
- Microprocessor

For information on external memory, see **Section 18.6 "Using External Memory"**.

17.4.4.4 16-BIT CORE (PIC17) PERIPHERALS

Along with providing core support, MPLAB SIM supports the following peripheral modules, in addition to general purpose I/O:

- Timer0
- Timer1 and Timer2
- Timer3 and Capture
- PWM
- OSC Control of IO
- IO Ports

The delays are implemented on all peripherals, but the interrupt latency is not.

Timer0

Timer0 and the interrupt it can generate on overflow is fully supported, and will increment by the internal or external clock. Delay from external clock edge to timer increment has also been simulated, as well as the interrupt latency period. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to the stimulus file requirements.

Timer1 and Timer2

Timer1 and Timer2 in its various modes is fully supported. Delays from clock edge to increment (when configured to increment from rising or falling edge of external clock) is simulated as well as the interrupt latency periods. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to the stimulus file requirements.

Timer3 and Capture

The MPLAB simulator fully supports Timer3 and the Capture module in all of its modes. Delays from clock edge to increment (when configured in external mode), delay for capture and interrupt latency periods are fully supported. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to the stimulus file requirements.

PWM

All PWM outputs are supported (resolution greater than 1 TCY only).

OSC Control of IO

The I/O pins are controlled for I/O depending on oscillator settings and whether the oscillator uses the pins.

IO Ports

All I/O ports are supported for input/output, interrupt and change events. The I/O is supported to a register level and not a pin level. See **Chapter 20. "Using Stimulus – PIC17 Devices"**.

17.4.5 16-Bit Core Device Simulation – PIC18

The following topics discuss the enhanced 16-bit core device features modeled in the simulator.

- 16-bit Core (PIC18) Interrupts
- 16-bit Core (PIC18) CPU
- 16-bit Core (PIC18) Processor Modes
- 16-Bit Core (PIC18) Special Registers
- 16-bit Core (PIC18) Peripherals

17.4.5.1 16-BIT CORE (PIC18) INTERRUPTS

The following interrupts are supported:

- External interrupt on INT pin
- TMR0 overflow interrupt
- External interrupt on RA0 pin
- Port B input change interrupt
- Timers/Counters
- CCP/ECCP
- PWM
- A/D
- Comparators
- EEPROM
- UARTS

17.4.5.2 16-BIT CORE (PIC18) CPU

Reset Conditions

All Reset conditions are supported by the MPLAB SIM simulator.

The Time out (TO) and Power-Down (PD) bits in the RCON register reflect appropriate Reset condition. This feature is useful for simulating various power-up and time-out forks in the user code.

You cannot Reset by toggling $\overline{\text{MCLR}}$ using stimulus control. You must use *Debugger> Reset>MCLR Reset*.

Sleep

When executing a Sleep instruction, MPLAB SIM will appear "asleep" until a wake-up from sleep condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the pre/post-scaler setting).

An example of a wake-up-from-sleep condition would be an input change on Port B. If the interrupt is enabled (RBIE) and the priority bit (RBIP) is selected, the processor will wake up and will resume executing from the instruction following the Sleep command.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the pre/postscaler Configuration bits WDTPS0:2. The minimum period (with pre/postscaler at 1:1) may be set on the **Break Options** tab of the Settings dialog (*Debugger>Settings*).

Setting the Configuration bit WDTEN to '0' will disable the WDT, unless it is enabled by the SWDTEN bit of the WDTCON register. Setting the Configuration bit WDTEN to '1' will enable the WDT regardless of the value of the SWDTEN bit.

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the pre/postscaler.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or Reset, depending on the selection in the **Break Options** tab of the Settings dialog.

17.4.5.3 16-BIT CORE (PIC18) PROCESSOR MODES

The following processor modes are supported by MPLAB SIM for devices which allow them:

- Microcontroller (Default)
- Extended Microcontroller
- Microprocessor
- Microprocessor with Boot Block

For information on external memory, see **Section 18.6 "Using External Memory"**.

17.4.5.4 16-BIT CORE (PIC18) SPECIAL REGISTERS

To aid in debugging this device, certain items that are normally not observable have been declared as "special" registers. Prescalers cannot be declared in user code as "registers", so the following special symbols are available in the Special Function Registers window:

- T0PRE (Prescaler for Timer 0)
- WDTPRE (Prescaler for WDT)

17.4.5.5 16-BIT CORE (PIC18) PERIPHERALS

Along with core support, MPLAB SIM supports the following peripheral modules, in addition to general purpose I/O:

- Timers
- CCP/ECCP
- PWM
- Comparators
- A/D Converter
- USART
- EEPROM Data Memory
- Peripheral Remapping Functionality
- OSC Control of IO
- IO Ports

The delays and interrupt latency are implemented on all peripherals.

Timers

Timer0 (and the interrupt it can generate on overflow) is fully supported, and will increment by the internal or external clock. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to stimulus requirements.

All Other Timers in their various modes are supported, except for modes using an external crystal. MPLAB SIM supports Timer interrupts generated on overflow, and interrupts generated by wake-up from sleep. Although the external oscillator is not simulated, a clock stimulus can be assigned to those pins. The prescaler for TimerX is made accessible as TXPRE in the SFR window.

CCP/ECCP

Capture

MPLAB SIM fully supports capture and the interrupt generated.

Compare

MPLAB SIM supports compare mode, its interrupt and the special event trigger (resetting a Timer by CCP).

PWM

PWM output is fully supported (resolution greater than 1 TCY only) except for Dead Band delay.

PWM

PWM output is fully supported (resolution greater than 1 TCY only) except for Dead Band delay.

Comparators

Only comparator modes that do not use Vref are simulated in MPLAB SIM.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

Toggling a comparator pin will not work since toggling involves reading a value and inverting it. Since the value always reads '0', the bit/pin never toggles. Instead, use two statements to toggle a comparator pin, e.g.,

- RA1 set low
- RA1 set high

A/D Converter

All the registers, timing function and interrupt generation are implemented. The simulator, however, does not load any meaningful value into the A/D result registers (ADRES) at the end of a conversion.To load meaningful data, use an injection file (see **Section 19.4.6 "Register Injection Tab"**). A read of the A/D registers will load this data into the registers.

Note:	If you have trouble with I/O pins on processors that have A/D, make certain that the ADCON registers are configuring those pins for digital I/O rather than for analog input. For most processors, these default to analog inputs and the associated pins cannot be used for I/O until the ADCON registers are set properly.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

USART

USART functionality is supported. For more information, see **Section 18.7 "Using a USART/UART"**.

The following limitations apply to USART operations:

- The receiver register must be stimulated using a **message-based** stimulus file. The characters are clocked into the receiver register at the default or current baud rate, starting at the time defined within the message-based stimulus file. If the receiver is not enabled when the data starts arriving, the data is lost.

EEPROM Data Memory

The EEPROM data memory is fully simulated. The registers and the read/write cycles are fully implemented. The write cycle time is device dependent (to nearest instruction cycle multiple).

The simulator simulates the functions of WRERR and WREN control bits in the EECON1 register. WRERR can be set using stimulus for testing purposes.

Peripheral Remapping Functionality

The remapping of I/O pins is fully supported. The lock/unlock functionality to change the mappings is also supported.

OSC Control of IO

The I/O pins are controlled for I/O depending on oscillator settings and whether the oscillator uses the pins.

IO Ports

All I/O ports are supported for input/output, interrupt and change events. The I/O is supported to a register level and not a pin level. See **Chapter 19. "Using Stimulus"**.

17.4.6 24-Bit Core Device Simulation – PIC24 MCUs / dsPIC DSCs

The following topics discuss the PIC24 MCU and dsPIC DSC device features modeled in the simulator.

- 24-Bit Exceptions (Traps/Interrupts)
- 24-Bit System Integration Block
- 24-Bit Peripherals

17.4.6.1 24-BIT EXCEPTIONS (TRAPS/INTERRUPTS)

The simulator supports all core and peripheral traps and interrupts, even if the peripheral is currently not supported.

The dsPIC DSC core features a vectored exception processing structure for up to 8 traps and 54 interrupts or 62 total independent vectors. Each interrupt is prioritized, based on a user-assigned priority between 1 and 7 (1 being the lowest priority and 7 being the highest). If a conflict occurs (two interrupts at the same priority), interrupt service will be based on a predetermined 'natural order' which is hardware-based.

DMA Transfer Request

In the simulator, you may initiate a DMA transfer request by setting the interrupt source flag in software. For example, if Timer2 is chosen as the source of the DMA transfer request, setting the T2IF bit in the IFS0 register will trigger a DMA transfer.

17.4.6.2 24-BIT SYSTEM INTEGRATION BLOCK

Reset Sources

All Reset sources are supported by the MPLAB SIM simulator.

Status bits from the RCON register are set or cleared differently in different Reset situations, as indicated in device data sheet. These bits are used in software to determine the nature of the Reset.

A $\overline{\text{MCLR}}$ Reset during normal operation or during Sleep/Idle can easily be simulated by driving the $\overline{\text{MCLR}}$ pin low (and then high) via stimulus or by selecting *Debugger> Reset>MCLR Reset*.

Sleep/Idle

When executing a `PWRSAV` instruction, MPLAB SIM will appear "asleep" or "idle" until a wake-up condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the pre/post-scaler setting).

An example of a wake-up-from-sleep condition would be Timer1 wake up from sleep. In this case, when the processor is asleep, Timer1 would continue to increment until it matches the period counter. If the interrupt is enabled, the timer will wake the processor and branch to the interrupt vector.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The Watchdog Timer can be "Enabled" or "Disabled" through a Configuration bit (FWDTEN) in the Configuration register FWDT. Setting FWDTEN = `1` enables the Watchdog Timer. Setting FWDTEN = `0` allows user software to enable/ disable the Watchdog Timer via the SWDTEN (RCON<5>) control bit.

The period of the WDT is determined by the pre/postscaler settings in the FWDT register. The minimum period (with pre/postscaler at 1:1) may be set on the **Break Options** tab of the Settings dialog (*Debugger>Settings*).

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the pre/postscalers.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or Reset, depending on the selection in the **Break Options** tab of the Settings dialog.

17.4.6.3 24-BIT PERIPHERALS

MPLAB SIM supports the following peripherals:

- Input Capture/Output Compare
- Change Notify
- Interrupts
- OSC Control of IO
- Timers
- IO Ports
- Comparators
- PWM
- A/D Converter
- UART
- Peripheral Pin Mapping Support
- DMA Mode
- EEPROM Data Memory

The delays and interrupt latency are implemented on all peripherals.

Input Capture/Output Compare

All Input Capture and Output Compare modules are fully supported in all modes.

Change Notify

All Change Notify functionality and interrupts are supported.

Interrupts

All interrupts are supported. Both vector tables are supported. All interrupts including priority settings are simulated.

OSC Control of IO

The I/O pins are controlled for I/O depending on oscillator settings and whether the oscillator uses the pins.

Timers

All timers are fully supported in all modes. External clock must be a minimum of 1 Tcy. Timers can be combined to create 32-bit counters.

IO Ports

All I/O ports are supported for input/output, interrupt and change events. The I/O is supported to a register level and not a pin level. See **Chapter 19. “Using Stimulus”**.

Comparators

Only comparator modes that do not use Vref are simulated in MPLAB SIM.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as ‘0’, as required for analog, although injected stimulus may have actually changed the value to ‘1’.

Toggling a comparator pin will not work since toggling involves reading a value and inverting it. Since the value always reads ‘0’, the bit/pin never toggles. Instead, use two statements to toggle a comparator pin, e.g.,

- RA1 set low
- RA1 set high

PWM

PWM output is fully supported (resolution greater than 1 TCY only) except for Dead Band delay.

A/D Converter

All the registers, timing function and interrupt generation are implemented. To load meaningful data into the A/D result buffer at the end of a conversion, use an injection file (see **Section 19.4.6 "Register Injection Tab"**).

Note:	If you have trouble with I/O pins on processors that have A/D, make certain that the ADPCFG and TRIS registers are configuring those pins for digital I/O rather than for analog input. For most processors, these default to analog inputs and the associated pins cannot be used for I/O until these registers are set properly.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

UART

UART functionality is supported. For more information, see **Section 18.7 "Using a USART/UART"**.

The following limitations apply to UART operations:

- The receiver register must be stimulated using a **message-based** stimulus file. The characters are clocked into the receiver register at the default or current baud rate, starting at the time defined within the message-based stimulus file. If the receiver is not enabled when the data starts arriving, the data is lost. If the receiver is enabled the characters are read into the FIFO buffer until the FIFO is full. If the characters in the FIFO are not read out in time the remaining characters in the stimulus file will be lost and the OV status bit will be set.
- The transmitter register must be captured using an on-demand stimulus file. When the transmitter is enabled, characters written to the transmitter register are clocked into the transmitter FIFO at the current baud rate. When a character becomes available in the FIFO, it is written to the on-demand response file.
- The receiver and transmitter FIFOs are not visible to the user.
- Certain errors can occur due to the fact that the receiver register is stimulated from an on-demand file. The errors are not simulated and receiver buffer overrun error.

Peripheral Pin Mapping Support

The remapping of I/O pins is fully supported. The lock/unlock functionality to change the mappings is also supported.

DMA Mode

The DMA is fully supported for all transfer modes.

EEPROM Data Memory

The EEPROM data memory is fully simulated. The registers and the read/write cycles are fully implemented. The write cycle time is device dependent (to nearest instruction cycle multiple).

The simulator simulates the functions of WRERR and WREN control bits in the EECON1 register. WRERR can be set using stimulus for testing purposes.

17.4.7 32-Bit Core Device Simulation – PIC32MX

The following topics discuss the PIC32 MCU device features modeled in the simulator.

- 32-bit Exceptions (Traps/Interrupts)
- 32-bit Core (PIC328) CPU
- 32-bit Core (PIC32) Processor Modes
- 32-bit Core (PIC32) Peripherals

17.4.7.1 32-BIT EXCEPTIONS (TRAPS/INTERRUPTS)

The simulator supports all core and peripheral traps and interrupts, even if the peripheral is currently not supported.

The PIC32 MCU core features a vectored exception processing structure for up to 96 interrupt sources and 63 independent vectors. Each interrupt has group priorities, based on a user-assigned priority between 1 and 7 (1 being the lowest priority and 7 being the highest). In addition there are 2 bits for sub priority within each group, also based on a user-assigned priority between 0 and 3 (0 being the lowest priority and 3 being the highest). Interrupts within a group will not preempt each other. If a conflict occurs (two interrupts at the same priority), interrupt service will be based on a predetermined 'natural order' which is hardware-based, or the user can use the sub priority bits to change the natural order. Traps are handled by the MIPS Core registers.

17.4.7.2 32-BIT CORE (PIC328) CPU

Reset Conditions

All Reset conditions are supported by the MPLAB SIM simulator.

The Time out (TO) and Power-Down (PD) bits in the RCON register reflect appropriate Reset condition. This feature is useful for simulating various power-up and time-out forks in the user code.

You cannot Reset by toggling $\overline{\text{MCLR}}$ using stimulus control. You must use *Debugger>Reset>MCLR Reset*.

Sleep

When executing a Sleep instruction, MPLAB SIM will appear “asleep” until a wake-up from sleep condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the pre/postscaler setting).

An example of a wake-up-from-sleep condition would be an input change on Port B. If the interrupt is enabled (RBIE) and the priority bit (RBIP) is selected, the processor will wake up and will resume executing from the instruction following the Sleep command.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the pre/postscaler Configuration bits WDTPS0:2. The minimum period (with pre/postscaler at 1:1) may be set on the **Break Options** tab of the Settings dialog (*Debugger>Settings*).

Setting the Configuration bit WDTEN to ‘0’ will disable the WDT, unless it is enabled by the SWDTEN bit of the WDTCON register. Setting the Configuration bit WDTEN to ‘1’ will enable the WDT regardless of the value of the SWDTEN bit.

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the pre/postscaler.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or Reset, depending on the selection in the **Break Options** tab of the Settings dialog.

17.4.7.3 32-BIT CORE (PIC32) PROCESSOR MODES

The following processor modes are supported by MPLAB SIM for devices which allow them:

- Microcontroller (Default)
- Extended Microcontroller
- Microprocessor
- Microprocessor with Boot Block

For information on external memory, see **Section 18.6 "Using External Memory"**.

17.4.7.4 32-BIT CORE (PIC32) PERIPHERALS

The following PIC32MX peripheral modules are simulated:

- Input Capture/Output Compare
- Change Notify
- Interrupts
- Timers
- IO Ports
- Comparators
- PWM
- A/D Converter
- UART

Input Capture/Output Compare

All Input Capture modules and Output Compare modules are fully supported in all modes.

Change Notify

All Change Notify functionality and interrupts are supported.

Interrupts

All interrupts are supported. Both vector tables are supported. All interrupts including priority settings are simulated.

Timers

All timers are fully supported in all modes. External clock must be a minimum of 1 Tcy. Timers can be combined to create 64-bit counters.

IO Ports

All I/O ports are supported for input/output, interrupt and change events. The I/O is supported to a register level and not a pin level. See **Chapter 19. "Using Stimulus"**.

Comparators

Only comparator modes that do not use Vref are simulated in MPLAB SIM.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

Toggling a comparator pin will not work since toggling involves reading a value and inverting it. Since the value always reads '0', the bit/pin never toggles. Instead, use two statements to toggle a comparator pin, e.g.,

- RA1 set low
- RA1 set high

PWM

PWM output is fully supported (resolution greater than 1 TCY only) except for Dead Band delay.

A/D Converter

All the registers, timing function and interrupt generation are implemented. The simulator, however, does not load any meaningful value into the A/D result register (ADRES) at the end of a conversion unless an injection file has been attached to ADRES (L). To load meaningful data, use an injection file (see **Section 19.4.6 "Register Injection Tab"**).

Note:	If you have trouble with I/O pins on processors that have A/D, make certain that the A/D pin control registers (ADCON registers) are configuring those pins for digital I/O rather than for analog input. For most processors, these default to analog inputs and the associated pins cannot be used for I/O until the ADCON (or ADCON1) register is set properly.

Because simulation is to the register level, and not the pin level, bit/pin names will be read as '0', as required for analog, although injected stimulus may have actually changed the value to '1'.

UART

UART functionality is supported. For more information, see **Section 18.7 "Using a USART/UART"**.

The following limitations apply to UART operations:

- The receiver register must be stimulated using a **message-based** stimulus file. The characters are clocked into the receiver register at the default or current baud rate, starting at the time defined within the message-based stimulus file. If the receiver is not enabled when the data starts arriving, the data is lost. If the receiver is enabled the characters are read into the FIFO buffer until the FIFO is full. If the characters in the FIFO are not read out in time the remaining characters in the stimulus file will be lost and the OV status bit will be set.
- The transmitter register must be captured using an on-demand stimulus file. When the transmitter is enabled, characters written to the transmitter register are clocked into the transmitter FIFO at the current baud rate. When a character becomes available in the FIFO, it is written to the on-demand response file.
- The receiver and transmitter FIFOs are not visible to the user.
- Certain errors can occur due to the fact that the receiver register is stimulated from an on-demand file. The errors are not simulated and receiver buffer overrun error.

17.5 SIMULATOR EXECUTION

MPLAB SIM operation is specified in the following topics.

- Execution Speed
- Execution on Instruction Cycle Boundaries
- I/O Timing

17.5.1 Execution Speed

When MPLAB SIM is simulating running in real time, instructions are executing as quickly as the PC's CPU will allow. This is usually slower than the actual device would run at its rated clock speed.

The speed at which the simulator runs depends on the speed of your computer and how many other tasks you have running in the background. The software simulator must update all of the simulated registers and RAM, as well as monitor I/O, set and clear flags, check for break and trace points in software and simulate the instruction with instructions being executed on your computer's CPU.

The execution speed of a discrete-event software simulator is orders of magnitude less than a hardware oriented solution. Slower execution speed may be viewed as a handicap or as a tool. The simulator attempts to provide the fastest possible simulation cycle, and depending upon the mode of operation, can operate on the order of milliseconds per instruction.

Note:	Often loops will be used in your code to generate timing delays. When using the simulator, you might wish to decrease these time delays or conditionally remove those sections of your code with "IFDEF" statements to increase simulation speed.

Turning off simulator trace will increase simulation speed by up to 50%. Therefore, use the tracing function only as needed. See **Section 18.3 "Using Simulator Trace"**.

In general, when this discussion says "real time" and you are in the simulator mode, the software simulation is executing simulated code as fast as your PC can simulate the instructions.

17.5.2 Execution on Instruction Cycle Boundaries

The simulator executes on instruction cycle boundaries, and resolutions shorter than one instruction cycle (TCY) can not be simulated. The simulator is a discrete-event simulator where all stimuli are evaluated, and all responses are generated, at instruction boundaries, which occur at TCY = 4*Tosc or TCY = 2*Tosc depending on the device, where Tosc is the input clock period. Therefore, some physical events can not be accurately simulated. These fall into the following categories:

- Purely asynchronous events
- Events that have periods shorter than one instruction cycle

The net result of instruction boundary simulation is that all events get synchronized at instruction boundaries, and events smaller than one instruction cycle are not recognized.

The following functions are **not** supported by the simulation on instruction cycle boundaries:

- Clock pulse inputs smaller than one cycle even though timer prescalers are capable of accepting clock pulse inputs smaller than one cycle.
- PWM output pulse resolution less than one cycle.

- In unsynchronized counter mode, clock inputs smaller than one cycle cannot be used.
- Showing the oscillator waveform on RC0/RC1 pins.
- Serial I/O to the pin level.

17.5.3 I/O Timing

External timing in the simulator is processed only once during each instruction cycle. Transient signals, such as a spikes on MCLR smaller than an instruction cycle, will not be simulated.

Note:	Stimulus is injected into the simulator prior to the next instruction cycle.

NOTES:

Chapter 18. Getting Started with MPLAB SIM

18.1 INTRODUCTION

If you are new to MPLAB IDE and the simulator, please refer to the tutorials in the MPLAB IDE documentation to help you set up MPLAB IDE for use with MPLAB SIM. Select *Help>Topics* and then "MPLAB IDE Help" in the dialog, or click the link below (and then "Open" if a security dialog appears).

- MPLAB IDE Help

Once you are familiar with basic simulator operation under MPLAB IDE, the following topics will help you work with simulator-specific features.

- Using Stimulus
- Using Simulator Trace
- Using the Simulator Logic Analyzer
- Using the Stopwatch
- Using External Memory
- Using a USART/UART
- Using Code Coverage

18.2 USING STIMULUS

Simulator stimulus is a simulation of hardware signals. It is a powerful tool that allows you to control inputs to the simulator and exercise your application code. Stimulus control is detailed and covered in separate chapters:

- **Chapter 19. "Using Stimulus"**
- **Chapter 20. "Using Stimulus – PIC17 Devices"**

18.3 USING SIMULATOR TRACE

Tracing allows you to record the step-by-step execution of your code and examine this recording.

Note:	Simulation speed is reduced by up to 50% when using trace.

The default is set for 64K records and is recorded internally in RAM to create fast access to the trace buffer. For records larger than 64K, a RAM file buffer (essentially disk space) is used and the speed is much slower. Except when there is no alternative for capturing a large buffer of data, it is recommended that records are kept to less than 64K to minimize the delay.

Open the Trace window from *View>Simulator Trace*. For more on this window, see MPLAB IDE documentation on the Trace Memory window. To set up trace in general, select *Debugger>Settings>OSC/Trace*. In the Trace Options pane, select "Trace All".

To set up a limited trace, open the file (editor) window containing application code and select either "Add Filter-in Trace" or "Add Filter-out Trace" from the right mouse menu. For more information on filter trace, see MPLAB Editor on-line help.

18.4 USING THE SIMULATOR LOGIC ANALYZER

The simulator logic analyzer allows you to graphically view digital pin signals over a defined time period. The logic analyzer uses the same data as the simulator trace, from the trace buffer. Therefore, trace must be enabled (*Debugger>Settings*, either the **Osc/Trace** tab or **Trace/Pins** tab, "Trace All" checkbox.)

Note: There is no logic analyzer for PIC17 devices.

Open the logic analyzer window from *View>Simulator Logic Analyzer*. For more on this window, see MPLAB IDE documentation on the Logic Analyzer window.

18.5 USING THE STOPWATCH

The stopwatch is useful for simple timing between program halts. Reach this dialog from *Debugger>Stopwatch*.

Instruction Cycles and Time

The simulator updates the "Instruction Cycles" and "Time" fields, including the Time units, as your program runs.

PIC MCUs and dsPIC30F/PIC24F devices use 4 clock cycles per instruction to calculate time. dsPIC33F/PIC24H devices use 2 clock cycles per instruction to calculate time.

- Click **Synch** to synchronize the stopwatch to the total simulated values.
- Click **Zero** to set the "Instruction Cycles" and "Time" values to zero at any time.

Also, stopwatch time may be reset to zero by issuing a Processor Reset (*Debugger>Reset>Processor Reset*).

Processor Frequency

This field, including Frequency units, will be set from the **Osc/Trace** (or **Clock** for PIC17 MCUs) tab of the *Debugger>Settings* dialog.

18.6 USING EXTERNAL MEMORY

Simulating a device that uses external memory requires the following steps:

1. Setting Configuration bits to enable external memory modes – *Configure>Configuration Bits*
2. Setting external memory usage in MPLAB IDE – *Configure>External Memory*

For more information on external memory and MPLAB IDE, see MPLAB IDE documentation on external memory setup and use.

For the simulator, external memory is simulated the same as on-chip memory – with PC memory. Therefore, there are no communication delays as there may be with hardware and use of an external bus interface.

18.7 USING A USART/UART

For devices that have a USART or UART peripheral, there are two ways in which to simulate this operation:

1. Using the UART1 I/O Tab – USART/UART simulation may be enabled and set up using the **UART1 IO** tab of the Settings dialog (*Debugger>Settings*). For more on this tab, see **Section 22.4.4 "UART1 IO Tab"**.
2. Using SCL Stimulus – USART/UART simulation may accomplished using SCL stimulus to USART registers. For more on SCL stimulus, see **Chapter 19. "Using Stimulus"**.

For an example of simulating USART/UART operation, see below.

- PIC18F MCU USART Example – Setup
- PIC18F MCU USART Example – UART IO Tab
- PIC18F MCU USART Example – SCL Stimulus
- Updated PICDEM™ 2 Example Code

18.7.1 PIC18F MCU USART Example – Setup

Follow the steps below to set up the example.

1. Example code for demonstrating USART operation in the simulator is based on code found on the Microchip website under "PICDEM™ 2 example code" as "USART demo for the 18CXXX" (`usart.asm`). The updated code (`usart2.asm`) is listed in **Section 18.7.4 "Updated PICDEM™ 2 Example Code"**.
2. In MPLAB IDE, use the Project Wizard to create a project.
   - Select PIC18F452 as the device.
   - Select "Microchip MPASM Toolsuite" as the active toolsuite. Make sure the paths to the executables are correct.
   - Place the project in its own folder and name the project "usart".
   - Add the file `usart2.asm` to the project. You will not use the linker in this example, so no linker script file is needed.
3. Disable the Watchdog Timer so it will not interrupt program execution.
   - Select *Configure>Configuration Bits*. Deselect the checkbox "Configuration Settings set in code".
   - Under the "Settings" column, select Watchdog Timer as "Disabled".
4. Build the project (*Project>Build All*). Build results are displayed on the **Build** tab of the Output window. Fix any reported errors.

18.7.2 PIC18F MCU USART Example – UART IO Tab

For MPLAB IDE v7.20 and above, follow the steps below to use the **UART IO** tab of the Settings dialog to simulate USART/UART operation.

1. Once the project is completed, you will need an input file to simulate USART signal input.
   - Select *File>Open* to open an editor window.
   - Enter message-based data for RCREG input (see **Section 19.4.6.3 "Message-Based Data File Description"** for information on this format). Example input might be:
     ```
     //single-packet example
     wait 20 ms
     10 20 34
     ```
   - Select *File>Save* and name the file `input.txt`.
2. Now select the UART IO tab to enable and set up the USART peripheral.
   - Select *Debugger>Settings*, **UART IO** tab.
   - Check "Enable UART1 I/O".
   - Select `input.txt` as the "Input" file. Check "Rewind Input" so the program will read the input file again once it has reached the end-of-file.
   - Click "Window" under "Output" to display USART output on the **SIM UART1** tab of the Output window.
3. Build the project again and run it to see the output.
   - Select *Project>Build All*.
   - Run the program to see the output on the **SIM UART1** tab of the Output window.

18.7.3 PIC18F MCU USART Example – SCL Stimulus

To use SCL Stimulus input to simulate USART/UART operation, follow the steps below.

> **Note:** The "Input File:" field must be blank on the **UART IO** tab, Settings dialog (*Debugger>Settings*) or this method will not work.

1. Once the project is completed, you will need an input file to simulate USART signal input.
   - Select *File>Open* to open an editor window.
   - Enter message-based data for RCREG input (see **Section 19.4.6.3 "Message-Based Data File Description"** for information on this format). Example input might be:
     ```
     //single-packet example
     wait 20 ms
     10 20 34
     ```
   - Select *File>Save* and name the file `input.txt`.
2. Select *Debugger>Stimulus>New Workbook*. Click on the **Register Injection** tab.
   - Under "Register", select "RCREG".
   - Under "Trigger", "Message" will be selected. (There are no other choices.)
   - Under "Data Filename", locate and select `input.txt`.
   - Under "Wrap", select "Yes".
   - Under "Format", "Pkt", or packet, will be selected. (There are no other choices.)
   - Click **Apply**.
   - Save the workbook (`usart.sbs`).

3. Open a Watch window to see the USART registers update.
   - Select *View>Watch*.
   - In the Watch window, add the SFRs RCREG and TXREG.
4. Build the project again and run it to see the output in the Watch window.
   - Select *Project>Build All*.
   - Place a breakpoint at the beginning of the interrupt vector (`IntVector`) code, i.e., at line
     "`btfss  PIR1,RCIF  ; Did USART cause interrupt?`".
   - Run the code till the breakpoint halt to see the value of RCREG change.
   - Step to see the value of TXREG change to match the injected value of RCREG.
   - Run and step again to see the values change as per the input file.
   - Select *Debugger>Settings*, **UART IO** tab.
   - Check "Enable UART1 I/O".
   - Click "Window" under "Output" to display USART output on the **SIM UART1** tab of the Output window.

18.7.4 Updated PICDEM™ 2 Example Code

```
;
; Software License Agreement
;
; The software supplied herewith by Microchip Technology Incorporated
; (the "Company") for its PIC® Microcontroller is intended and
; supplied to you, the Company's customer, for use solely and
; exclusively on Microchip PIC Microcontroller products. The
; software is owned by the Company and/or its supplier, and is
; protected under applicable copyright laws. All rights are reserved.
;  Any use in violation of the foregoing restrictions may subject the
; user to criminal sanctions under applicable laws, as well as to
; civil liability for the breach of the terms and conditions of this
; license.
;
; THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
; WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
; TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
; IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
; CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
;
;*********************************************************
;   PIC18F452 EXAMPLE CODE FOR PICDEM 2
;
;   TITLE: USART Demonstration
;   FILENAME: usart.asm
;   REVISION HISTORY:  A 5/13/00 jb   format change
;                      B 2/28/05 em   update for example
;   HARDWARE: PICDEM 2 board
;   FREQUENCY: 4MHz
;
;*********************************************************
; This program demonstrates basic functionality of the USART.
;
; Port B is connected to 8 LEDs.
; When the PIC18F452 receives a word of data from
; the USART, the value is displayed on the LEDs and
; is retransmitted to the host computer.
```

```
;
; Set terminal program to 9600 baud, 1 stop bit, no parity

   list p=18f452       ; set processor type
   list n=0            ; supress page breaks in list file
   include <p18f452.inc>

;*************************************************************
; Reset and Interrupt Vectors

   org    00000h       ; Reset Vector
   goto   Start

   org    00008h       ; Interrupt vector
   goto   IntVector

;*************************************************************
; Program begins here

   org    00020h       ; Beginning of program EPROM
Start
   clrf   LATB         ; Clear PORTB output latches
   clrf   TRISB        ; Config PORTB as all outputs
   bcf    TRISC,6      ; Make RC6 an output

   movlw  19h          ; 9600 baud @4MHz
   movwf  SPBRG

   bsf    TXSTA,TXEN   ; Enable transmit
   bsf    TXSTA,BRGH   ; Select high baud rate

   bsf    RCSTA,SPEN   ; Enable Serial Port
   bsf    RCSTA,CREN   ; Enable continuous reception

   bcf    PIR1,RCIF    ; Clear RCIF Interrupt Flag
   bsf    PIE1,RCIE    ; Set RCIE Interrupt Enable
   bsf    INTCON,PEIE  ; Enable peripheral interrupts
   bsf    INTCON,GIE   ; Enable global interrupts

;*************************************************************
; Main loop

Main
   goto   Main         ; loop to self doing nothing

;*************************************************************
; Interrupt Service Routine

IntVector
; save context (WREG and STATUS registers) if needed.

   btfss  PIR1,RCIF  ; Did USART cause interrupt?
   goto   OtherInt   ; No, some other interrupt

   movlw  06h        ; Mask out unwanted bits
   andwf  RCSTA,W    ; Check for errors
   btfss  STATUS,Z   ; Was either error status bit set?
   goto   RcvError   ; Found error, flag it

   movf   RCREG,W    ; Get input data
```

```
   movwf  LATB       ; Display on LEDs
   movwf  TXREG      ; Echo character back
   goto   ISREnd     ; go to end of ISR, restore context, return

RcvError
   bcf    RCSTA,CREN ; Clear receiver status
   bsf    RCSTA,CREN
   movlw  0FFh       ; Light all LEDs
   movwf  PORTB
   goto   ISREnd     ; go to end of ISR, restore context, return

OtherInt
   goto   $          ; Find cause of interrupt and service it
                     ; before returning from interrupt. If not,
                     ; the same interrupt will re-occur
                     ; as soon as execution returns to
                     ; the interrupted program.

ISREnd
; Restore context if needed.
   retfie

   end
```

18.8 USING CODE COVERAGE

The code coverage feature provides visibility as to what portions of the code are being executed. This code is checkmarked in the Program Memory window.

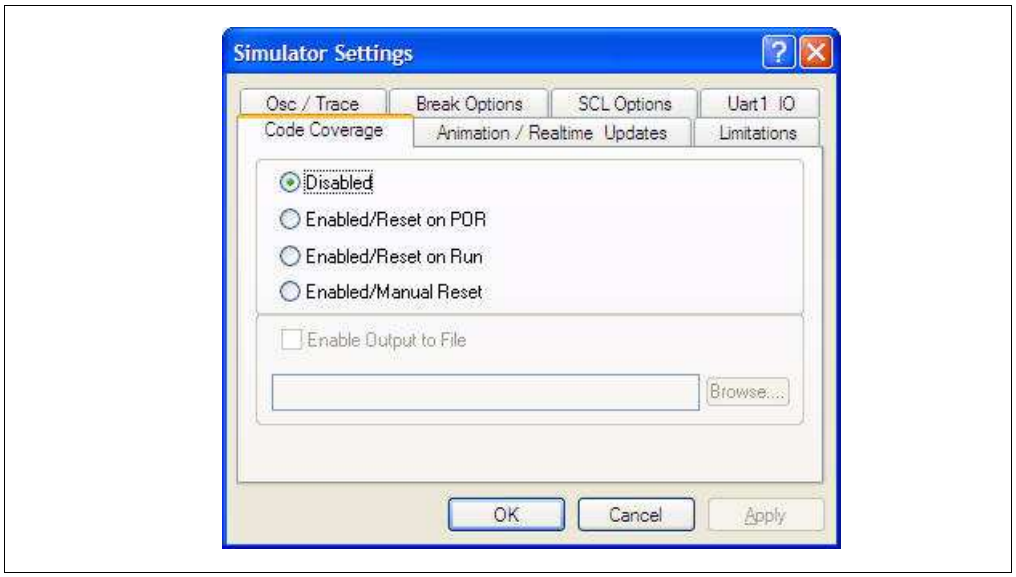
Code coverage differs from traced code in the following way:

- Trace displays code that has been executed and tells when it was executed.
- Code coverage marks code that has been executed, but does not display when it was executed.

This feature works by latching addresses of the opcode as it executes. With code coverage enabled, the next halt (step, software breakpoint or halt command) will cause ROM locations that have been executed to be checkmarked in the Program Memory window.

Select *Debugger>Settings* and then click the **Code Coverage** tab.

FIGURE 18-1: MPLAB SIMULATOR – CODE COVERAGE TAB



18.8.1 Enabling/Disabling Code Coverage

If "Enabled/Reset on POR" is selected, code coverage is reset when power-on reset is applied.

If "Enabled/Reset on Run" is selected, code coverage is reset every time run is executed within the simulator.

If "Enabled/Manual Reset" is selected, code coverage is reset when the Clear Code Coverage option is explicitly selected under the Debugger menu.

When a reset has occurred, and simulation has run and then halted, all executed program memory locations appear checkmarked in the Program Memory window. They will remain checkmarked until the next reset. Single stepping will also checkmark addresses that are executed.

To disable code coverage, select *Debugger>Settings*, **Code Coverage** tab, and select the "Disabled" option, then click **OK**.

18.8.2 Creating a Code Coverage Report

Code coverage information may be saved into a file by checking the “Enable Output to File” option on the **Code Coverage** tab. One of the Enabled options must be selected for this option to be available.

1. In the text box below the “Enable Output to File” checkbox, specify a file in which to save the information.
   - For a new file, type in the file name and path or browse to a file location. When you have reached the location at which you want the new file, type in the file name and click **Save**.
   - For an existing file, click the **Browse** button to search for the desired file.
2. Click **Apply**, **OK** or **Save**.

Whenever the simulation is halted and “Enable Output to File” is enabled, the output is regenerated and written to the file simultaneously to setting the checkmarks in the Program Memory window.

NOTES:

Chapter 19. Using Stimulus

19.1 INTRODUCTION

During simulation, the program being executed by the simulator may require stimuli from the outside. Stimulus is the simulation of hardware signals/data into the device. This stimulus could be a level change or a pulse to an I/O pin of a port. It could be a change to the values in an SFR (Special Function Register) or other data memory.

In addition, stimulus may need to happen at a certain instruction cycle or time during the simulation. Alternately, stimulus may need to occur when a condition is satisfied; for example, when the execution of program has reached a certain instruction address during simulation.

Basically, there are two types of stimulus:

- Asynchronous – A one-time change to the I/O pin or RCREG triggered by a firing button on the stimulus GUI within the MPLAB IDE.
- Synchronous – A predefined series of signal/data changes to an I/O pin, SFR or GPR (e.g., a clock cycle).

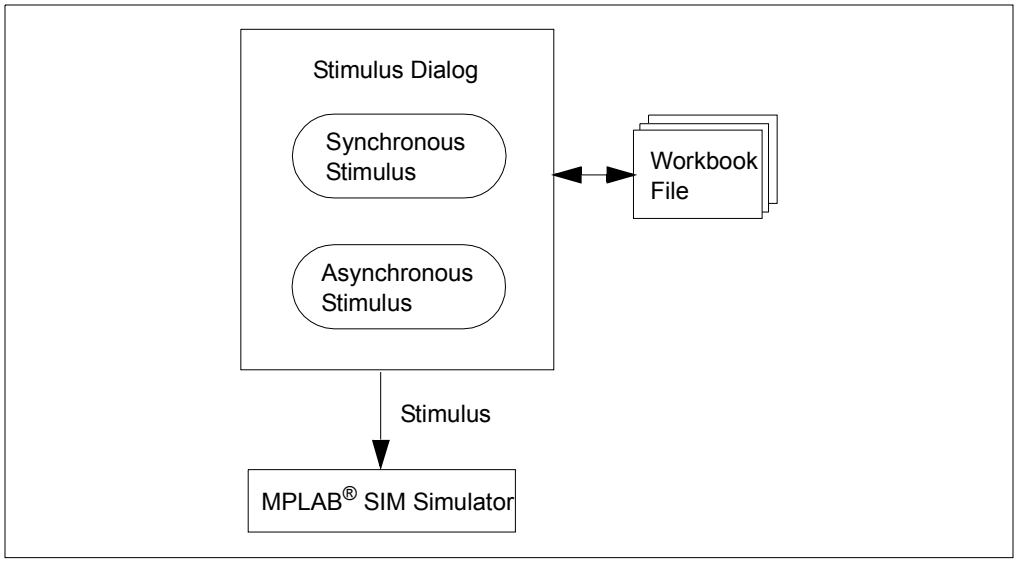
To define when, what and how external stimuli are to happen to a program, you would use the Stimulus Dialog tabs to create both asynchronous and synchronous stimulus on a stimulus workbook. Advanced users can attach and generate an SCL (Simulator Control Language) file for custom stimulus needs.

If you will be using multiple forms of stimulus input, you should be aware of input interaction (see **Section 19.6 "Stimulus Input Interaction"**).

19.2 BASIC MODE

The basic mode (Figure 19-1) is intended for most users. The stimulus settings are saved to a workbook. The asynchronous and synchronous stimuli are available for the simulator when the Stimulus dialog is open.

FIGURE 19-1: BASIC MODE

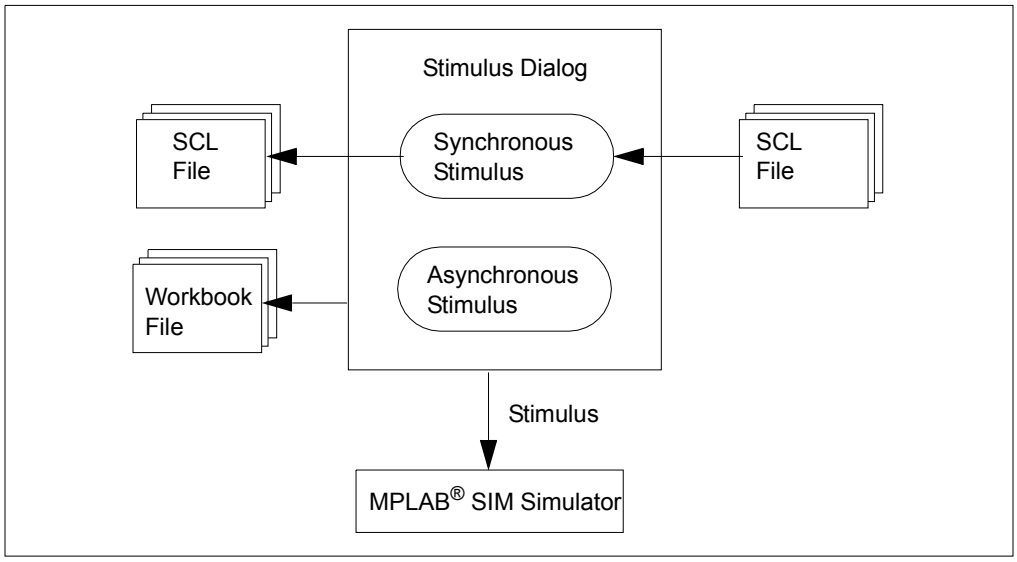


19.3 ADVANCED MODE

For advanced users (Figure 19-2), the synchronous stimulus can be exported to an SCL file. SCL files can be attached. Note that attaching an SCL file would not populate the synchronous Stimulus tabs (last five tabs). Once an SCL file is attached, the synchronous stimulus tabs are inaccessible until the SCL file is detached, but the asynchronous stimulus tab (first tab) is still accessible. Attaching an SCL file only affects the last five tabs.

The Stimulus setup and the asynchronous stimulus can be saved to a workbook. A workbook contains the asynchronous and synchronous stimulus setup as well as the SCL filename which is currently attached to the stimulus dialog, if used.

FIGURE 19-2: ADVANCED MODE



19.4 STIMULUS DIALOG

Use the Stimulus dialog to create asynchronous or synchronous stimuli. The Stimulus dialog allows you to enter stimulus information which is saved in a file called a workbook.

To open a new workbook, select *Debugger>Stimulus>New Workbook*.

To open an existing workbook for editing, select *Debugger>Stimulus>Open Workbook*.

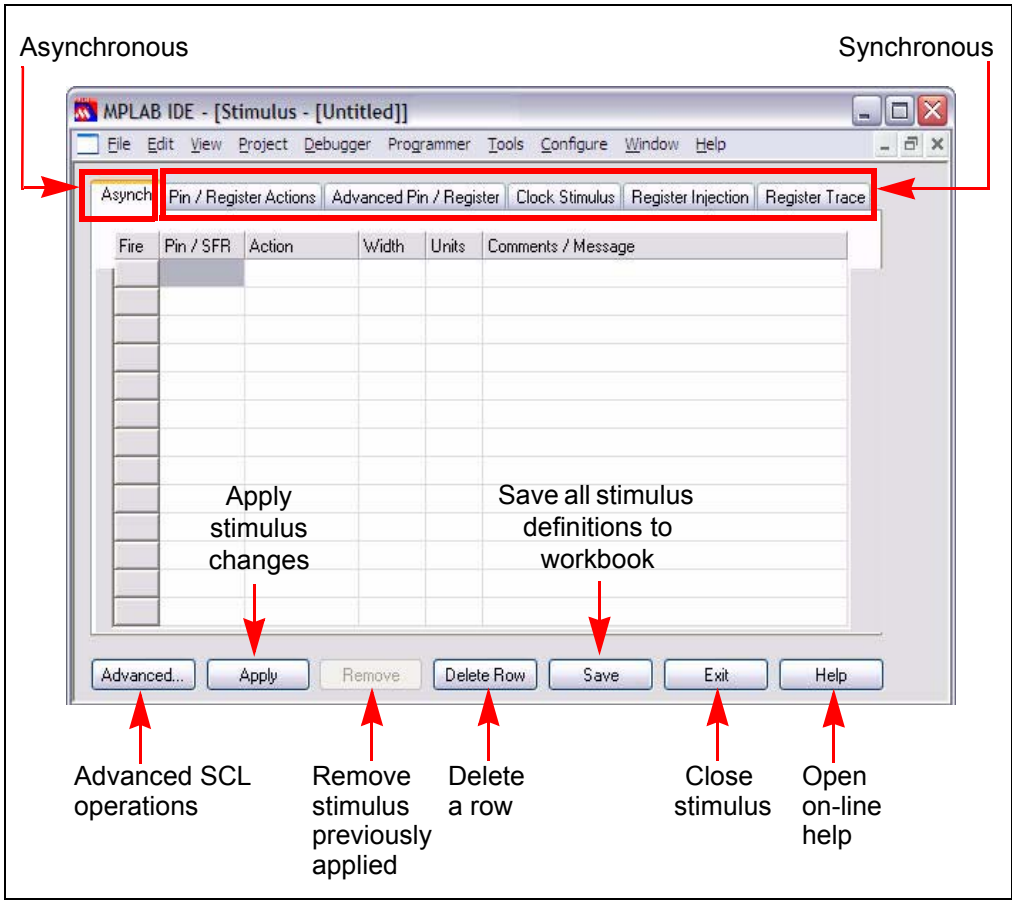
The Stimulus dialog (see Figure 19-3) has these tabs.

- Asynch Tab
- Pin/Register Actions Tab
- Advanced Pin/Register Tab
- Clock Stimulus Tab
- Register Injection Tab
- Register Trace Tab

When setting up multiple tabs, be aware of input interaction (see **Section 19.6 "Stimulus Input Interaction"**).

Note:	The Stimulus dialog must be open for stimulus to be active during simulation.

FIGURE 19-3: STIMULUS DIALOG



Default Unit Values

All tabs have these default unit values:

- Time Units are always in decimal.
- PC values are always in hex.
- Pin values are always either '0' or '1'.
- Register values are always in hex.
- Bitfield values are always in binary.

Label Constructs

Labels must be of the format:

alpha[_|alpha|numeric]

where:

- **alpha** = alphabetic character(s)
- **_** = underscore character
- **numeric** = numeric character(s)

All labels must:

- must be unique; they cannot be the same as any pin name or SFR name
- begin with an alphabetic character
- not end with an underscore (_)
- not include two underscores together

19.4.1 Asynch Tab

Note: The Stimulus dialog must be open for stimulus to be active during simulation.

Use the **Asynch** tab to control asynchronous events generated by the user.

Enter any asynchronous, or user-fired, stimulus here, row by row. To remove a row, select the row and then click **Delete Row**.

There are two types of asynchronous stimulus: regular stimulus, for most pins/SFRs, and message-based stimulus, for USART/UART SFRs.

Item	Definition
Fire	Click the button in this column corresponding to the row for which you want to trigger stimulus. Obviously, you must set up all other row items before you can use “Fire”.
Pin/SFR	Select or change the pin/SFR on which stimulus will be applied. Available pin names are listed in a drop-down list. The actual names depend on the device selected in the MPLAB® IDE.
Action	Select or change a stimulus action. **Regular Stimulus:** Choose “Set High”, “Set Low”, “Toggle”, “Pulse* High”, or “Pulse* Low”. **Message-Based Stimulus:** Choose “File Message” or “Direct Message”. “File Message” means you will use a file containing message packets. For a file with more than one packet, comments will delineate the packets. Each click of “Fire” will inject one packet, until the end-of-file is reached, where the file will automatically rewind. (For file format information, see **Section 19.4.6.3 “Message-Based Data File Description”**.) “Direct Message” means you will use the Comments/Message cell to define a one-line message packet.
Width	If “Pulse” was chosen for the Action, then you may specify or change a pulse width value here. Enter the units for this value in the next cell of the row.
Units	If “Pulse” was chosen for the Action, then you may specify or change a pulse width unit here. Enter the value for this unit in the previous cell of the row.
Comments / Message	**Regular Stimulus:** Specify or change a comment about the stimulus. **Message-Based Stimulus:** Specify or change the stimulus message. **Note:** This message must be pure data, i.e., no wait time allowed.

* “Pulse” on a pulse selection. The current pin state is read. The appropriate pulse state is applied and the pin state is returned to its previous state. Therefore, a pin which is already high and an asynchronous pulse high is applied will never change state.

19.4.2 Pin/Register Actions Tab

Basic synchronous pin and/or register actions may be entered here. For more complex actions, use the Advanced Pin/Register Tab tab.

This is the simplest time based stimulus. Some possible uses for this tab could be:

1. Initialize pin states at time 0 (zero) so when ever you re-start a simulation run the pins will be in a predetermined state after each POR.
   IO port pins do not change state on a reset and the simulator starts off with treating all IO pins as inputs of a zero state.
2. Set a register to a value at a specific time.
3. Set multiple interrupt flags at exactly the same time to see the effect within the interrupt handler for priority
4. Create a pulse train with different periods of a pulse over time, or an irregular wave form based on run time
5. Repeat a sequence of events for endurance testing

To enter data in this tab:

1. Select the unit of time in the "Time Units" list box that you will use to trigger all stimulus.
2. Click on the text that says "Click here to Add Signals" to open the Add/Remove Pin/Registers dialog (see **Section 19.4.4 "Add/Remove Pin/Registers Dialog"**). In that dialog, you select the pins, registers or other signals to which you will apply stimulus. These selections will become the titles of the columns.
3. Fill out each row, entering a trigger time ("Time") and value for each pin/register column. Trigger time for each row is accumulative time (since the simulation began), not interval time between adjacent rows.
4. To delete a row, select it and then click **Delete Row**.
5. Check the checkbox "Repeat after X (decimal)" to repeat the stimulus on the tab after the last stimulus has occurred. Specify a delay interval for when to repeat the stimulus.
6. To restart at a specific time, make a selection from the "Restart at: (decimal)" list box. The list box selections are determined by the trigger times (in the Time column) for each row.

Once the tab is filled out, you may proceed to another tab or click **Apply** to use the stimulus. To remove a previously applied stimulus, click **Remove**.

To scroll through signals:

- If you add more signals to the dialog than will fit in the window, a scroll bar will appear.
- You may scroll through all the signals to view their values.

> **Note:** The Time column remains fixed when you scroll horizontally through the signal columns.

Time Units

Time units chosen apply to all Time values in the Time column. Selectable units are:

- cyc – Instruction cycles
- h:m:s – Hours:minutes:seconds
- ms – milliseconds
- us – microseconds
- ns – nanoseconds

When h:m:s is chosen, values input in the Time column will have the following meanings:

- 1 – means 1 second
- 1:00 – means 1 minute
- 60 – becomes 1:00
- 999 – becomes 16:39
- 9999999 – becomes 2777:46:39
- 1:5 – becomes 1:05
- 60:5 – becomes 1:00:05

The time value range is 0 to (2^31 – 1) seconds. Anything outside this range will be changed to the default (0 for the first tab and re-arm delay; 1 for the condition wait).

Care should be taken when changing the time units for Time values already entered. As an example, if a time of 100 cyc is initially specified but then the Time Units are changed to h:m:s, 100 will be interpreted as an integer and converted into 1:40.

19.4.3 Advanced Pin/Register Tab

Advanced (complex) synchronous pin/register actions may be entered here. For basic actions, use the Pin/Register Actions Tab.

For advanced synchronous pin/register actions, first define conditions, then define triggers.

19.4.3.1 DEFINE CONDITIONS

> **Note: Precautions for Using SFR Values As Conditions**
>
> Conditions will ONLY occur when the SFR is updated by the user code, not the peripheral.
>
> For instance, the condition within Advanced Pin Stimulus dialog is set up to trigger when `TMR2 = 0x06`. When TMR2 is incremented past 0x06, the condition will not be met. However, if the following sequence is executed in user code, then the condition will occur:
>
> ```
> MOVLW 0X06
> MOVWF TMR2
> ```

Define the conditions for one or more stimuli in each row of this section as shown in Table 19-1.

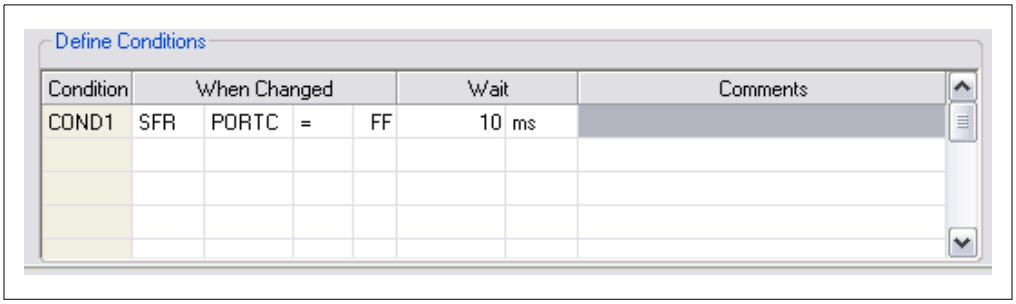
TABLE 19-1: DEFINITIONS OF STIMULI CONDITIONS

Item	Definition
Condition	A name for the condition you are specifying is automatically generated when you enter data in any other column. This label will be used to identify the condition in the Condition column of the Define Triggers section of this tab.
When Changed	Define the change condition. I.e., the condition is true when the value of the pin/register in Column 2 (its type specified in Column 1) changes to the relationship of Column 3 to the value of Column 4. **Note:** Conditions are only checked on changes, not every cycle. **Column 1:** Select the type of pin/register, either "SFR", "Bitfield", "Pin" or "All" of the above. This will filter the content of Column 2. **Column 2:** Select the pin/register to which the condition will apply. **Column 3:** Select the condition, either equal (=), not equal (!=), less than or equal (<=), greater than or equal (>=), less than (<) or greater than (>). **Column 4:** Enter the value for the condition. **Note:** Care must be taken when using SFR values as conditions.
Wait	Once the condition defined above is true, specify how long to wait until the stimulus is applied. **Column 1:** Wait time value. **Column 2:** Wait time value units.
Comments	Add descriptive information about the condition.

As an example (see Example 19-1), set up a condition, COND1, such that when the value of register PORTC equals FF, stimulus defined in "Define Triggers" is applied 10 ms later.

> **Note:** If PORTC has an initial value of FF, or never changes to FF, the condition will never be met and no stimulus will be applied.

EXAMPLE 19-1: REGISTER EQUALS A VALUE



19.4.3.2 DEFINE TRIGGERS

Define stimulus triggers in each row of this section as shown in Table 19-2, with reference to the conditions set up in “Define Conditions”.

TABLE 19-2: DEFINITIONS OF STIMULUS TRIGGERS

Item	Definition
Enable	Alternately enable or disable the trigger setup in this row.
Condition	Refers to the label of the condition set up in the “Define Conditions” section of this tab. Select a condition from the list.
Type	Select whether the trigger condition will apply once (1x) or continuously/ repeatedly (Cont).
Re-Arm Delay	If Type = Cont, then enter a delay until the trigger condition is checked again. The delay value is entered in the first column and the delay value unit is selected in the second column.
Click here to Add Signals	Click on the column title to open the Add/Remove Pin/Registers Dialog (see **Section 19.4.4 “Add/Remove Pin/Registers Dialog”**). In this dialog, you select the pins, registers or other signals to which you will apply stimulus. These selections will become the titles of the columns.

To scroll through signals:

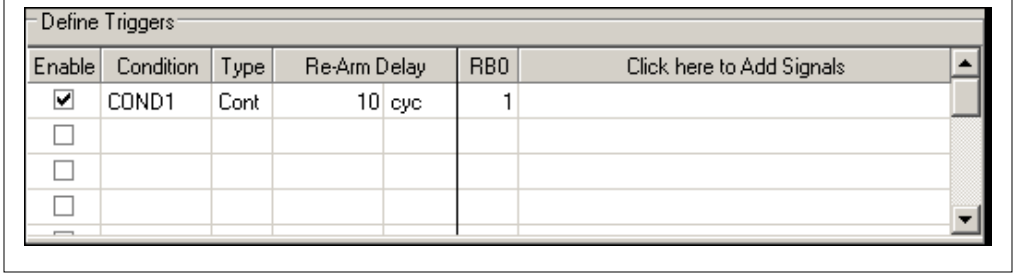
- If you add more signals than will fit in the window, a horizontal scroll bar will appear.
- You may scroll through all the signals to view their values.

Note: The columns to the left of the signal column(s) remains fixed when you scroll through the signal columns.

For the condition set up in Example 19-1, COND1, set up the following stimulus trigger:

1. Make the pin RB0 high when the COND1 is met (see Example 19-2).
2. Wait 10 instruction cycles and check for the condition again. If and when it occurs, make pin RB0 high again.
3. Repeat step 2 until the program is halted.

EXAMPLE 19-2: MAKE PIN HIGH ON REGISTER VALUE



19.4.3.3 "DEFINE CONDITIONS" WAIT VS. "DEFINE TRIGGER" RE-ARM DELAY

In "Define Conditions", the Wait time refers to the time from when the condition is true until the stimulus is applied. In "Define Trigger", the re-arm delay is time from when the stimulus was applied until the condition for triggering is checked again.

19.4.4 Add/Remove Pin/Registers Dialog

This dialog (Figure 19-4) opens when you click on "Click here to Add Signals" column heading on either the **Pin/Register Actions** tab or the **Advanced Pin/Register** tab. Use this dialog to add or remove pins, registers or other signals to which you will apply stimulus. These selections will become the titles of the columns in the associated Pin/Register tab.

FIGURE 19-4: ADD/REMOVE PIN/REGISTERS DIALOG

To Add a Signal:

- From the "Available Signals" drop-down box, select a type of signal, either "SFR and Bitfield", "Pin Only" or "All Signals".
- In the list below, either:
  - Click on a signal and then click **Add** to add the signal to the "Selected Signals" list.

    or
  - Double click on a signal to add it to the "Selected Signals" list.

> **Note:** Multiplexed pins that support peripheral I/O (e.g., CCP2) that are configured through the configuration bits to use (e.g., RB3 or RC1) will only be listed by the digital I/O name RB3 or RC1.

- Click **OK**.

This signal will now appear as a column head on the associated **Pin/Register Actions** or **Advanced Pin/Register** tab.

To Delete a Signal:

- In the “Selected Signals” list, either:
  - Click on the signal and then click **Remove** to remove the signal from this list.

    or
  - Double click on the signal to remove the signal from this list.
- Click **OK**.

This signal will no longer appear as a column head on the associated pin/register tab.

To Change the Order of Signals on the Pin/Register Tab:

- Click on the signal from the “Selected Signals” list.
- Click either **Move Up** or **Move Down** to change order of this signal in the list.
- Click **OK**.

Its location will now be reflected in the order of the column heads on the associated pin/register tab.

19.4.5 Clock Stimulus Tab

Pulse (high and low) values applied to a pin is clocked stimulus as shown in Table 19-3. Clocked stimulus may be entered here.

TABLE 19-3: DEFINITIONS OF CLOCK STIMULUS

Item	Definition	
Label	Unique name of the clock stimulus you are specifying (optional).	
Pin	Choose the pin on which you will apply clocked stimulus.	
Initial	Enter the initial state of the clocked stimulus, either low or high.	
Low Cycles	Enter a value for the number of low cycles in a clock pulse.	
High Cycles	Enter a value for the number of high cycles in a clock pulse.	
Begin	Click here to activate the selection in the Begin section:	
	Always (default)	Begin stimulus immediately on program run.
	PC=	Begin stimulus when the program counter equals the entered value.
	Cycle=	Begin stimulus when the instruction cycle count equals the entered value. *absolute time* (cyc) – relative to the beginning of simulation. *after last clock* (cyc+) – relative to the End Cycle (see below). After the last clock relative to the End Cycle (see below)
	Pin=	Begin stimulus when the selected pin has the selected value (low or high).
End	Click here to activate the selection in the End section:	
	Never (default)	Apply stimulus until program halt.
	PC=	End stimulus when the program counter equals the entered value.
	Cycle=	End stimulus when the instruction cycle count equals the entered value. *absolute time* (cyc) – relative to the beginning of simulation. *from clock start* (cyc+) – relative to the Begin Cycle (see above).
	Pin=	End stimulus when the selected pin has the selected value (low or high).
Comments	Add descriptive information about the stimulus.	

19.4.6 Register Injection Tab

Registers may be injected with values set up in a file (see Table 19-4). Enter information for register injection here.

For General Purpose Register injection, more than one byte may be injected (for arrays). Therefore, care must be taken to not overwrite data.

TABLE 19-4: DEFINITIONS OF REGISTER INJECTIONS

Item	Definition
Label	Name the register injection you are specifying (optional).
Reg/Var	Select a destination register for data injection from the list*. Listed registers include SFRs (top of list) and any GPRs used (bottom of list). **Note:** These GPR variables are shown only after the program is compiled.
Trigger	Select when to trigger injection. For most registers, this is either on **Demand** or when the **PC** equals a specified value (see next column). If the peripheral is implemented, it can only be triggered on demand. If it is a GPR, it must be triggered on PC. E.g., if SPIBUF is used when SPIBUF is accessed (Read), injection into SPIBUF will occur. For USART/UART registers, injection is **Message**-based. For more on simulating the USART/UART, see **Section 18.7 "Using a USART/UART"**.
PC Value	If Trigger=PC, enter a PC value for triggering injection. This can be an absolute address or a label in the code. **Note:** For MPLAB® C18/C30 code, labels are not supported and therefore local variables may not be in scope when expected. Use absolute addresses or function names. (Void functions can be defined if needed. E.g. void PCTrigger(void){ };
Width	If Trigger=PC, the number of bytes to be injected.
Data Filename	Browse for the injection (data) file. See below for more information.
Wrap	**Yes** – Once all the data from the file has been injected, start again from the beginning of the file. **No** – Once all the data from the file has been injected, the last value will continue to be used for injection.
Format	Select the format of the injection file. Regular Data File: **Hex** – ASCII hexadecimal **Raw** – Raw image, interpreted as binary **SCL** – SCL format. See spec for definition. **Dec** – ASCII decimal Message-Based Data File: **Pkt** – Hex or Raw packet format (used with UART receive register) **Note:** When importing STI stimulus files the maximum line length of any line within the old files must not exceed 260 characters.
Comments	Add descriptive information about the register injection.
* Not all registers may be displayed. E.g., selecting the ADRESL register on the PIC18F458 for injection will actually inject stimulus into both the ADRESH and ADRESL registers.	

19.4.6.1 REGULAR DATA FILE EXAMPLE

Below is an example of an injection file.

EXAMPLE 19-3: REGISTER STIMULUS FILE IN HEX

```
110
02E
A38
541
1A0
0FD
```

19.4.6.2 ADC BUFFER INJECTION EXAMPLE - dsPIC/PIC24 DEVICES

A data file with the ADC result values is listed in Example 19-4.

EXAMPLE 19-4: ADC REGISTER STIMULUS FILE IN HEX

```
0x0045
0x0023
0x0015
0x0000
0x0012
0x0024
0x0049
```

For ADC configured to convert 3 samples, when it completes its first conversion, it puts the result 0x0045 into ADCBUF0. When it completes its second conversion, it puts the result 0x0023 into ADCBUF1. When it completes its third conversion, it puts the result 0x0015 into ADCBUF2.

For the ADC, the BUF0 or ADRESL SFR is used to hook the file into the peripheral. The peripheral simulation will read data from the file when a conversion is complete and place it into the relevant SFR for the result. (This also applies to 10-bit A/Ds where ADRESH and ADRESL need to be populated or adjusted at the conversion completion.)

19.4.6.3 MESSAGE-BASED DATA FILE DESCRIPTION

Data in the file will be interpreted in packets as follows:

- HEX-style – spaced, separated hexadecimal numbers will be treated as one packet, e.g., 05 07 6A 6B 105 107. Since the UART is 9-bits wide, receive values larger than 0x1FF will be masked off.
- RAW-style – text between a pair of quotation characters (") will be treated as one packet, e.g., "this is a packet"

Two commands will be supported to specify the wait time between two packets:

- wait *n unit* – where *n* is a non-negative integer (0 to 2^{64}-1) and *unit* is any time unit (ps, ns, us, ms, sec, min, hr). This command specifies the time period to wait before the next packet will be injected. If *unit* is invalid, then wait defaults to 0 seconds.
- rand *lower upper unit* – where *lower* and *upper* are non-negative integers (0 to 2^{64}-1, lower < upper) and *unit* is any time unit (ps, ns, us, ms, sec, min, hr). This command specifies a random time period, bounded by *lower* and *upper*, to wait before the next packet can be injected. If *unit* is invalid, then rand defaults to 0 seconds.

If there is a parsing error for either command (e.g., out-of-bound *n*, *lower*, or *upper*, insufficient number of parameters), the command will be treated as wait 0 sec.

A comment line in the data file will begin with two forward slash characters in the first and second character position, e.g. //. An empty line will be ignored.

Adjacent packet lines between wait, rand, and <EOF> will be concatenated as one single packet (without the new line character present at the end-of-line).

19.4.6.4 MESSAGE-BASED DATA FILE EXAMPLES

Below are examples of message-based injection files.

EXAMPLE 19-5: REGISTER STIMULUS FILE 1

```
//wait for 0 second, then a packet of "abc cba",
//followed by a 9-bit number 0x163
wait 0 sec
61 62 63 20
63 62 61 163
//pause for 20 seconds after the previous packet,
//then a packet "efgh"
wait 20 sec
"efgh"
//immediately have another packet "2D4"
wait 0 ms
32 45 34
//pause for 10 ns
wait 10 ns
//pause for 10 more ns, then packet "43$24!"
wait 10 ns
34 33 24 32 34 21
//pause for a random time from 15 to 20 sec, then a packet
rand 15 20 sec
89 90 91 92 93 94
//pause for a random time from 0 to 100 min, then a packet
rand 0 100 min
11 22 33
```

EXAMPLE 19-6: REGISTER STIMULUS FILE 2

```
wait 0 sec
61 62 63 20 6A 62 61
wait 200 ms
// the following is equivalent to "the quick brown fox"
"the quick brown "
"fox"
// CR-LF in a packet
wait 20 sec
65 66 67 68 13 10
69 70
// 9-bit value in a packet
wait 20 sec
165 166 167 168
wait 30 ms
// the following is "012mix"
32 33 34
"mix"
wait 100 ns
// the following blank line is ignored

64 34 33 24 32 34 21
rand 15 20 sec
89 90 91 92 93 94
rand 0 100 min
11 22 33
```

19.4.7 Register Trace Tab

The value of a specified register may be saved to a file (traced) during a run for certain conditions (see Table 19-5). Specify register trace information here.

Note: If you have a file attached in order to output values of a register, before you can view this file in some editors, you must detach the file. Detaching the file allows it to close and save the data. If left attached, an input stream continues going to this file and the data may not be available for display.

TABLE 19-5: DEFINITIONS OF REGISTER TRACE

Item	Definition
Label	Name the register trace you are specifying (optional).
Reg/Var	Select the source register for tracing from the list. Listed registers include SFRs (top of list) and any GPRs used (bottom of list).
Trigger	Select when to trigger the trace, either on Demand or when the PC equals a specified value (see next column). If the peripheral is implemented, it can only be triggered on demand. If using a GPR, it must be triggered on PC. E.g., if PORTA is used whenever writing to PORTA, tracing will occur.
PC Value	If Trigger=PC, enter a PC value for triggering tracing. This can be an absolute address or a label in the code. **Note:** For MPLAB® C18/C30 code, labels are not supported and therefore local variables may not be in scope when expected. Use absolute addresses or function names.
Width	If Trigger=PC, the number of bytes to be traced.
Trace Filename	Browse for a location for the trace file.
Format	Select the format of the trace file. **Hex** – ASCII hexadecimal **Raw** – Raw image, interpreted as binary **SCL** – SCL format. See spec for definition. **Dec** – ASCII decimal
Comments	Add descriptive information about the register trace.

GPR injection and Trace can be used to either populate an array with specific data at the start of your program (using PC=main();) or extract data from an array after a specific algorithm has been executed.

19.5 ADVANCED OPERATIONS

When setting up SCL file input as well as asynchronous stimulus, be aware of input interaction (see **Section 19.6 “Stimulus Input Interaction”**).

If you have developed a synchronous stimulus file using SCL, click **Advanced** to open the Advanced SCL Operations dialog.

Item	Definition
Override Work-book with SCL File	Displays SCL filename and path selected.
Attach/Detach	Toggles between Attach and Detach depending on whether or not the Override Workbook with SCL File field is populated. Use to browse for file to attach or to detach file shown in Override Workbook with SCL File field.
Import/Merge	Use to import an SCL file or merge multiple SCL files.
Generate SCL File	Generates an SCL file based on the settings on any of the first five tabs (synchronous stimulus).
OK	Acknowledges action to take based on choices in this dialog.
Help	Opens on-line help for the Simulator.

Search for the file and location, and click on **Attach** to attach it to the controller.

To remove an SCL file from the controller, click **Detach**.

If you wish to merge two SCL files into one, click **Import/Merge**.

19.6 STIMULUS INPUT INTERACTION

If a pin (e.g., RB1) assignment and a port (e.g., PORTB.RB) assignment happen on the same cycle, the port supersedes the pin assignment. But, if they happen on different cycles, you can mix and match pin and port assignment.

In addition, you can use PORT injection with pin/port assignment.

NOTES:

Chapter 20. Using Stimulus – PIC17 Devices

20.1 INTRODUCTION

Stimulus functions allow you to direct the simulator to apply artificial stimuli while debugging your code. You can set pins high or low and inject values directly into registers.

Select Stimulus from the Debugger menu, then choose the tab for the stimulus type.

Topics covered in this chapter:

- Using Pin Stimulus – Synchronous or Asynchronous stimulus on I/O pins
- Using File Stimulus – Triggered stimulus on I/O pins or file registers, set up from files

20.2 USING PIN STIMULUS

Pin Stimulus is user-defined. It may be:

- Asynchronous – Fired by the user.
- Synchronous – Regular, repeating series of high/low voltages with adjustable duty cycle set by number of clock cycles per high/low time.

Pins can be set high or low at program counter addresses. Addresses in the list will be looked for sequentially.

The contents of the pin stimulus list can be edited, saved and restored from a file.

Note:	Changing stimulus files outside the Stimulus dialog is NOT recommended.

- Creating/Editing Pin Stimulus
- Applying Pin Stimulus
- Pin Stimulus Display

20.2.1 Creating/Editing Pin Stimulus

1. Select *Debugger>Stimulus Controller* and then click the **Pin Stimulus** tab.
2. To create a new file, click **Add Row** to create a new row for entering data. To open an existing pin stimulus file for editing, click **Load** and select the file using the Open dialog.
3. Click in the "Type" column of the table row for which you wish to add or change data. Select or change the type of stimulus (Async or Synch). Depending on the stimulus type chosen, each cell gets an appropriate control which allows you to specify the value of that cell.

Asynchronous Setup

a) Click on "Pin" to select or change the pin on which stimulus will be applied. Available pin names are listed. The actual names depend on the device selected in MPLAB IDE.
b) Click on "Action" to select or change a stimulus action. You may choose Pulse, High, Low or Toggle.
c) Click on "Comments" to specify or change a comment which will be saved and restored in the pin stimulus file.

Synchronous Setup

a) Click on "Pin" to select or change the pin on which stimulus will be applied. Available pin names are listed. The actual names depend on the device selected in MPLAB IDE.
b) Click on "High Cycles" to set or change the number of Clock cycles till the high state is applied. Then click on "Low Cycles" to set or change the number of Clock cycles till the low state is applied.
c) Click on "Invert" to set or change whether high and low cycles are inverted.
d) Click on "Comments" to specify or change a comment which will be saved and restored in the pin stimulus file.

4. When you have completed adding or editing data in a row, click **Edit Complete**.
5. Continue adding or editing other rows of data till your pin stimulus is complete. If you wish to remove a row, click on a cell in the row to select the row and then click **Delete Row**.
6. When you are done, click **Save** and save the pin stimulus file (.psti).

20.2.2 Applying Pin Stimulus

Once you have created/edited a pin stimulus file, you may apply the stimulus as follows:

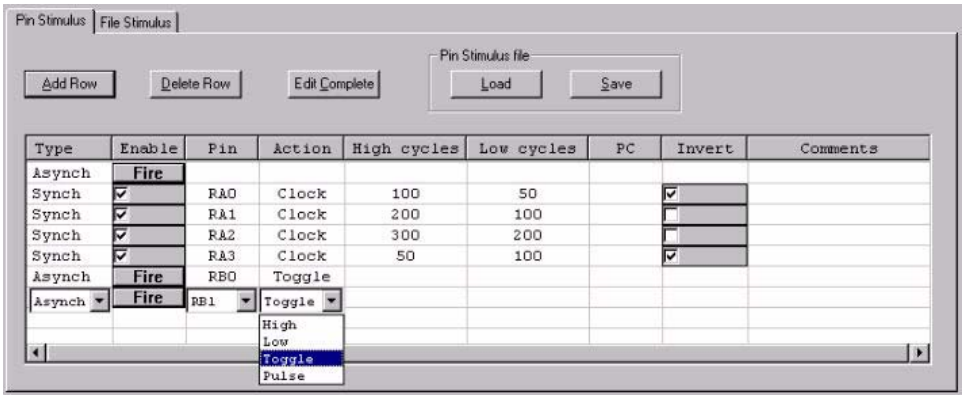
- Asynchronous stimulus (Type = Asynch): Press the Fire button. The designated event occurs on the designated pin.
- Synchronous stimulus (Type = Synch): Check the enable field. A message is sent to MPLAB SIM to update the stimuli. Thus whatever is displayed is current.

Note:	The Stimulus dialog must be open for stimulus to be active.

20.2.3 Pin Stimulus Display

Select *Debugger>Stimulus* and then click the Pin Stimulus tab to set up pin stimulus. Closing the Stimulus dialog disables stimulus.

Note 1:	If any files are changed externally while in use by this dialog tab, the changes will NOT be reflected in the stimuli passed to the simulator. Changing stimulus files outside the Stimulus dialog is NOT recommended.
2:	Unsaved changes will be lost without notice when you close this dialog.



Buttons – Row Edit

- Add Row – allows you to add a new pin stimulus. A default pin row at the bottom of the pin list will be created and the cursor will be moved there for editing.
- Delete Row – removes the row at the cursor from the list.
- Edit Complete – turns off the display of any list boxes, so all the data appear clearly.

Buttons – Pin Stimulus File

- Load – allows you to specify a previously saved pin stimulus file and add its contents to the stimulus list.
- Save – allows you to save the current contents of the stimulus list to a pin stimulus file of your choice.

Table Entries

- Type – has a choice list containing “Asynch” and “Synch”. The Type you select determines what is available in some of the remaining cells in that row.
- Enable – has a checkbox for Synch stimulus, a Fire button for Asynch Stimulus. This is not a data cell.
- Pin – contains the names of the available pins. The actual names depend on the device selected in the IDE.

- Action
  Type = Asynch:
  Pulse: Change the state of the pin to its opposite and return.
  High: Change the state of the pin to high.
  Low: Change the state of the pin to low.
  Toggle: Change to state of the pin to its opposite.
  Type = Synch: Clock
- High cycles – number of cycles high state will be applied for Synch stimulus.
- Low cycles – number of cycles low state will be applied for Synch stimulus.
- Invert – inverts high and low cycles when checked for Synch stimulus.
- Comments – allows you to specify a comment which will be saved and restored if you maintain Pin Stimulus in a file.

20.3 USING FILE STIMULUS

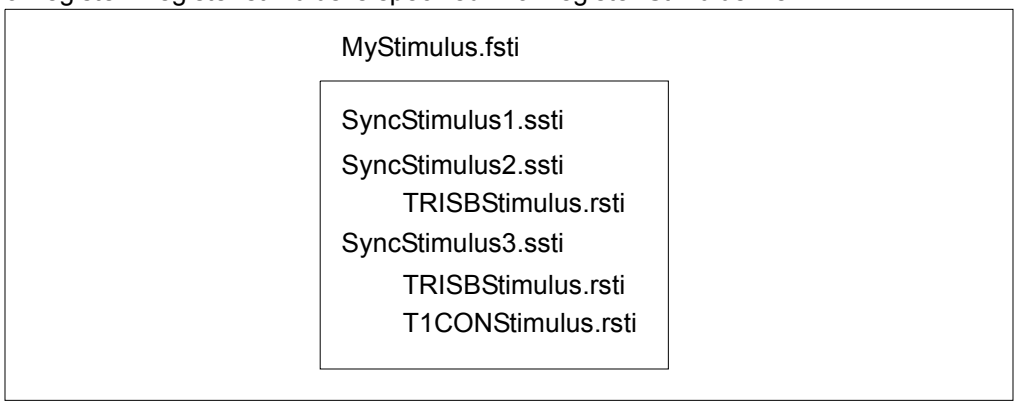
MPLAB SIM triggers changes to pin and register values specified in a file by using File Stimulus. These files specify values that will be sent to a pin or register when a trigger is fired. This trigger may be a cycle count for pins or a program memory address for registers.

Note 1:	Changing stimulus files outside the Stimulus dialog is NOT recommended.
2:	If you have already set pin stimulus on a port pin, you will not be able to inject stimulus into that pin or corresponding register using file stimulus.

- Creating/Editing File Stimulus
- Applying File Stimulus
- File Stimulus Display

20.3.1 Creating/Editing File Stimulus

Select *Debugger>Stimulus Controller* and then click the **File Stimulus** tab. A File Stimulus file is composed of one or more Synchronous Stimulus files. A Synchronous Stimulus file contains information on triggers used for applying stimulus to either a pin or register. Register stimulus is specified in a Register Stimulus file.



- Creating/Editing a File Stimulus File (.fsti)
- Creating/Editing a Synchronous Stimulus File (.ssti)
- Creating/Editing a Register Stimulus File (.rsti)

20.3.1.1 CREATING/EDITING A FILE STIMULUS FILE (.FSTI)

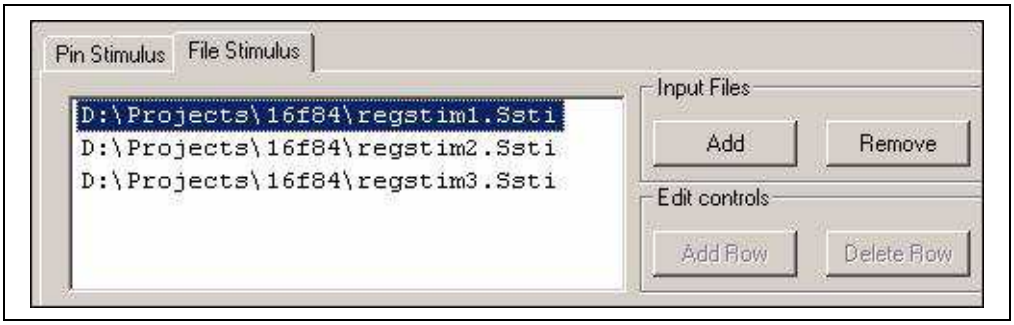
A file stimulus file consists of one or more synchronous stimulus files (*.ssti) that will be applied to the simulator.

To create a file stimulus file:

1. Create or open one or more synchronous stimulus files in the file list (below the tab).
2. Click **Save Setup** under File Stimulus.

To edit an existing file stimulus file:

1. Click **Load Setup** under File Stimulus to open an existing file stimulus file.
2. Use the file list and “Input Files” buttons **Add** and **Remove** to edit the list.



20.3.1.2 CREATING/EDITING A SYNCHRONOUS STIMULUS FILE (.SSTI)

1. If there are no stimulus files listed in the file list (below the tab), click **Add** under “Input Files” to open the Open file dialog. Browse to a location to create a new file or to edit an existing file. To create a new stimulus file, enter a name and click **Open**. To open an existing stimulus file for editing, click on it to select it and click **Open**. The stimulus file location and name should now appear in the file list.
2. Click on the stimulus file in the list to select it for editing. Then click **Edit**. If you realize you have selected the wrong file, click **Cancel** and then select another file.
3. For a new file, click **Add Row** to create a new row for entering data.
4. Click in the “Trigger On” column of the table row for which you wish to add or change data. Select or change the type of trigger (Cycles and PC). Depending on the trigger type, each cell gets an appropriate control which allows you to specify the value of that cell.

   Cycles Setup

   a) Click on “Trig Value” to set the cycle count at which the trigger fires.
   b) From “Pin/Register”, choose the pin to which the value in the next column will be applied when the trigger fires. The actual pin names available depend on the device selected.

Note:	Do not choose a register. It will not work with cycle stimulus.

   c) Click on “Value” and enter or change a value to be applied when the trigger occurs. For a Pin, either 0 or 1. These values will be applied in turn, one at each trigger event.
   d) For setting up additional pins, click **Add Column** to add “Pin/Register” and “Value” columns.
   e) Click on “Comments” to enter or change a text string which will be saved and restored in the file.

PC Setup

a) Click on "Trig Value" to set the PC address at which the trigger fires.

b) From "Pin/Register", choose the register to which the value in the next column will be applied when the trigger fires. The actual register names available depend on the device selected.

Note: Do not choose a pin. It will not work with PC stimulus.

Note: You cannot inject file values into port registers, e.g., PORTB. Use pin values. Also, you cannot inject file values into the W register.

c) Click on "Value" and enter or change a value to be applied when the trigger occurs. For a Register, the name of a file (`*.rsti`) which contains a sequence of values. These values will be applied in turn, one at each trigger event.

d) Click on "Comments" to enter or change a text string which will be saved and restored in the file.

e) For register stimulus, you may only have one stimulus file attached to one register and vice versa. If you add additional rows/columns, only the last occurrence will be executed.

f) If you wish to remove a row, click on a cell in the row to select the row and then click **Delete Row**.

5. When you are done, click **Save** to save the synchronous stimulus file.

20.3.1.3 CREATING/EDITING A REGISTER STIMULUS FILE (.RSTI)

A register stimulus file contains register stimulus information in hexadecimal, return-delimited format. Any text editor or spreadsheet application may be used to create or edit this file. The default extension is .rsti, but any extension may be used. Assign a register stimulus file to a register in the Value column of a Synchronous Stimulus file (*.ssti).

Note: You may only have one register stimulus file assigned to one register at a time.

EXAMPLE 20-1: REGISTER STIMULUS FILE

```
10
2E
38
41
A0
FD
```

20.3.2 Applying File Stimulus

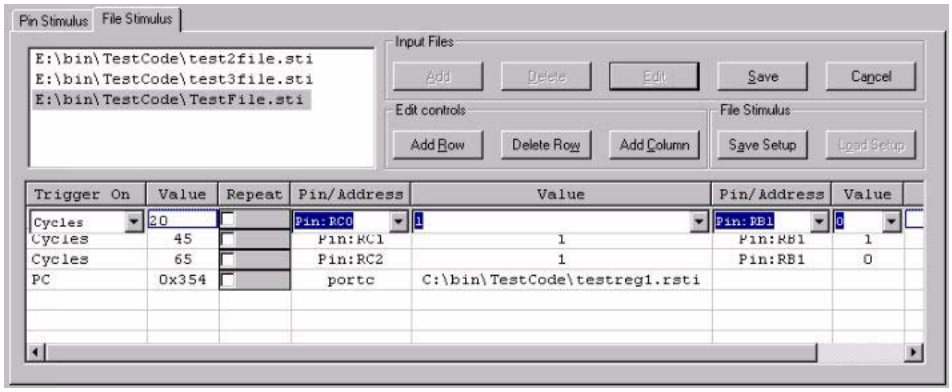
Once you have created/edited a file stimulus file, the stimulus is applied as the program is run under MPLAB IDE. When the designated trigger is activated, the designated event occurs.

Note: The Stimulus dialog must be open for stimulus to be active.

20.3.3 File Stimulus Display

Select *Debugger>Stimulus Controller* and then click the File Stimulus tab to set up file stimulus. Closing the Stimulus dialog disables stimulus.

Note 1:	This is only available when the MPLAB SIM tool has been selected from the Debugger menu.
2:	If any files are changed externally while in use by this dialog tab, the changes will NOT be reflected in the stimuli passed to the simulator. Changing stimulus files outside the Stimulus dialog is NOT recommended.
3:	Unsaved changes will be lost without notice when you close this dialog.



File List

List of input files available for use as file stimulus. Add, delete, edit and save files from this list by using the Input Files buttons.

Buttons – Input Files

- Add – brings up a browse dialog to select a new file to add to the file list. You may select an existing file or enter the name of a file to be created.
- Delete – deletes the highlighted file from the list (not from your drive)
- Edit – displays the contents of the selected file in the stimulus list and allows you to edit the contents. It also disables the Add and Delete buttons.
- Save – saves the contents of the stimulus list into a file, clears the stimulus list, disables editing and reenables the Add and Delete buttons.
- Cancel – Cancel selection of file.

Note:	You may display and edit only one file from the file list at a time.

Buttons – Edit Controls

- Add Row – creates a default row at the bottom of the stimulus list and moves the cursor there for editing.
- Delete Row – removes the row at the cursor from the list.
- Add Column – adds another Pin/Register-Value pair of columns to the stimulus list to the left of the Comments column, allowing you specify additional stimuli for each trigger.

Buttons – File Stimulus

These buttons enable you to save and restore combined file setups.

- Save Setup saves the names of the files currently in the file list into a file.
- Load Setup loads the filenames in a file into the file list, just as though you had added them individually.

Note:	There is currently no way to view the combined contents of a File Stimulus setup.

Table Entries

- Trigger On – choose between Cycles (Instruction Cycles) and PC (Program Counter). The selection determines what is available in some of the remaining cells in that row.
- Trig Value – Cycle count or the PC address at which the trigger fires. Value entered can be decimal or hexadecimal. Hex numbers must be preceded with 0x.
- Pin/Register – where to apply the value; a choice among the names of the available pins and registers. The actual names depend on the device selected in the IDE.
- Value – value to be applied when the trigger occurs. For a Pin, either '0' or '1'. For a Register, a single value or the name of a file which contains a sequence of values. These values will be applied in turn, one at each trigger event.
- Comments – text string which will be saved and restored in the file.

Note:	A Register stimulus will be effective only with a PC trigger. You may specify a Cycle trigger for a Register, but it will have no effect. In addition, Register values must be specified from a file; specifying a numeric value will have no effect.

Chapter 21. Simulator Troubleshooting

21.1 INTRODUCTION

This section is designed to help you troubleshoot any problems, errors or issues you encounter while using MPLAB SIM. If none of this information helps you, please see Support for ways to contact Microchip Technology.

- Common Problems/FAQ
- Limitations

21.2 COMMON PROBLEMS/FAQ

- The simulator does not use the stimulus file I created
- I cannot attach more than one register stimulus file to a register
- My UART input file is not working
- I cannot use symbols to set breakpoints
- The simulator is not recognizing my breakpoints, although they seem to be set and enabled in my code windows
- I cannot reset my program using MCLR
- My program keeps resetting
- The simulator is not halting on WDT time-out
- For PIC18 devices, the SFR window is not updating for Timers 1 and 3
- No data in the logic analyzer
- Animation doesn't work
- View simulator trace or simulator logic analyzer is grayed out

The simulator does not use the stimulus file I created

If your file is incorrect, the simulator will not notify you; it will simply ignore its contents. You should only set up your stimulus files using either:

- the Stimulus dialog

- the Pin Stimulus and File Stimulus dialogs (PIC17 Devices)

Do not edit files externally.

In addition, there may be pin/port interaction concerns. See the documentation for the type of stimulus you are using.

I cannot attach more than one register stimulus file to a register

MPLAB SIM only allows one register stimulus file (.rsti) per register in stimulus for PIC17 MCU devices.

My UART input file is not working

Ensure that the data file begins with a `wait` statement.

The UART data file is attached and treated like real hardware injection. If there is no initial `wait` statement, the data is consumed at the default baud rate, often before the UART can even be initialized.

I cannot use symbols to set breakpoints

Ensure a build is successful before attempting to set a breakpoint. Symbols are available only after a build.

The simulator is not recognizing my breakpoints, although they seem to be set and enabled in my code windows

Be sure "Global Break Enable" is set on the **Break Options** tab, Settings dialog (PIC17's only).

I cannot reset my program using $\overline{\text{MCLR}}$

For dsPIC DSC or PIC18F devices, you cannot reset the device using pins. You must use the Reset commands on the Debugger menu.

My program keeps resetting

Check the Configuration bits settings (*Configure>Configuration Bits*) for your selected device. Some Reset functions (such as Watchdog Timer Reset) are enabled by default.

The simulator is not halting on WDT time-out

All simulators have a choice for WDT Timeout (Break, Break+Warn or Reset). Also, you must enable WDT time-out in the Configuration Bits dialog. For PIC17's only, be sure "Global Break Enable" is set in the **Break Options** tab, Settings dialog.

For PIC18 devices, the SFR window is not updating for Timers 1 and 3

For PIC18 devices, Timer 1 and 3 have a buffer for the high byte. If RD16 is set, TMR1H and TMR3H will remain at their present values until a read is done. After the read, the TMRxH buffer value will be transferred to the TMRxH register and the value will be visible in the SFR window. Otherwise, the buffer value is not displayed in the SFR window. For convenience, under the Special Function Registers window internal SFR representation of the full 16-bit timers and the prescalers are available.

See your device data sheet for more on Timer 1 and 3 operation.

No data in the logic analyzer

The Logic Analyzer display needs to be populated with signals/channels to observe. Click **Channels** to open the Configure Channel dialog. For more on this window, see MPLAB IDE documentation on the Logic Analyzer window.

Animation doesn't work

The Program Memory window should be kept in focus and a debug file is required for animation to work. Therefore, you should assemble/compile your code with debug information (generated by the linker) if you want to use Animate.

View simulator trace or simulator logic analyzer is grayed out

Check if the trace is enabled within the *Debugger>Settings* dialog or if a trace/logic analyzer window is already open.

21.3 LIMITATIONS

General and device-specific limitations for the simulator may be found in on-line help for MPLAB SIM.

Chapter 22. Simulator Reference

22.1 INTRODUCTION

Once MPLAB SIM has been selected as the debug tool in MPLAB IDE (*Debugger> Select Tool>MPLAB SIM*), the following simulator-specific features are available:

- Debugging Functions
- Debugging Dialogs and Windows
- Settings Dialog
- Settings Dialog – PIC17 Devices

22.2 DEBUGGING FUNCTIONS

Simulator-specific debug items/functions are added to the following MPLAB IDE features:

- Debugger Menu
- View Menu
- Right Mouse Button Menus
- Toolbar and Status Bar

22.2.1 Debugger Menu

In addition to the standard MPLAB IDE Debug menu items, the following items/operations are unique to the simulator:

Reset

- MCLR Reset – during normal operation can easily be simulated by selecting *Debugger>Reset>MCLR Reset*.
- Watchdog Timer Reset – simulates a watchdog timer timeout and sets bits in the control register.
- Brown Out Reset – simulates a brown out condition and sets bits in the control register.
- Processor Reset F6 – will reset the processor, the stopwatch and the simulation time. All other Resets will only reset the processor.

Stopwatch

Time the execution of your code with a stopwatch. For more information, see **Section 18.5 "Using the Stopwatch"**.

Complex Breakpoints

Open the **Section 22.3.1 "Simulator Complex Breakpoints Dialog"** (SimBreakpoints) dialog. Set multiple breakpoints in this dialog.

Stimulus Controller – for PIC17 devices only

For PIC17 devices: Setup I/O Pin and File Register stimuli for your code and apply them as your program executes. For more information, see **Chapter 20. "Using Stimulus – PIC17 Devices"**.

Stimulus

Develop code in SCL to control stimulus firing. For more information, see **Chapter 19. "Using Stimulus"**.

Profile

Show a profile of the current state of the simulator in the Output window or clear (Reset) the current state.

Clear Code Coverage

Manual reset of code coverage checkmarks if enabled on the **Code Coverage** tab of the Simulator Settings dialog. For more information, see **18.8 "Using Code Coverage"**.

Refresh PM

Refresh program memory to restart your program.

22.2.2 View Menu

In addition to the standard MPLAB IDE View menu items, the following items are unique to the simulator:

Simulator Trace

Display the window containing the current memory trace of your program's execution. For more information, see **Section 18.3 "Using Simulator Trace"**.

Simulator Logic Analyzer

Display the logic analyzer for selected signals. For more information, see **Section 18.4 "Using the Simulator Logic Analyzer"**.

22.2.3 Right Mouse Button Menus

Right mouse button menus will contain the standard MPLAB IDE Debug menu items. For Reset function, see the Debugger Menu.

22.2.4 Toolbar and Status Bar

The toolbar for the simulator will be the Standard Debug toolbar. For Reset function, see the Debugger Menu.

The status bar will identify MPLAB SIM as the debug tool.

22.3 DEBUGGING DIALOGS AND WINDOWS

Open the following debug dialogs and windows using the menu items mentioned in **Section 22.2 "Debugging Functions"**:

- Simulator Complex Breakpoints Dialog
- Set Breakpoint Dialog
- Sequenced Breakpoints Dialog
- AND Dialog

22.3.1 Simulator Complex Breakpoints Dialog

Set up simulator complex breakpoints in this dialog. Click on **Add Breakpoint** to open the Set Breakpoints dialog and add complex breakpoints to the window. Then use the other buttons for more advanced breakpoint options.

This dialog consists of an area for Complex Breakpoint Information and Complex Breakpoint Buttons.

22.3.1.1 COMPLEX BREAKPOINT INFORMATION

Information about each breakpoint is visible in this part of the dialog.

TABLE 22-1: SIMULATOR COMPLEX BREAKPOINT DIALOG

Control	Function
Breakpoint Type	Type of complex breakpoint – program or data
Address	Hex address of complex breakpoint location
File line # / Var Name	File name and line number or variable name of complex breakpoint location
Enabled	Check to enable a complex breakpoint

Once a breakpoint has been added to the window, you may right click on it to open a menu of options:

- Delete – delete selected breakpoint
- Edit/View – open the SimBreakpoints Dialog
- Delete All – delete all listed breakpoints
- Enable All – enable all listed breakpoints
- Disable All – disable all listed breakpoints

22.3.1.2 COMPLEX BREAKPOINT BUTTONS

Use the buttons to add a complex breakpoint and set up additional break conditions.

TABLE 22-2: SIMULATOR COMPLEX BREAKPOINT BUTTONS

Control	Function	Related Dialog
Add Breakpoint	Add a breakpoint	**Section 22.3.2 “Set Breakpoint Dialog”**
Sequenced Breakpoints	Set up a sequence until break	**Section 22.3.3 “Sequenced Breakpoints Dialog”**
ANDED Breakpoints	Set up ANDED condition until break	**Section 22.3.4 “AND Dialog”**

22.3.2 Set Breakpoint Dialog

Select a breakpoint for the Simulator Complex Breakpoints Dialog here.

- Program Memory Tab
- Data Memory Tab

22.3.2.1 PROGRAM MEMORY TAB

Set up a program memory breakpoint here.

TABLE 22-3: PROGRAM MEMORY BREAKPOINT

Control	Function
Address	Location of breakpoint in hex.
Breakpoint Type	The type of program memory breakpoint. See the device data sheet for more information on table reads/writes. *Program Memory Execution* – break on execution of above address *TBLRD Program Memory* – break on table read of above address *TBLWT Program Memory* – break on table write to above address
Pass Count	Break on pass count condition/ *Always break* – always break as specified in "Breakpoint type" *Break occurs Count instructions after Event* – wait Count (0-65535) instructions before breaking after event specified in "Breakpoint type" *Event must occur Count times* – break only after event specified in "Breakpoint type" occurs Count (1-65535) times

22.3.2.2 DATA MEMORY TAB

Set up a data memory breakpoint here.

TABLE 22-4: DATA MEMORY BREAKPOINT

Control	Function
Address	Location of breakpoint in hex.
Breakpoint Type	The type of data memory breakpoint. See the specific device data sheet for more information on breakpoint type reads/writes. The following is typical for 16-bit devices: *X Bus Read* – break on an X bus read of above address *X Bus Read Specific Byte* – break on an X bus read of above address for the specific byte value in "Specific Value" *X Bus Read Specific Word* – break on an X bus read of above address for the specific word value in "Specific Value" *X Bus Write* – break on an X bus write of above address *X Bus Write Specific Byte* – break on an X bus write of above address for the specific byte value in "Specific Value" *X Bus Write Specific Word* – break on an X bus write of above address for the specific word value in "Specific Value"
	For dsPIC devices only: *Y Bus Read* – break on an Y bus read of above address *Y Bus Read Specific Word* – break on an Y bus read of above address for the specific word value in "Specific Value" **Note:** Selecting the breakpoint type as *X Bus Write Specific Word* or *X Bus Write* for a Variable or SFR, then performing a write to the upper byte of an SFR using bit instructions does not halt the processor. As a work-around, put the upper byte address in the field "Address" instead of the SFR or Variable name.
Pass Count	Break on pass count condition/ *Always break* – always break as specified in "Breakpoint type" *Break occurs Count instructions after Event* – wait Count (0-65535) instructions before breaking after event specified in "Breakpoint type" *Event must occur Count times* – break only after event specified in "Breakpoint type" occurs Count (1-65535) times

22.3.3 Sequenced Breakpoints Dialog

Set up a sequential occurrence of breakpoints. Sequence execution of breakpoints is bottom-up; the last breakpoint in the sequence occurs first.

To add a breakpoint to a sequence:

- Select a breakpoint from the list of “Available Breakpoints”. Available breakpoints/triggers are those previously added to the breakpoint dialog.
- Select a sequence for the list of “Sequences”.
- Click **Add**.

To change the order of breakpoints in a sequence, drag-and-drop the breakpoint in the “Sequences list”.

To remove a breakpoint from a sequence:

- Select the breakpoint in the “Sequences” list.
- Click **Remove**.

22.3.4 AND Dialog

Set up an ANDED condition for breaking, i.e., breakpoint 1 AND breakpoint 2 must occur at the same time before a program halt. This can only be accomplished if a data breakpoint and a program memory breakpoint occur at the same time.

To add a breakpoint to the AND condition:

- Select a breakpoint from the list of “Available Breakpoints”. Available breakpoints/triggers are those previously added to the breakpoint dialog.
- Click **Add**.

To remove a breakpoint from a sequence:

- Select the breakpoint in the “ANDed Breakpoints” list.
- Click **Remove**.

22.4 SETTINGS DIALOG

Select *Debugger>Settings* to open the Settings dialog.

Note:	This is only available when the simulator has been selected from the Debugger Menu.

This dialog is composed of several tabs for setting up debugger features and functions.

- Osc/Trace Tab
- Break Options Tab
- SCL Options Tab
- UART1 IO Tab
- Code Coverage Tab
- Animation/Realtime Updates Tab
- Limitations Tab

22.4.1 Osc/Trace Tab

Select *Debugger>Settings* and then click the Osc / Trace tab.

Processor Frequency

In this dialog you set the clock speed at which you want your code to run. These values will also be used to set certain fields in the Stopwatch.

1. Select the Units of frequency
2. Enter the Frequency value

Note:	Configuration bit settings will have no effect on processor speed, e.g., if using x4 PLL, simulator will run at processor frequency, not 4 times processor frequency.

Trace Options

To trace all lines of an executing program for display in the trace window (and the related logic analyzer), check the "Trace All" checkbox. To stop program execution when the trace buffer is full, check "Break on Trace Buffer Full".

You may enter a value in the Buffer Size field and select one of these options: "K lines," "M lines," or "G lines" as a qualifier for the value. The size of the buffer may affect the simulation speed.

To see the trace window, select *View>Simulator Trace*. For more on tracing, see MPLAB IDE documentation on the Trace Memory window.

To see the logic analyzer window, select *View>Simulator Logic Analyzer*.

Note:	If you are watching the simulator logic analyzer and using a large trace buffer size (MB), the display will take several seconds to update as it passes (deciphers) the data from the trace buffer.

For more on this window, see MPLAB IDE documentation on the Logic Analyzer window.

22.4.2 Break Options Tab

Select *Debugger>Settings* and then click the **Break Options** tab. Set up break options for specific processor areas or conditions.

Messages are displayed as they are generated in the Output window under the MPLAB SIM tab. The messages will be displayed in a standard format as shown below:

```
Component-message number :  message
```

For example,

```
CORE-E0001: Stack Error occurred
ADC-W0001: Cannot change ADCON1 SSRC when ADON=1
```

Core (Including Stack)

Select break options from the drop-down lists for core warnings and errors:

- Break – Report and break on core errors
- Ignore – Ignore core errors (trap, normal behavior)
- Report – Report and trap on core errors

The core includes stack operation. Therefore, a break option for a stack overflow/underflow is set by selecting a break option for a core error. The default for core errors is Break, which means a break occurs on a stack error. For PIC12/16 devices, the simulator will never reset. For PIC18 devices, whether the simulator resets or not depends on the
Configuration bits (*Configure>Configuration Bits*).

The simulator reports the stack error before the possible Reset action. Thus, you can set the core error break option accordingly to capture that moment before the Reset.

Peripheral

Select break options from the drop-down lists for peripheral (e.g., timers, ADC, USART, etc.) warnings and errors:

- Break – Report and Break on Peripheral errors
- Ignore – Ignore Peripheral errors (Trap/Reset, normal behavior)
- Report – Report and Trap on Peripheral errors

WDT Timeout

Select break options from the drop-down list for watchdog timer warnings:

- Break – Break on WDT
- Break+Warn – Report and Break on WDT
- Reset – Reset on WDT (normal behavior)

WDT Period

Enter a watchdog timer time-out period value in milliseconds. The range is from 1 to 32767 ms. To return to the default WDT period, click **Default**.

22.4.3 SCL Options Tab

Select *Debugger>Settings* and then click the SCL Options tab. Set up SCL error screening options. For more on SCL, see **Section 19.4 “Stimulus Dialog”**.

Messages are displayed as they are generated in the Output window under the MPLAB SIM tab. The messages will be displayed in a standard format as shown below:

```
Component-message number :  message
```

For example,

```
SCL-E0003: Syntax error in vector file
```

SCL

Select break options from the drop-down lists for SCL warnings and errors:

- Report and Break on SCL errors
- Ignore SCL errors (Trap/Reset, normal behavior)
- Report and Trap on SCL errors.

22.4.4 UART1 IO Tab

Select *Debugger>Settings* and then click the UART1 IO tab.

Combined with the UART is the ability to use the standard C library IO capability. When the "Enable UART1 I/O" option is checked, the UART1 is used to read data from a file and write data to a file or the output window.

For more on the simulator and USART/UART operation, see **Section 18.7 "Using a USART/UART"**.

Note: The SIM UART IO settings are primarily for `printf` that supports ASCII readable text. When the UART transmits a register loaded with 8-bit data, the eighth bit is truncated. Any output when using the SIM UART1 IO settings will be in ASCII format regardless of the destination (window or file). To use 8- or 9-bit transmission use Register Trace and specify Hex or Dec format.

Input

The stimulus file used for input is selected by clicking the **Browse** button and using the Open file dialog.

Checking "Rewind Input" means that when a stimulus file is read to its end, it will wrap and restart at the beginning, i.e., you will have a continuous loop of input data. Unchecking this item means that when a stimulus file is read to its end, it will stop and no more stimulus input will be injected (until you reset).

Output

If a file is used for the output, click "File" and select the file name by clicking the **Browse** button and using the Save As file dialog.

If the Output window is to be used for output, click "Window".

22.4.5 Code Coverage Tab

Select *Debugger>Settings* and then click the **Code Coverage** tab.

Option	Description
Disabled	Code coverage disabled.
Enabled/Reset on POR	Code coverage enabled resets on power-on reset.
Enabled/Reset on Run	Code coverage enabled resets every time simulation is run.
Enabled/Manual Reset	Code coverage is reset when clear code coverage is clicked on in the Debugger menu. See **Section 22.2.1 "Debugger Menu"**.

For additional information on code coverage, see **Section 18.8 "Using Code Coverage"**.

Code Coverage Report

1. On the **Code Coverage** tab, select one of the Enabled options, then select the "Enable Output to File" option.
2. In the text box, specify a file in which to save the information. Type in a new file name and path or browse for an existing file.
3. Click **Apply** or **OK**.

22.4.6 Animation/Realtime Updates Tab

Select *Debugger>Settings* and then click the **Animation/Realtime Updates** tab.

Animation is the method of program execution preformed when selecting *Debugger>Animate*. Set the rate at which you want animation to run by entering sliding the slider under “Animate step time”. The range is 0 to 5 sec.

Realtime Updates apply to how often the Watch window is updated. By default, Watch window data is updated on program halt. To enable real-time updates, check the “Enable realtime watch updates” checkbox. Then slide the slider to select an update rate (0.1 to 5 sec.) Realtime updates can also be used with the Data Monitor and Control Interface (DMCI) under the Tools menu. Enable the function here to use with the DMCI.

22.4.7 Limitations Tab

Select *Debugger>Settings* and then click the **Limitation** tab.

This dialog presents known MPLAB SIM limitations for the device currently selected (*Configure>Select Device*).

If you want additional information about device limitations, click **Details**.

22.5 SETTINGS DIALOG – PIC17 DEVICES

Select *Debugger>Settings* to open the Settings dialog.

Note:	This is only available when the simulator has been selected from the Debugger Menu.

This dialog is composed of several tabs for setting up debugger features and functions.

- Clock Tab
- Break Options Tab
- Trace/Pins Tab
- Animation/Realtime Updates Tab (See previous section)
- Limitations Tab (See previous section)

22.5.1 Clock Tab

Select *Debugger>Settings* and then click the Clock tab.

In this dialog you set the clock speed at which you want your code to run. These values will also be used to set certain fields in the Stopwatch.

1. Select the Units of frequency
2. Enter the Frequency value

22.5.2 Break Options Tab

Select *Debugger>Settings* and then click the Break Options tab.

- Global Break Enable – If checked, program breakpoints will be enabled.
- Stack Enable – If checked, the stack will be enabled.
- Enable WDT Expiration Warning – If checked, a message will pop up when the WDT times out.

Stack Options

These options control the simulator's behavior on Stack Full/Underflow

- Disable Stack Full/Underflow Warning

If this is checked, the simulator will suppress warnings when it detects Stack Full and Stack Underflow conditions.

Choose one of the following to determine the simulator behavior when it detects Stack Full and Stack Underflow conditions.

- Break on Stack Full/Underflow
- Reset on Stack Full/Underflow

Note: The Stack group is disabled unless "Stack Enabled" is checked above.

WDT (Watchdog Timer) Options

Choose one of the following to determine the simulator behavior when it detects WDT Time-out.

- Break On WDT time-out
- Reset on WDT time-out

Note: The WDT group is disabled if the WDT is disabled in the Configuration Bits window (*Configure>Configuration Bits*).

22.5.3 Trace/Pins Tab

Select *Debugger>Settings* and then click the **Trace/Pins** tab.

- MCLR Pull-up Enabled - If this is checked, $\overline{\text{MCLR}}$ Pull-up will be enabled.

Trace Options

- Trace Enabled - If checked, instruction execution will be traced.
- Break on Trace Buffer Full - If checked, the simulator will halt when the trace buffer is full.

Appendix A. Revision History

Revision A (10/2004)

- Initial release of this document.

Revision B (04/2006)

- Updated MPLAB SIM simulator information
- Updated debug support from Editor window with Filter In/Out
- Added support information for these items on the Tools menu:
  - Sensorless Motor Tuning Interface (AN901)
  - ACIM Tuning Interface (AN908)
  - DCMI
- Added Find in (All) Files search ability within MPLAB Editor
- Added code folding information
- Added Simulator peripheral support information
- RTOS Viewer support added
- Added Subversion VCS Support information
- Additional corrections and updates throughout document text

Revision C (01/2009)

- Updates in files "WalkThrough", "IDEDesktop", "IDEDialogs".
- Reorganized Parts:
  - Getting Started split into Overview and Tutorials.
  - Integrated tools chapters moved to Overview from Features.
  - Features part expanded.
- Labeled or deleted obsolete tools - MPLAB C17, PRO MATE II, MPLAB ICE 4000.
- Added tool information - MPLAB REAL ICE emulator, PICkit 1 & 2, MATLAB.
- MPLAB IDE Windows:
  - Moved some sections around to place in alphabetically order.
  - New docking windows scheme.
  - New "P" code display window symbol.
  - New Flash Data window.
- MPLAB IDE Dialogs
  - Build Options dialog - General tab updated.
  - Project Wizard - Create the project, Add files dialogs updated.
  - File Management dialog - added Add New File to Project.
  - Select Device dialog - updated.
- MPLAB Editor Properties dialog updated.
- MPLAB SIM - stimulus changed.
- MPLAB SIM - code coverage added.

Minor updates (spelling, etc.) plus the following:

- Chapter 1: Added more program and data memory info. Removed MPLAB IDE release numbering info (old).
- Chapter 2: Removed MPLAB C17 info (no longer sold), renamed C compilers, updated HI-TECH info. Added tool information - MPLAB ICD 3, PICkit 3, PC-Lint.
- Chapter 3: Added "Starter Kits" and "Mature Tools" sections.
- Chapter 4: Updates screen captures.
- Chapter 5: Added tool information - MPLAB ICD 3, PICkit 3.
- Chapter 6: Add MPLAB C32 Toolsuite info.
- Chapter 7: Added auto linker script addition to project, generic linker scripts, MPLAB C32 Toolsuite info.
- Chapter 8: MPLAB ICD 3 info, double click for breakpoint requirements, clarified breakpoints in C code, mouseover for toolbar buttons.
- Chapter 9: Clarified config bits in window vs. code, "External Memory" PIC17 MCU references removed.
- Chapter 10: No updates.
- Chapter 11: Added powered hub info, removed v7.xx info from FAQ - in Readme.
- Chapter 12: Added "Goto Locator", "Go Backward", "Go Forward" items; added block comment formatting info; added more Tool menu items; added info on grayed out items; Package in .zip and Locate Headers.
- Chapter 13: Added Window data update on halt info; added goto, bookmark symbols; External DIFF and other editor commends; Gauge conditions added; Watch window comment column.
- Chapter 14: Clarified custom build; debug formats: added elf and dwarf, removed cod; Settings dialog, External DIFF tab; Watch File Scope dialog.
- Chapter 15: Added Keyboard shortcuts information.
- Chapter 16: Added Format Comment Block info; External DIFF utility; match token highlight; delete whole word, word left.
- Chapter 17: Added PIC32MX architecture, standardized text between architectures.
- Chapter 18: Using Code Coverage.
- Chapter 19: Advanced stimulus dialog updates.
- Chapter 20: No updates.
- Chapter 21: No updates.
- Chapter 22: No updates.

Glossary

Absolute Section

A section with a fixed (absolute) address that cannot be changed by the linker.

Access Memory (PIC18 Only)

Special registers on PIC18 devices that allow access regardless of the setting of the Bank Select Register (BSR).

Address

Value that identifies a location in memory.

Alphabetic Character

Alphabetic characters are those characters that are letters of the arabic alphabet (a, b, …, z, A, B, …, Z).

Alphanumeric

Alphanumeric characters are comprised of alphabetic characters and decimal digits (0,1, …, 9).

ANSI

American National Standards Institute is an organization responsible for formulating and approving standards in the United States.

Application

A set of software and hardware that may be controlled by a PICmicro microcontroller.

Archive

A collection of relocatable object modules. It is created by assembling multiple source files to object files, and then using the archiver to combine the object files into one library file. A library can be linked with object modules and other libraries to create executable code.

Archiver

A tool that creates and manipulates libraries.

ASCII

American Standard Code for Information Interchange is a character set encoding that uses 7 binary digits to represent each character. It includes upper and lower case letters, digits, symbols and control characters.

Assembler

A language tool that translates assembly language source code into machine code.

Assembly Language

A programming language that describes binary machine code in a symbolic form.

Asynchronous Stimulus

Data generated to simulate external inputs to a simulator device.

Bookmarks

Use bookmarks to easily locate specific lines in a file.

Under the Edit menu, select Bookmarks to manage bookmarks. Toggle (enable / disable) a bookmark, move to the next or previous bookmark, or clear all bookmarks.

Breakpoint

Hardware Breakpoint: An event whose execution will cause a halt.

Software Breakpoint: An address where execution of the firmware will halt. Usually achieved by a special break instruction.

Build

Compile and link all the source files for an application.

C

A general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators.

Calibration Memory

A special function register or registers used to hold values for calibration of a PICmicro microcontroller on-board RC oscillator or other device peripherals.

Clean

Under the MPLAB IDE Project menu, Clean removes all intermediary project files, such as object, hex and debug files, for the active project. These files are recreated from other files when a project is built.

COFF

Common Object File Format. An object file of this format contains machine code, debugging and other information.

Command Line Interface

A means of communication between a program and its user based solely on textual input and output.

Compiler

A program that translates a source file written in a high-level language into machine code.

Configuration Bits

Special-purpose bits programmed to set PICmicro microcontroller modes of operation. A Configuration bit may or may not be preprogrammed.

Control Directives

Directives in assembly language code that cause code to be included or omitted based on the assembly-time value of a specified expression.

Cross Reference File

A file that references a table of symbols and a list of files that references the symbol. If the symbol is defined, the first file listed is the location of the definition. The remaining files contain references to the symbol.

Data Directives

Data directives are those that control the assembler's allocation of program or data memory and provide a way to refer to data items symbolically; that is, by meaningful names.

Data Memory

On Microchip MCU and DSC devices, data memory (RAM) is comprised of General Purpose Registers (GPRs) and Special Function Registers (SFRs). Some devices also have EEPROM data memory.

Debugging Information

Compiler and assembler options that, when selected, provide varying degrees of information used to debug application code. See compiler or assembler documentation for details on selecting debug options.

Device Programmer

A tool used to program electrically programmable semiconductor devices such as microcontrollers.

Digital Signal Controller

A microcontroller device with digital signal processing capability, i.e., Microchip dsPIC DSC devices.

Directives

Statements in source code that provide control of the language tool's operation.

Download

Download is the process of sending data from a host to another device, such as an emulator, programmer or target board.

DSC

See Digital Signal Controller.

EEPROM

Electrically Erasable Programmable Read Only Memory. A special type of PROM that can be erased electrically. Data is written or erased one byte at a time. EEPROM retains its contents even when power is turned off.

Emulation

The process of executing software loaded into emulation memory as if it were firmware residing on a microcontroller device.

Emulation Memory

Program memory contained within the emulator.

Emulator

Hardware that performs emulation.

Emulator System

The MPLAB ICE 2000 and MPLAB ICE 4000 emulator systems include the pod, processor module, device adapter, target board, cables, and MPLAB IDE software. The MPLAB REAL ICE system consists of a pod, a driver (and potentially a receiver) card, target board, cables, and MPLAB IDE software.

Environment – IDE

The particular layout of the desktop for application development.

Environment – MPLAB PM3

A folder containing files on how to program a device. This folder can be transferred to a SD/MMC card.

EPROM

Erasable Programmable Read Only Memory. A programmable read-only memory that can be erased usually by exposure to ultraviolet radiation.

Event

A description of a bus cycle which may include address, data, pass count, external input, cycle type (fetch, R/W), and time stamp. Events are used to describe triggers, breakpoints and interrupts.

Export

Send data out of the MPLAB IDE in a standardized format.

Extended Microcontroller Mode

In extended microcontroller mode, on-chip program memory as well as external memory is available. Execution automatically switches to external if the program memory address is greater than the internal memory space of the PIC18 device.

External Label

A label that has external linkage.

External Linkage

A function or variable has external linkage if it can be referenced from outside the module in which it is defined.

External Symbol

A symbol for an identifier which has external linkage. This may be a reference or a definition.

External Symbol Resolution

A process performed by the linker in which external symbol definitions from all input modules are collected in an attempt to resolve all external symbol references. Any external symbol references which do not have a corresponding definition cause a linker error to be reported.

External Input Line

An external input signal logic probe line (TRIGIN) for setting an event based upon external signals.

External RAM

Off-chip Read/Write memory.

File Registers

On-chip data memory, including General Purpose Registers (GPRs) and Special Function Registers (SFRs).

Filter

Determine by selection what data is included/excluded in a trace display or data file.

Flash

A type of EEPROM where data is written or erased in blocks instead of bytes.

FNOP

Forced No Operation. A forced NOP cycle is the second cycle of a two-cycle instruction. Since the PICmicro microcontroller architecture is pipelined, it prefetches the next instruction in the physical address space while it is executing the current instruction. However, if the current instruction changes the program counter, this prefetched instruction is explicitly ignored, causing a forced NOP cycle.

GPR

General Purpose Register. The portion of device data memory (RAM) available for general use.

Halt

A stop of program execution. Executing Halt is the same as stopping at a breakpoint.

Hex Code

Executable instructions stored in a hexadecimal format code. Hex code is contained in a hex file.

Hex File

An ASCII file containing hexadecimal addresses and values (hex code) suitable for programming a device.

High Level Language

A language for writing programs that is further removed from the processor than assembly.

ICD

In-Circuit Debugger. MPLAB ICD 2 and 3, and PICkit 2 and 3 (with Debug Express), are Microchip's in-circuit debuggers.

ICE

In-Circuit Emulator. MPLAB ICE 2000, MPLAB ICE 4000 and MPLAB REAL ICE system are Microchip's in-circuit emulators.

ICSP

In-Circuit Serial Programming. A method of programming Microchip embedded devices using serial communication and a minimum number of device pins.

IDE

Integrated Development Environment. MPLAB IDE is Microchip's integrated development environment.

Import

Bring data into the MPLAB IDE from an outside source, such as from a hex file.

Instruction Set

The collection of machine language instructions that a particular processor understands.

Instructions

A sequence of bits that tells a central processing unit to perform a particular operation and can contain data to be used in the operation.

Internal Linkage

A function or variable has internal linkage if it can not be accessed from outside the module in which it is defined.

International Organization for Standardization

An organization that sets standards in many businesses and technologies, including computing and communications.

Interrupt

A signal to the CPU that suspends the execution of a running application and transfers control to an Interrupt Service Routine (ISR) so that the event may be processed. Upon completion of the ISR, normal execution of the application resumes.

Interrupt Handler

A routine that processes special code when an interrupt occurs.

Interrupt Request

An event which causes the processor to temporarily suspend normal instruction execution and to start executing an interrupt handler routine. Some processors have several interrupt request events allowing different priority interrupts.

Interrupt Service Routine

User-generated code that is entered when an interrupt occurs. The location of the code in program memory will usually depend on the type of interrupt that has occurred.

IRQ

See Interrupt Request.

ISO

See International Organization for Standardization.

ISR

See Interrupt Service Routine.

Librarian

See Archiver.

Library

See Archive.

Linker

A language tool that combines object files and libraries to create executable code, resolving references from one module to another.

Linker Script Files

Linker script files are the command files of a linker. They define linker options and describe available memory on the target platform.

Listing Directives

Listing directives are those directives that control the assembler listing file format. They allow the specification of titles, pagination and other listing control.

Listing File

A listing file is an ASCII text file that shows the machine code generated for each C source statement, assembly instruction, assembler directive, or macro encountered in a source file.

Local Label

A local label is one that is defined inside a macro with the LOCAL directive. These labels are particular to a given instance of a macro's instantiation. In other words, the symbols and labels that are declared as local are no longer accessible after the ENDM macro is encountered.

Logic Probes

Up to 14 logic probes can be connected to some Microchip emulators. The logic probes provide external trace inputs, trigger output signal, +5V, and a common ground.

Machine Code

The representation of a computer program that is actually read and interpreted by the processor. A program in binary machine code consists of a sequence of machine instructions (possibly interspersed with data). The collection of all possible instructions for a particular processor is known as its "instruction set".

Machine Language

A set of instructions for a specific central processing unit, designed to be usable by a processor without being translated.

Macro

Macro instruction. An instruction that represents a sequence of instructions in abbreviated form.

Macro Directives

Directives that control the execution and data allocation within macro body definitions.

Makefile

Export to a file the instructions to Make the project. Use this file to Make your project outside of MPLAB IDE, i.e., with a `make`.

Under *Project>Build Options>Project*, **Directories** tab, you must have selected "Assemble/Compile/Link in the project directory" under "Build Directory Policy" for this feature to work.

Make Project

A command that rebuilds an application, recompiling only those source files that have changed since the last complete compilation.

MCU

Microcontroller Unit. An abbreviation for microcontroller. Also uC.

Message

Text displayed to alert you to potential problems in language tool operation. A message will not stop operation.

Microcontroller

A highly integrated chip that contains a CPU, RAM, program memory, I/O ports and timers.

Microcontroller Mode

One of the possible program memory configurations of PIC18 microcontrollers. In microcontroller mode, only internal execution is allowed. Thus, only the on-chip program memory is available in microcontroller mode.

Microprocessor Mode

One of the possible program memory configurations of PIC18 microcontrollers. In microprocessor mode, the on-chip program memory is not used. The entire program memory is mapped externally.

Mnemonics

Text instructions that can be translated directly into machine code. Also referred to as opcodes.

MPASM™ Assembler

Microchip Technology's relocatable macro assembler for PICmicro microcontroller devices, KeeLoq® devices and Microchip memory devices.

MPLAB ASM30/LINK30/LIB30

Previous names for Microchip's relocatable macro assembler, object linker and object archiver/librarian supporting 16-bit devices (dsPIC30F/33F DSCs and PIC24H/F MCUs.)

MPLAB C18/C30/C32

Previous names for various C compilers from Microchip. MPLAB C18 supports PIC18CXXX and PIC18FXXXX devices, MPLAB C30 supports dsPIC30F/33F DSCs and PIC24H/F MCUs and MPLAB C32 supports PIC32MX devices.

MPLAB *Language Tool* for *Device*

Microchip's C compilers, assemblers and linkers for specified devices. Select the type of language tool based on the device you will be using for your application, e.g., if you will be creating C code on a PIC18 MCU, select the MPLAB C Compiler for PIC18 MCUs.

MPLAB ICD 2/3

Microchip's in-circuit debuggers that works with MPLAB IDE. The ICDs supports Flash devices with built-in debug circuitry. The main component of each ICD is the pod. A complete system consists of a pod, header board (with a *device*-ICD), target board, cables, and MPLAB IDE software.

MPLAB ICE 2000/4000

Not recommended for new designs. See the MPLAB REAL ICE in-circuit emulator.

Microchip's in-circuit emulators that work with MPLAB IDE. MPLAB ICE 2000 supports 8-bit PIC MCUs. MPLAB ICE 4000 supports PIC18F and PIC24 MCUs and dsPIC DSCs. The main component of each ICE is the pod. A complete system consists of a pod, processor module, cables, and MPLAB IDE software.

MPLAB IDE

Microchip's Integrated Development Environment.

MPLAB PM3

A device programmer from Microchip. Programs PIC18 microcontrollers and dsPIC digital signal controllers. Can be used with MPLAB IDE or stand-alone. Will replace PRO MATE II.

MPLAB REAL ICE™ In-Circuit Emulator

Microchip's in-circuit emulators that works with MPLAB IDE. The MPLAB REAL ICE emulator supports PIC18F and PIC24 MCUs and dsPIC DSCs. The main component of each ICE is the pod. A complete system consists of a pod, a driver (and potentially a receiver) card, cables, and MPLAB IDE software.

MPLAB SIM

Microchip's simulator that works with MPLAB IDE in support of PICmicro MCU and dsPIC DSC devices.

MPLIB™ Object Librarian

Microchip's librarian that can work with MPLAB IDE. MPLIB librarian is an object librarian for use with COFF object modules created using either MPASM assembler (mpasm or mpasmwin v2.0) or MPLAB C18 C compiler.

MPLINK™ Object Linker

MPLINK linker is an object linker for the Microchip MPASM assembler and the Microchip C18 C compiler. MPLINK linker also may be used with the Microchip MPLIB librarian. MPLINK linker is designed to be used with MPLAB IDE, though it does not have to be.

MRU

Most Recently Used. Refers to files and windows available to be selected from MPLAB IDE main pull down menus.

Nesting Depth

The maximum level to which macros can include other macros.

Node

MPLAB IDE project component.

Non Real Time

Refers to the processor at a breakpoint or executing single-step instructions or MPLAB IDE being run in simulator mode.

Non-Volatile Storage

A storage device whose contents are preserved when its power is off.

NOP

No Operation. An instruction that has no effect when executed except to advance the program counter.

Object Code

The machine code generated by an assembler or compiler.

Object File

A file containing machine code and possibly debug information. It may be immediately executable or it may be relocatable, requiring linking with other object files, e.g., libraries, to produce a complete executable program.

Object File Directives

Directives that are used only when creating an object file.

Off-Chip Memory

Off-chip memory refers to the memory selection option for the PIC18 device where memory may reside on the target board, or where all program memory may be supplied by the emulator. The **Memory** tab accessed from *Options>Development Mode* provides the Off-Chip Memory selection dialog box.

One-to-One Project-Workspace Model

The most common configuration for application development in MPLAB IDE to is have one project in one workspace. Select *Configure>Settings*, **Projects** tab and check “Use one-to-one project-workspace model”.

Opcodes

Operational Codes. *See* Mnemonics.

Operators

Symbols, like the plus sign ‘+’ and the minus sign ‘-’, that are used when forming well-defined expressions. Each operator has an assigned precedence that is used to determine order of evaluation.

OTP

One Time Programmable. EPROM devices that are not in windowed packages. Since EPROM needs ultraviolet light to erase its memory, only windowed devices are erasable.

Pass Counter

A counter that decrements each time an event (such as the execution of an instruction at a particular address) occurs. When the pass count value reaches zero, the event is satisfied. You can assign the Pass Counter to break and trace logic, and to any sequential event in the complex trigger dialog.

PC

Personal Computer or Program Counter.

PC Host

Any PC running a supported Windows operating system.

PICmicro MCUs

PICmicro microcontrollers (MCUs) refers to all Microchip microcontroller families.

PICSTART Plus

A developmental device programmer from Microchip. Programs 8-, 14-, 28-, and 40-pin PICmicro microcontrollers. Must be used with MPLAB IDE software.

Plug-ins

The MPLAB IDE has both built-in components and plug-in modules to configure the system for a variety of software and hardware tools. Several plug-in tools may be found under the Tools menu.

Pod

MPLAB REAL ICE system: The box that contains the emulation control circuitry for the ICE device on the header or target board. An ICE device can be a production device with built-in ICE circuitry or a special ICE version of a production device (i.e., *device*-ICE).

MPLAB ICD 2/3: The box that contains the debug control circuitry for the ICD device on the header or target board. An ICD device can be a production device with built-in ICD circuitry or a special ICD version of a production device (i.e., *device*-ICD).

MPLAB ICE 2000/4000: The external emulator box that contains emulation memory, trace memory, event and cycle timers, and trace/breakpoint logic.

Power-on-Reset Emulation

A software randomization process that writes random values in data RAM areas to simulate uninitialized values in RAM upon initial power application.

PRO MATE II

No longer in Production. See the MPLAB PM3 device programmer.

A device programmer from Microchip. Programs most PICmicro microcontrollers as well as most memory and KEELOQ devices. Can be used with MPLAB IDE or stand-alone.

Profile

For MPLAB SIM simulator, a summary listing of executed stimulus by register.

Program Counter

The location that contains the address of the instruction that is currently executing.

Program Memory

The memory area in a device where instructions are stored. Also, the memory in the emulator or simulator containing the downloaded target application firmware.

Project

A project contains the files needed to build an application (source code, linker script files, etc.) along with their associations to various build tools and build options.

Prototype System

A term referring to a user's target application, or target board.

PWM Signals

Pulse Width Modulation Signals. Certain PICmicro MCU devices have a PWM peripheral.

Qualifier

An address or an address range used by the Pass Counter or as an event before another operation in a complex trigger.

Radix

The number base, hex, or decimal, used in specifying an address.

RAM

Random Access Memory (Data Memory). Memory in which information can be accessed in any order.

Raw Data

The binary representation of code or data associated with a section.

Read Only Memory

Memory hardware that allows fast access to permanently stored data but prevents addition to or modification of the data.

Real Time

When an in-circuit emulator or debugger is released from the halt state, the processor runs in Real Time mode and behaves exactly as the normal chip would behave. In Real Time mode, the real time trace buffer of an emulator is enabled and constantly captures all selected cycles, and all break logic is enabled. In an in-circuit emulator or debugger, the processor executes in real time until a valid breakpoint causes a halt, or until the user halts the execution.

In the simulator, real time simply means execution of the microcontroller instructions as fast as they can be simulated by the host CPU.

Real-Time Watch

A Watch window where the variables change in real-time as the application is run. See individual tool documentation to determine how to set up a real-time watch. Not all tools support real-time watches.

Recursion

The concept that a function or macro, having been defined, can call itself. Great care should be taken when writing recursive macros; it is easy to get caught in an infinite loop where there will be no exit from the recursion.

ROM

Read Only Memory (Program Memory). Memory that cannot be modified.

Run

The command that releases the emulator from halt, allowing it to run the application code and change or respond to I/O in real time.

Scenario

For MPLAB SIM simulator, a particular setup for stimulus control.

SFR

See Special Function Registers.

Shell

The MPASM assembler shell is a prompted input interface to the macro assembler. There are two MPASM assembler shells: one for the DOS version and one for the Windows version.

Simulator

A software program that models the operation of devices.

Single Step

This command steps though code, one instruction at a time. After each instruction, MPLAB IDE updates register windows, watch variables, and status displays so you can analyze and debug instruction execution. You can also single step C compiler source code, but instead of executing single instructions, MPLAB IDE will execute all assembly level instructions generated by the line of the high level C statement.

Skew

The information associated with the execution of an instruction appears on the processor bus at different times. For example, the executed opcodes appears on the bus as a fetch during the execution of the previous instruction, the source data address and

value and the destination data address appear when the opcodes is actually executed, and the destination data value appears when the next instruction is executed. The trace buffer captures the information that is on the bus at one instance. Therefore, one trace buffer entry will contain execution information for three instructions. The number of captured cycles from one piece of information to another for a single instruction execution is referred to as the skew.

Skid

When a hardware breakpoint is used to halt the processor, one or more additional instructions may be executed before the processor halts. The number of extra instructions executed after the intended breakpoint is referred to as the skid.

Source Code

The form in which a computer program is written by the programmer. Source code is written in a formal programming language which can be translated into machine code or executed by an interpreter.

Source File

An ASCII text file containing source code.

Special Function Registers

The portion of data memory (RAM) dedicated to registers that control I/O processor functions, I/O status, timers or other modes or peripherals.

Stack, Hardware

Locations in PICmicro microcontroller where the return address is stored when a function call is made.

Stack, Software

Memory used by an application for storing return addresses, function parameters, and local variables. This memory is typically managed by the compiler when developing code in a high-level language.

Static RAM or SRAM

Static Random Access Memory. Program memory you can read/write on the target board that does not need refreshing frequently.

Status Bar

The Status Bar is located on the bottom of the MPLAB IDE window and indicates such current information as cursor position, development mode and device, and active tool bar.

Step Into

This command is the same as Single Step. Step Into (as opposed to Step Over) follows a CALL instruction into a subroutine.

Step Over

Step Over allows you to debug code without stepping into subroutines. When stepping over a `CALL` instruction, the next breakpoint will be set at the instruction after the `CALL`. If for some reason the subroutine gets into an endless loop or does not return properly, the next breakpoint will never be reached. The Step Over command is the same as Single Step except for its handling of `CALL` instructions.

Step Out

Step Out allows you to step out of a subroutine which you are currently stepping through. This command executes the rest of the code in the subroutine and then stops execution at the return address to the subroutine.

Stimulus

Input to the simulator, i.e., data generated to exercise the response of simulation to external signals. Often the data is put into the form of a list of actions in a text file. Stimulus may be asynchronous, synchronous (pin), clocked and register.

Stopwatch

A counter for measuring execution cycles.

Symbol

A symbol is a general purpose mechanism for describing the various pieces which comprise a program. These pieces include function names, variable names, section names, file names, struct/enum/union tag names, etc. Symbols in MPLAB IDE refer mainly to variable names, function names and assembly labels. The value of a symbol after linking is its value in memory.

System Window Control

The system window control is located in the upper left corner of windows and some dialogs. Clicking on this control usually pops up a menu that has the items “Minimize,” “Maximize,” and “Close.”

Target

Refers to user hardware.

Target Application

Software residing on the target board.

Target Board

The circuitry and programmable device that makes up the target application.

Target Processor

The microcontroller device on the target application board.

Template

Lines of text that you build for inserting into your files at a later time. The MPLAB Editor stores templates in template files.

Tool Bar

A row or column of icons that you can click on to execute MPLAB IDE functions.

Trace

An emulator or simulator function that logs program execution. The emulator logs program execution into its trace buffer which is uploaded to MPLAB IDE’s trace window.

Trace Memory

Trace memory contained within the emulator. Trace memory is sometimes called the trace buffer.

Trigger Output

Trigger output refers to an emulator output signal that can be generated at any address or address range, and is independent of the trace and breakpoint settings. Any number of trigger output points can be set.

Uninitialized Data

Data which is defined without an initial value. In C,

```
int myVar;
```

defines a variable which will reside in an uninitialized data section.

Upload

The Upload function transfers data from a tool, such as an emulator or programmer, to the host PC or from the target board to the emulator.

USB

Universal Serial Bus. An external peripheral interface standard for communication between a computer and external peripherals over a cable using bi-serial transmission. USB 1.0/1.1 supports data transfer rates of 12 Mbps. Also referred to as high-speed USB, USB 2.0 supports data rates up to 480 Mbps.

Warning

An alert that is provided to warn you of a situation that would cause physical damage to a device, software file, or equipment.

Watch Variable

A variable that you may monitor during a debugging session in a Watch window.

Watch Window

Watch windows contain a list of watch variables that are updated at each breakpoint.

Watchdog Timer

A timer on a PICmicro microcontroller that resets the processor after a selectable length of time. The WDT is enabled or disabled and set up using Configuration bits.

WDT

See Watchdog Timer.

Workbook

For MPLAB SIM stimulator, a setup for generation of SCL stimulus.

WorkSpace

A workspace contains MPLAB IDE information on the selected device, selected debug tool and/or programmer, open windows and their location, and other IDE configuration settings.

Index

NOTES:

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4080

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/16/09

2. Phase Control

(a) Connect up the circuit as shown in Figure 2.

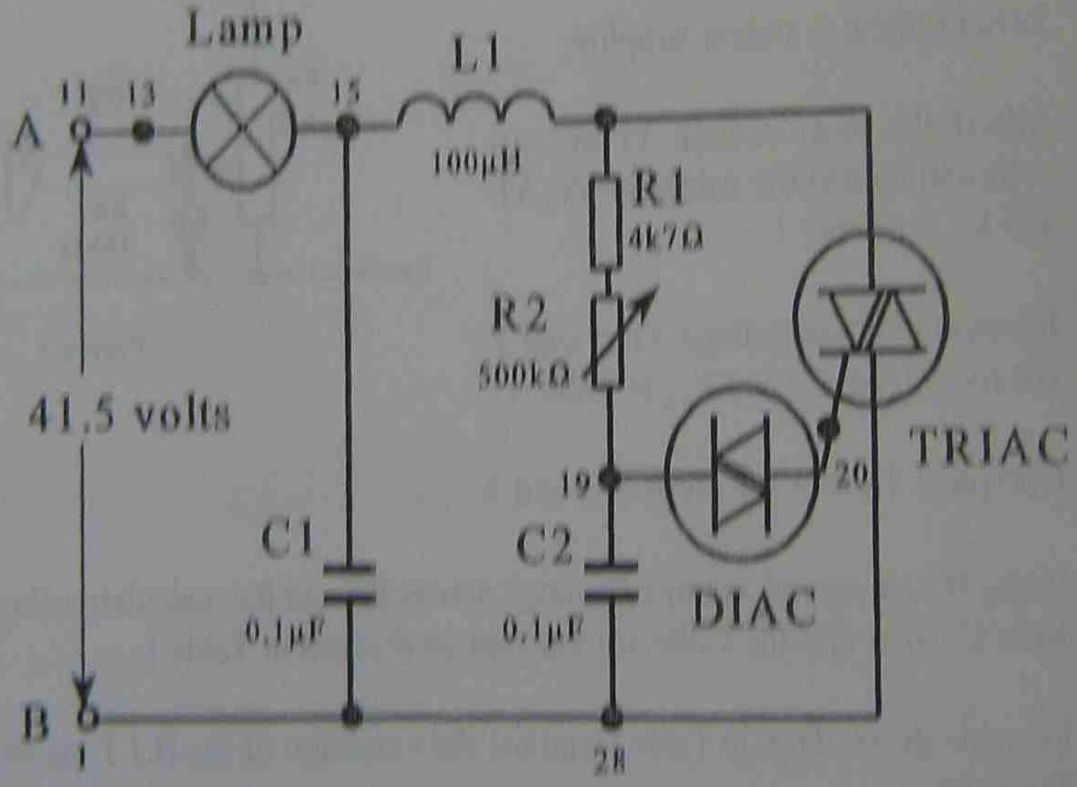
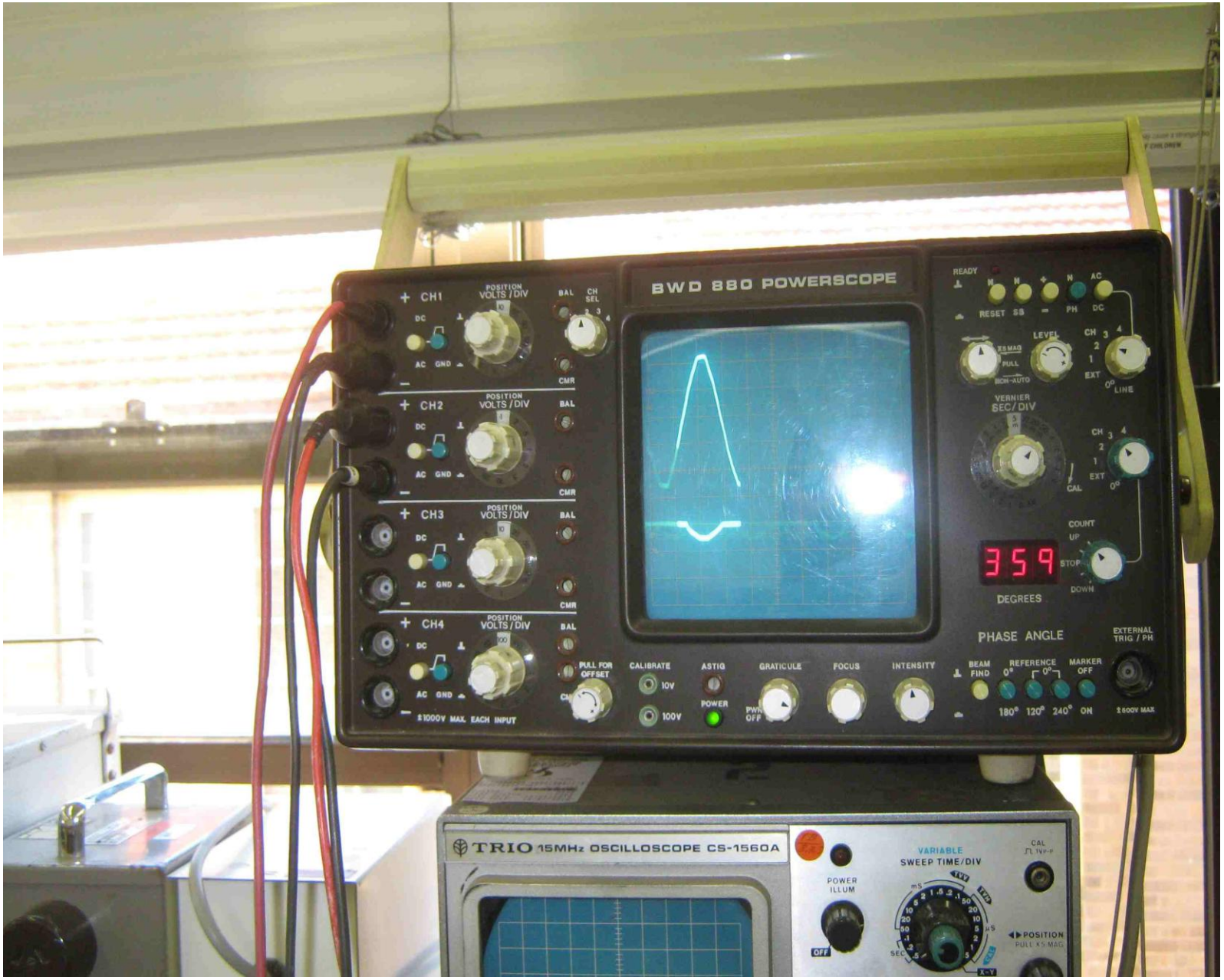
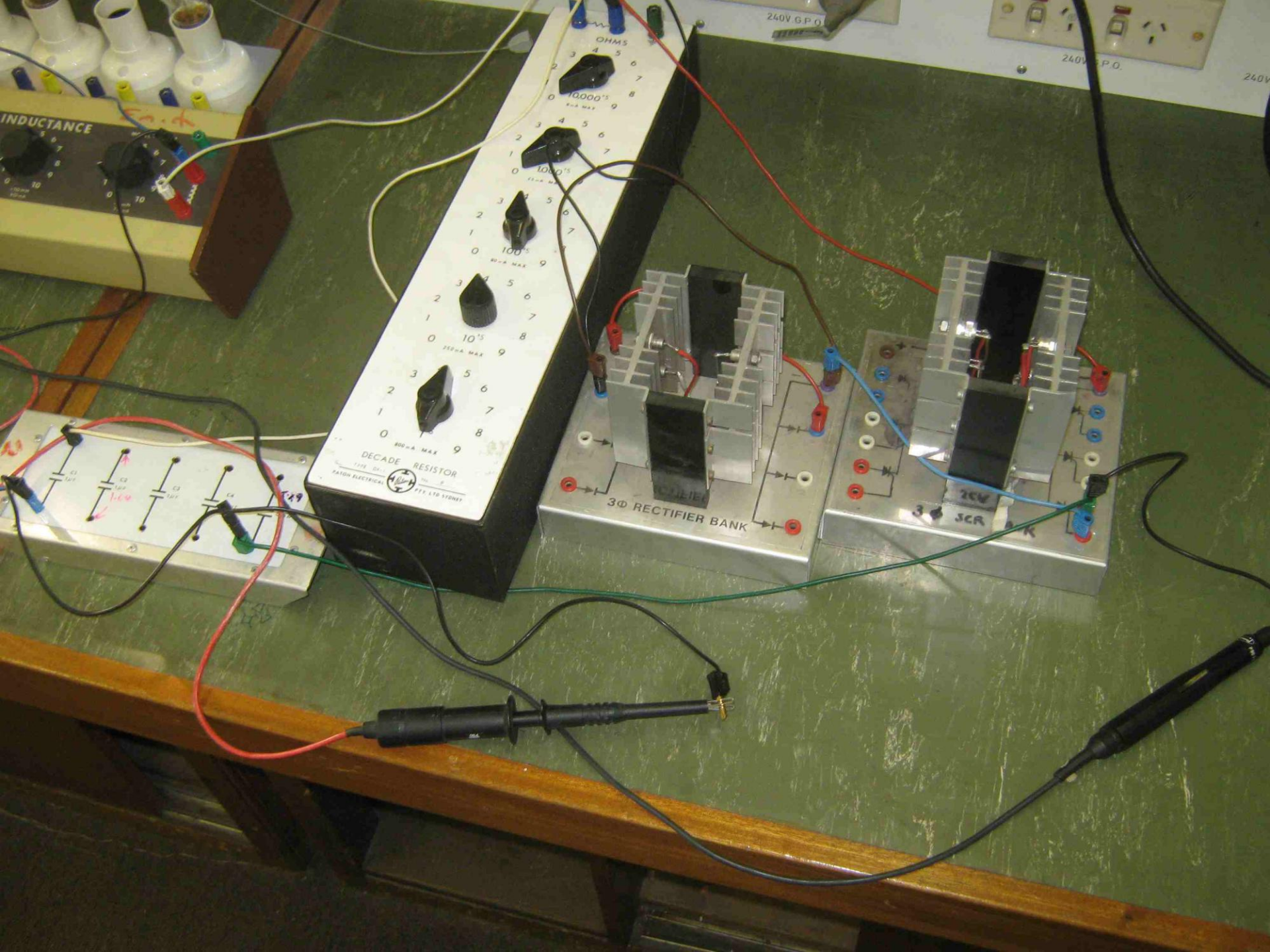


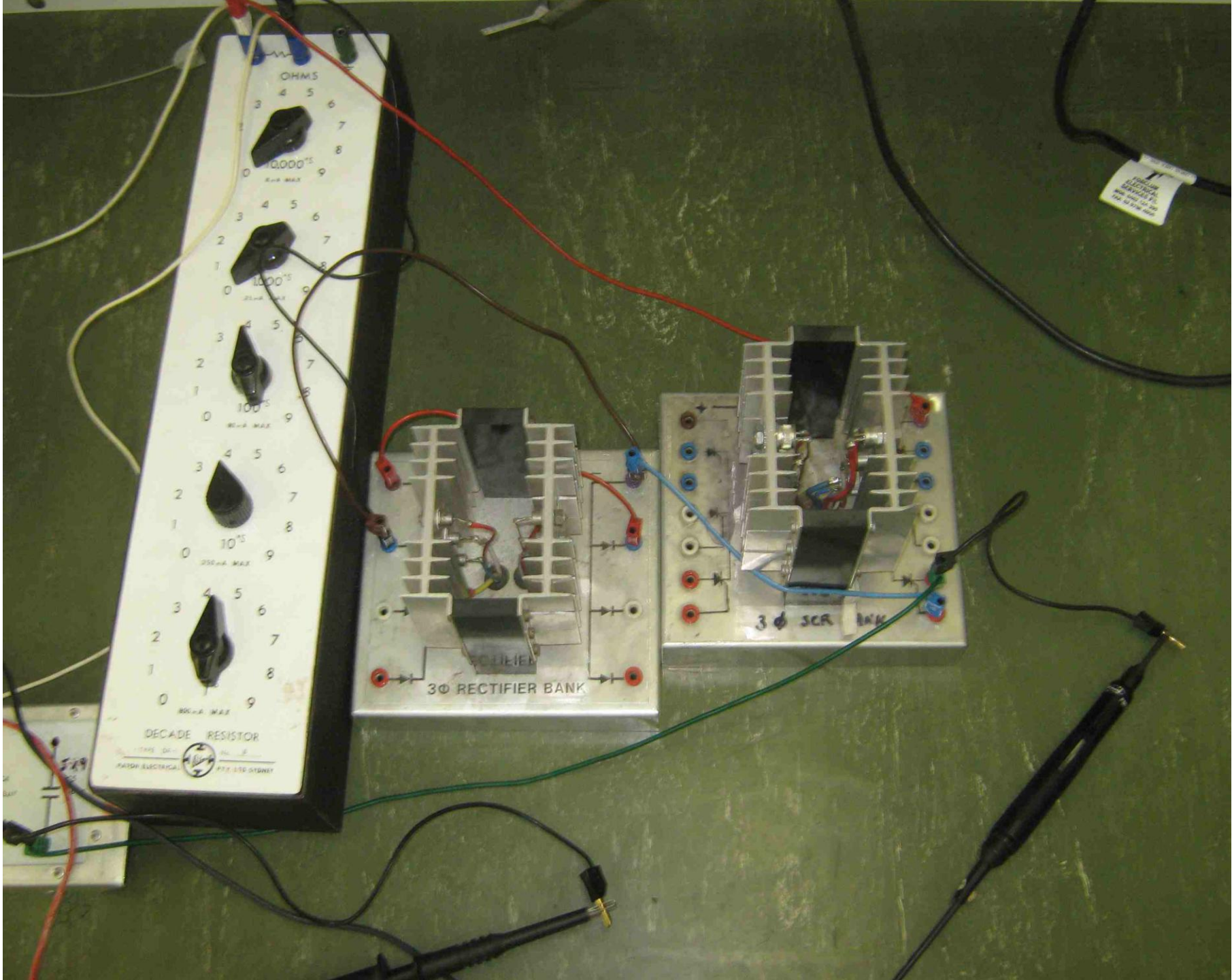
Figure 2

(b) Turn on the 41.5 volt a.c. supply.


BWD 880 POWERSCOPE
CH1
CH2
CH3
CH4
POSITION VOLTS/DIV
BAL
CMR
CH SEL
DC
AC
GND
READY
RESET
SS
PH
LEVEL
VERNIER SEC/DIV
CAL
COUNT UP
STOP
DOWN
359
DEGREES
PHASE ANGLE
EXTERNAL TRIG / PH
PULL FOR OFFSET
CALIBRATE
10V
100V
ASTIG
POWER
GRATICULE
FOCUS
INTENSITY
BEAM FIND
REFERENCE
MARKER OFF
180°
120°
240°
ON
±1000V MAX EACH INPUT
±600V MAX
TRIO 15MHz OSCILLOSCOPE CS-1560A
POWER ILLUM
OFF
VARIABLE SWEEP TIME/DIV
CAL
POSITION
PULL X5 MAG

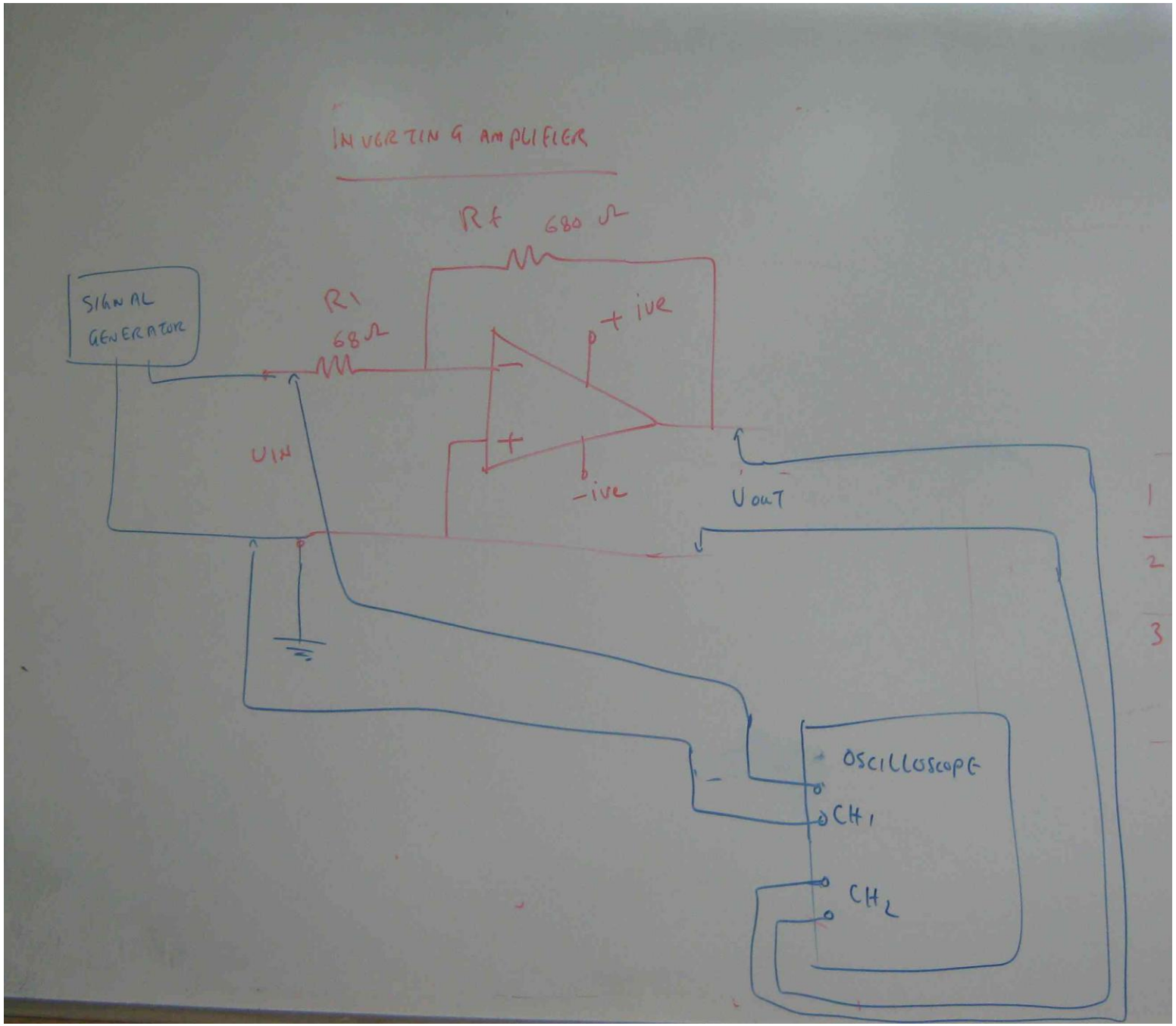

D.C LOW VOLTAGE
240V.A.C
REGULATED SUPPLY
240V.O.P.O.
H.T STANDBY
& TRIP
INDICATOR
250V.D.C
6.3V.A.C
240V.O.P.O.
240V MAIN SWITCH


INDUCTANCE
OHMS
DECADE RESISTOR
3Φ RECTIFIER BANK
3Φ SCR
240V G.P.O.


OHMS
10,000^S
1000^S
100^S
10^S
DECADE RESISTOR
3Φ RECTIFIER BANK
3 ϕ SCR


INDUCTANCE
240V A.C
REGULATED SUPPLY
240V.G.P.O
DECADE RESISTOR
3Φ RECTIFIER BANK


240V A.C.
REGULATED SUPPLY
INDICATOR
250V D.C.
0-280V
VARIAC OUTPUT ONLY
240V G.P.O.
INDUCTANCE
DECADE RESISTOR
3Φ RECTIFIER BANK


INVERTING AMPLIFIER
Rf 680 Ω
SIGNAL GENERATOR
R1 68Ω
+ ive
- ive
VIN
VOUT
OSCILLOSCOPE
CH1
CH2

INJECT INPUT VOLTAGE & FREQUENCY

SKETCH INPUT & OUTPUT WAVE FORMS FOR

	INPUT VOLTAGE	INPUT FREQUENCY	INPUT WAVE SKETCH	INPUT PEAK TO PEAK VALUE	INPUT RMS = $\frac{V_{p-p}}{\sqrt{2}}$	OUTPUT WAVE
1						
2						
3						

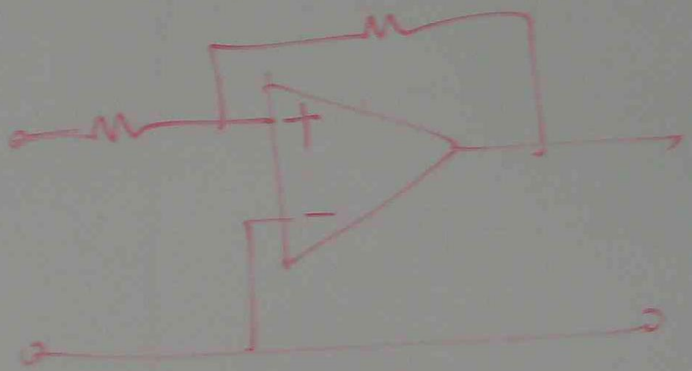
IN PUT PEAK TO PEAK VALUE	IN PUT RMS = $\frac{V_{p-p}}{\sqrt{2}}$	OUT PUT WAVE SKETCH	O/P PEAK TO PEAK	OUT PUT RMS	$A_V = \frac{V_o}{V_i}$

#

IN REAL DESIGN 68Ω ⟶ 68K

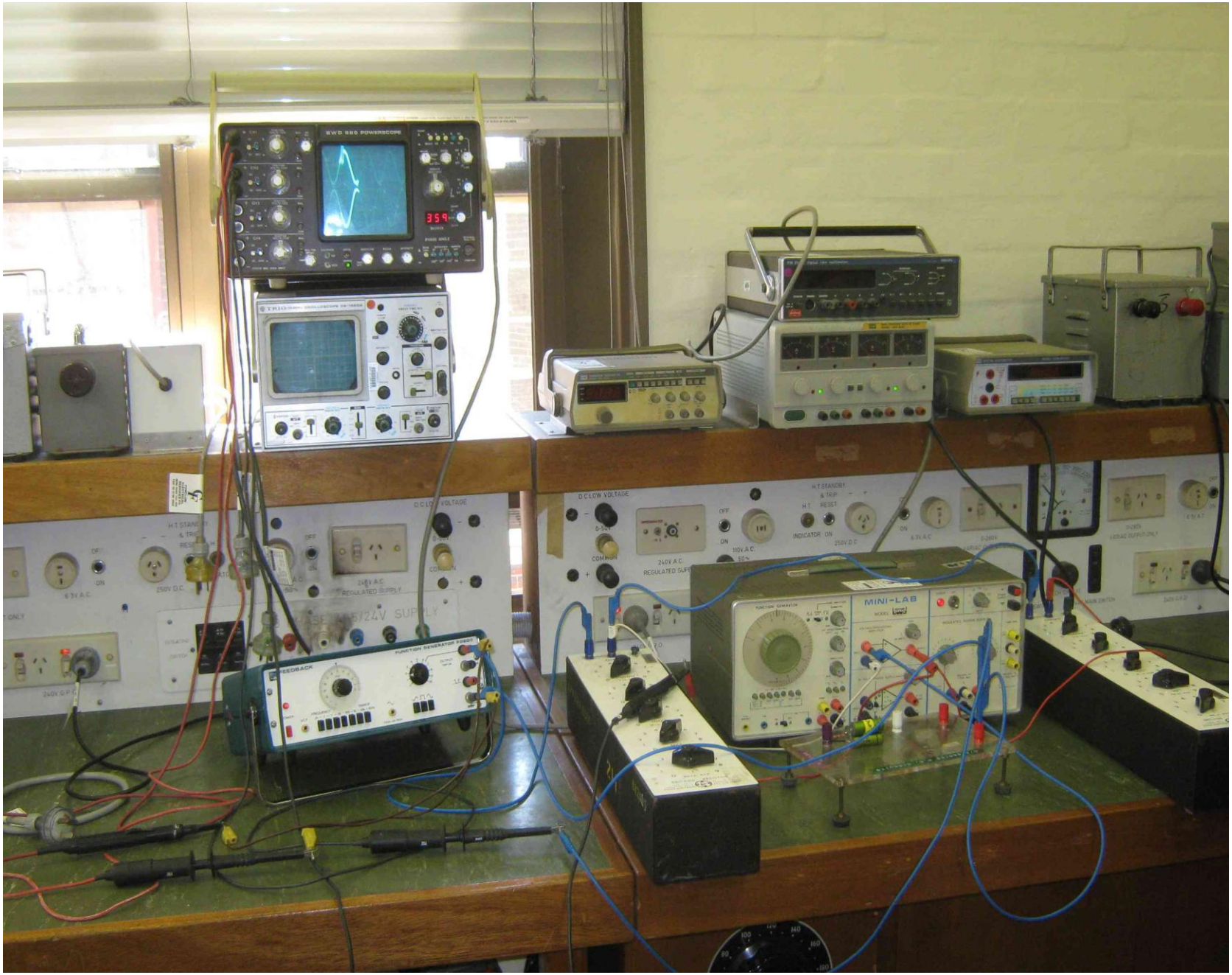
680Ω ⟶ 680K

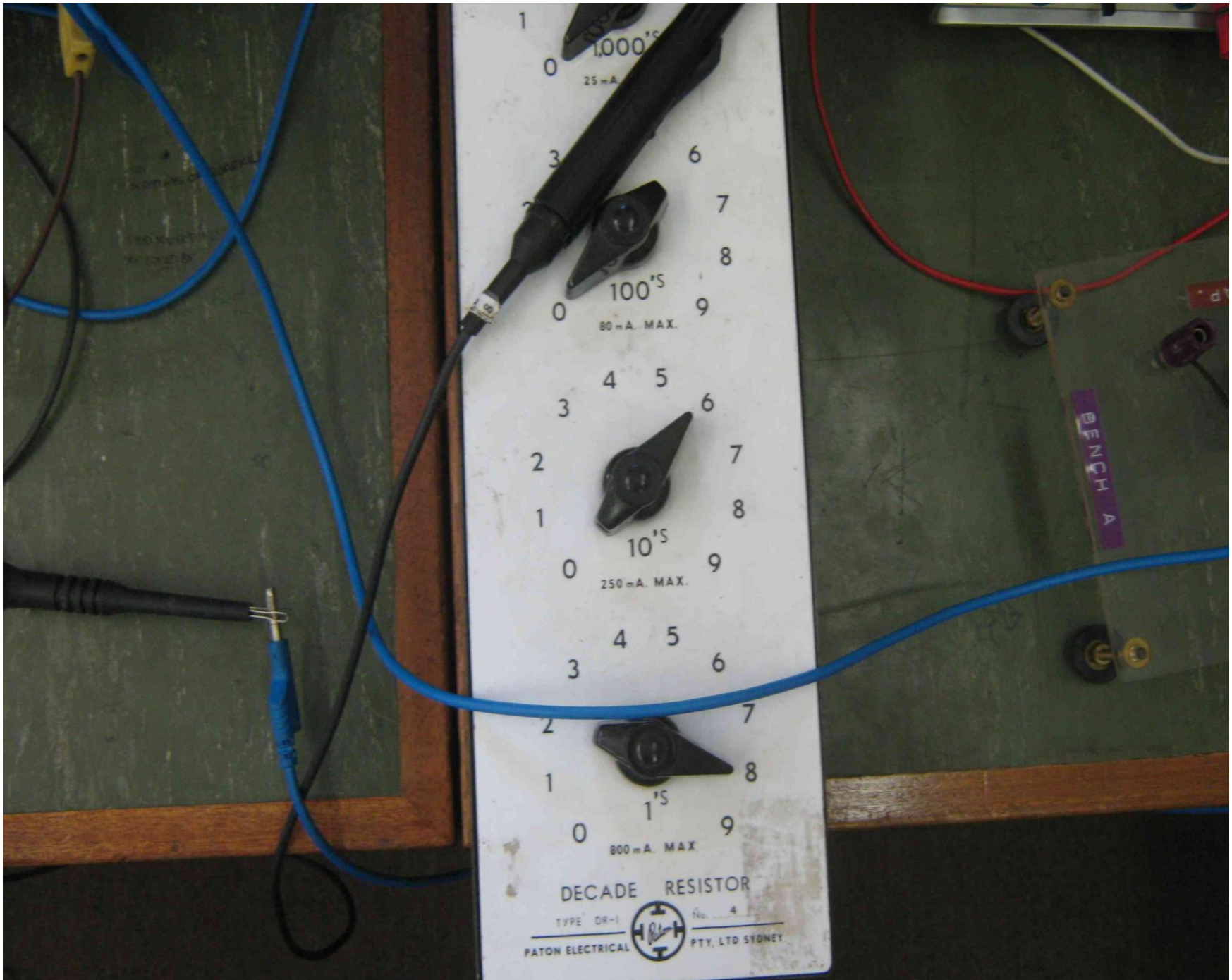
CONNECT THE FOLLOWING CIRCUIT

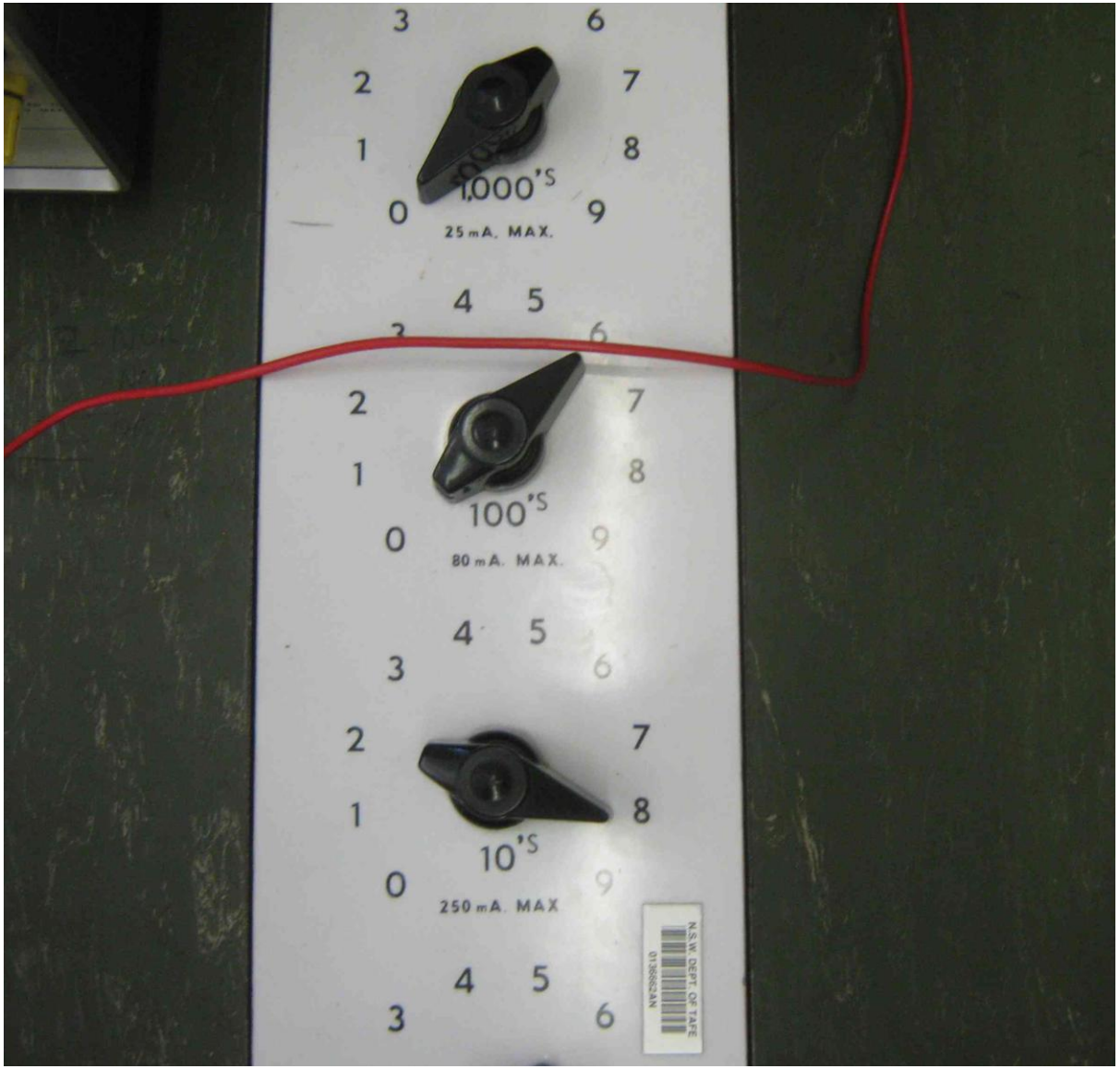


SHOW WHERE WILL YOU MEASURE

INPUT & OUTPUT SIGNALS.

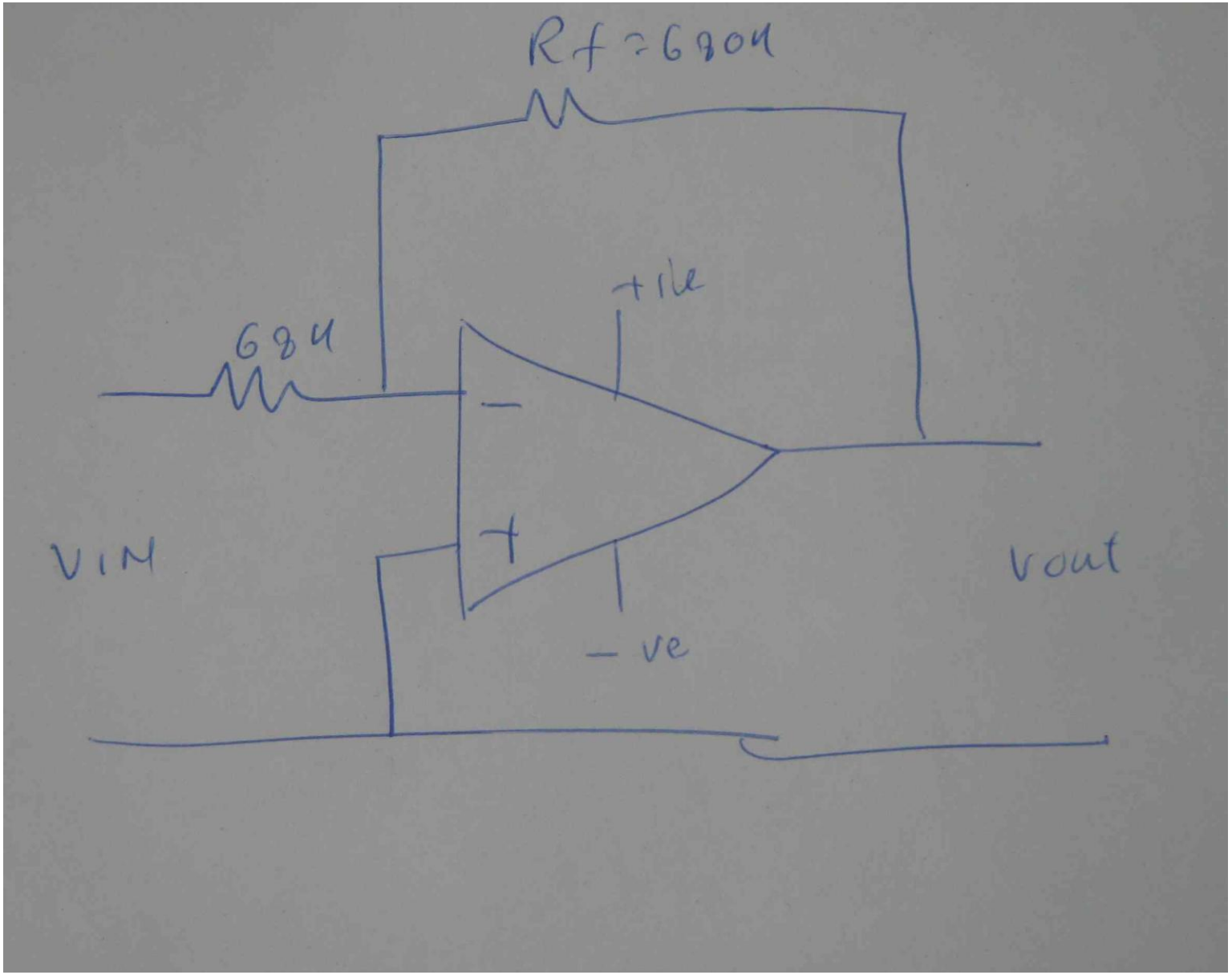

SWD 880 POWERSCOPE
359
D.C LOW VOLTAGE
H.T STANDBY & TRIP
H.T RESET
INDICATOR ON
240V.A.C. REGULATED SUPPLY
110V.A.C.
250V.D.C
6.3V.A.C
FEEDBACK
FUNCTION GENERATOR
MINI-LAB

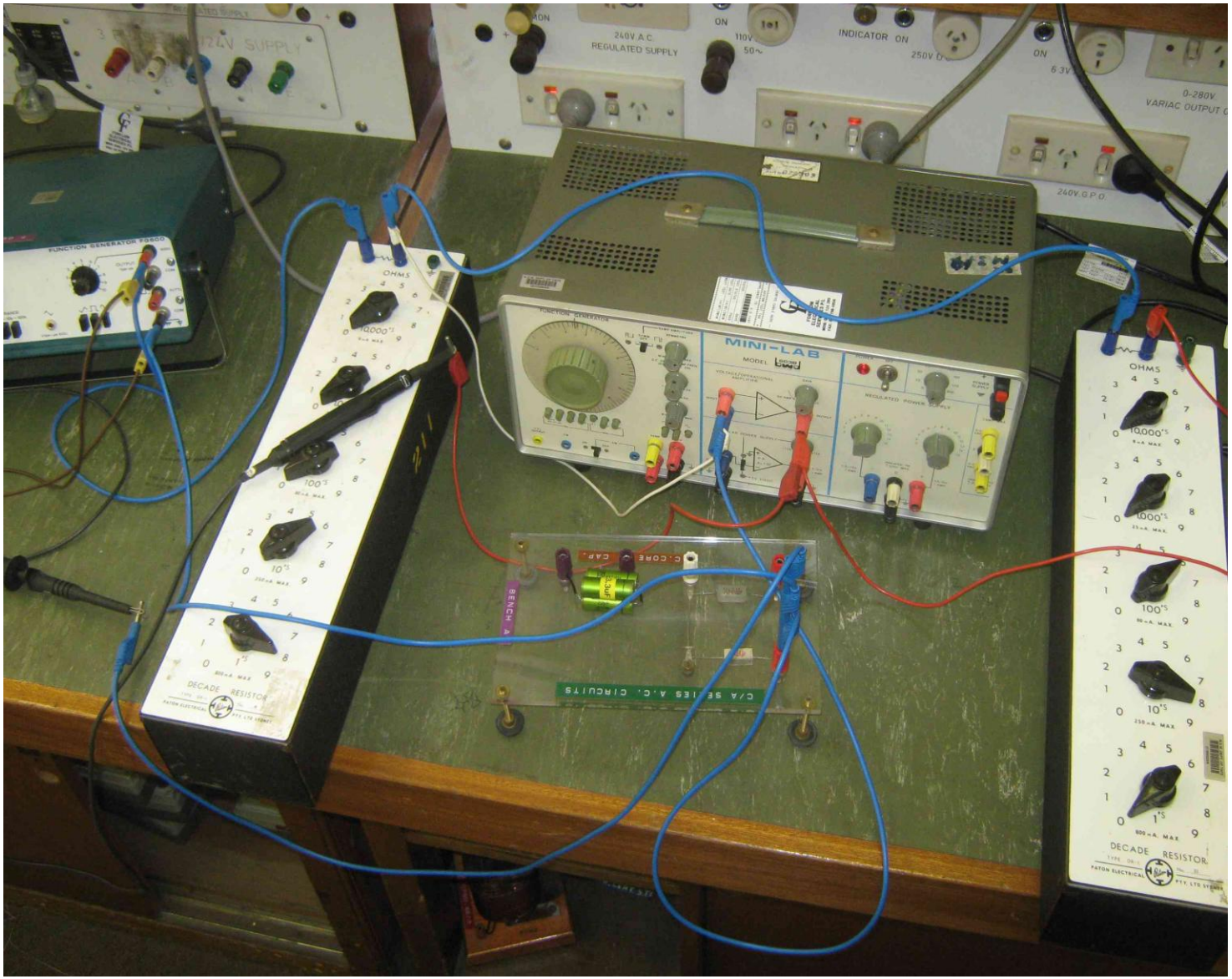

1,000's
25 mA.
100's
80 mA. MAX.
10's
250 mA. MAX.
1's
800 mA. MAX
DECADE RESISTOR
TYPE DR-1
No. 4
PATON ELECTRICAL
PTY. LTD SYDNEY
BENCH A

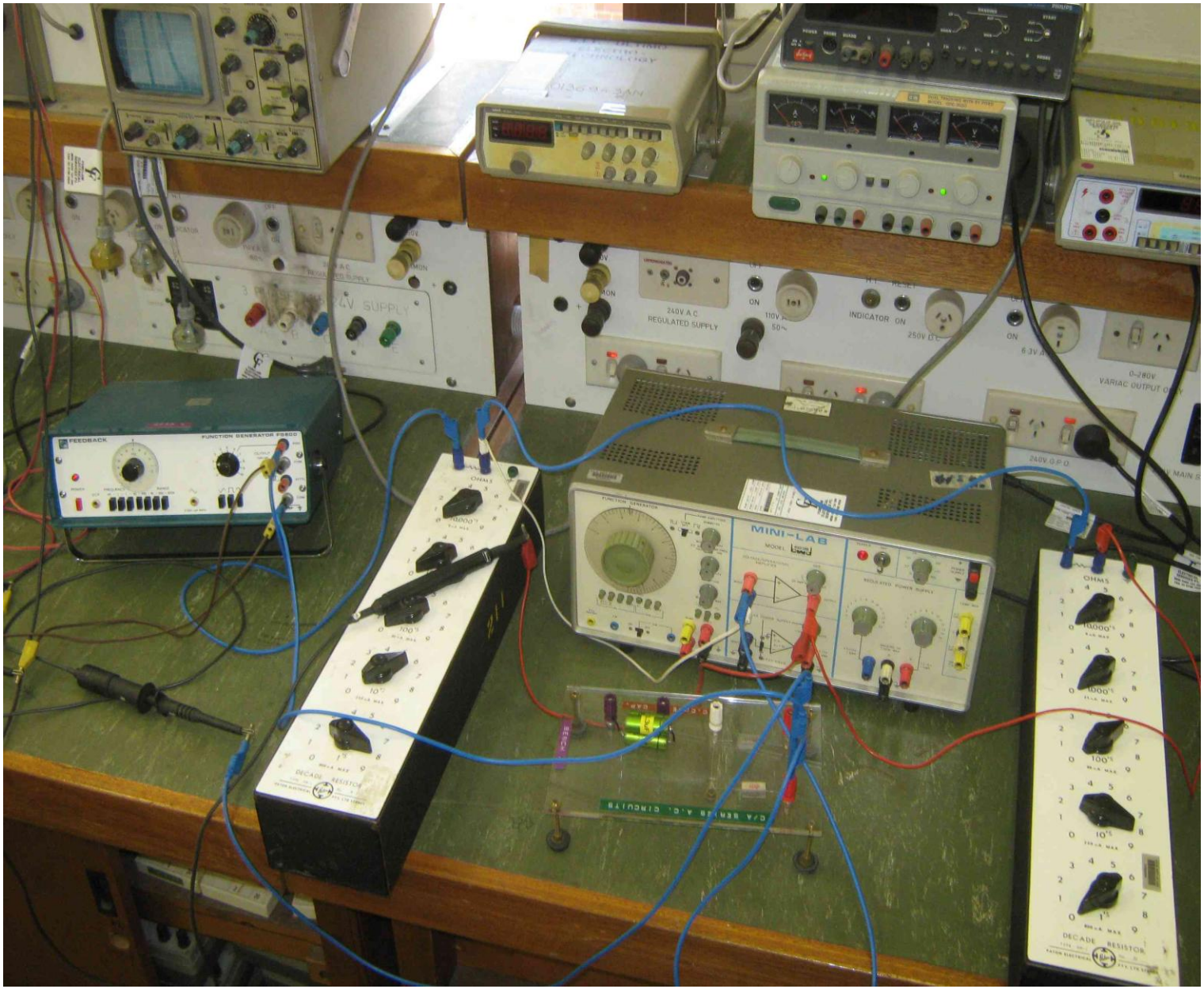

1,000'S
25 mA. MAX.
100'S
80 mA. MAX.
10'S
250 mA. MAX
N.S.W. DEPT. OF TAFE


FEEDBACK
FUNCTION GENERATOR FG6
Hz
POWER
VCF
FREQUENCY
.01 – .1 – 1 – 10 – 100 – 1k – 10k – 100k
RANGE
2Vpk-pk 600Ω
OUTPUT
Vpk-pk


BWD 880 POWERSCOPE
+ CH1
POSITION
VOLTS / DIV
BAL
CH SEL
DC
AC GND
CMR
+ CH2
+ CH3
+ CH4
±1000V MAX. EACH INPUT
PULL FOR OFFSET
CALIBRATE
10V
100V
ASTIG
POWER
GRATICULE
PWR OFF
FOCUS
INTENSITY
READY
RESET
SS
PH
LEVEL
AUTO
VERNIER
SEC/DIV
CAL
359
DEGREES
COUNT
UP
STOP
DOWN
PHASE ANGLE
BEAM FIND
REFERENCE
MARKER
OFF
180°
120°
240°
ON

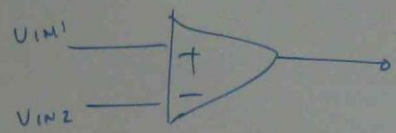

Rf = 680Ω
+ ve
680Ω
VIN
Vout
- ve


240V A.C.
REGULATED SUPPLY
INDICATOR ON
0-280V
VARIAC OUTPUT
240V.G.P.O.
FUNCTION GENERATOR
MINI-LAB
MODEL
OHMS
DECADE RESISTOR
C/A SERIES A.C. CIRCUITS
BENCH A


FEEDBACK
FUNCTION GENERATOR FG600
240V A.C.
REGULATED SUPPLY
INDICATOR ON
MINI-LAB
DECADE RESISTOR

PRACTICAL

OPERATIONAL AMPLIFIER COMPARATOR



OP-AMP COMPARES TWO INPUTS $V_{IN\,1}$ & $V_{IN\,2}$ AND THE DIFFERENCE IS AMPLIFIED AND SENT TO OUTPUT.

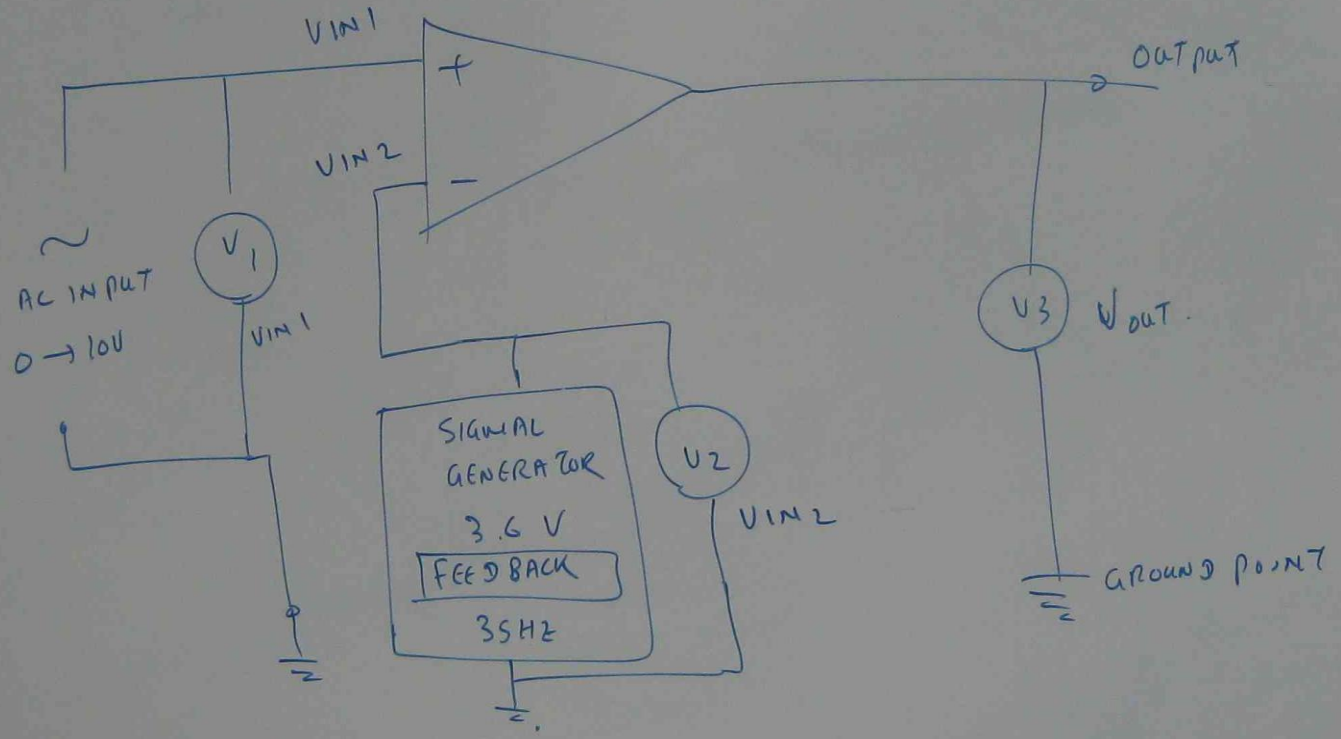
IF $V_{IN1} = 0$ & $V_{IN\,2}$ HAS REFERENCE VALUE, THE OUTPUT WILL BE MAXIMUM.

WHEN $V_{IN\,1}$ IS APPLIED AND INCREASED, THE OUTPUT WILL DECREASE AS V_{IN1} APPROACHES TO $V_{IN\,2}$.

IN THIS PRACTICAL, YOU ARE REQUIRED TO DETERMINE THE RELATION BETWEEN V_{IN1}, $V_{IN\,2}$ AND OUTPUT

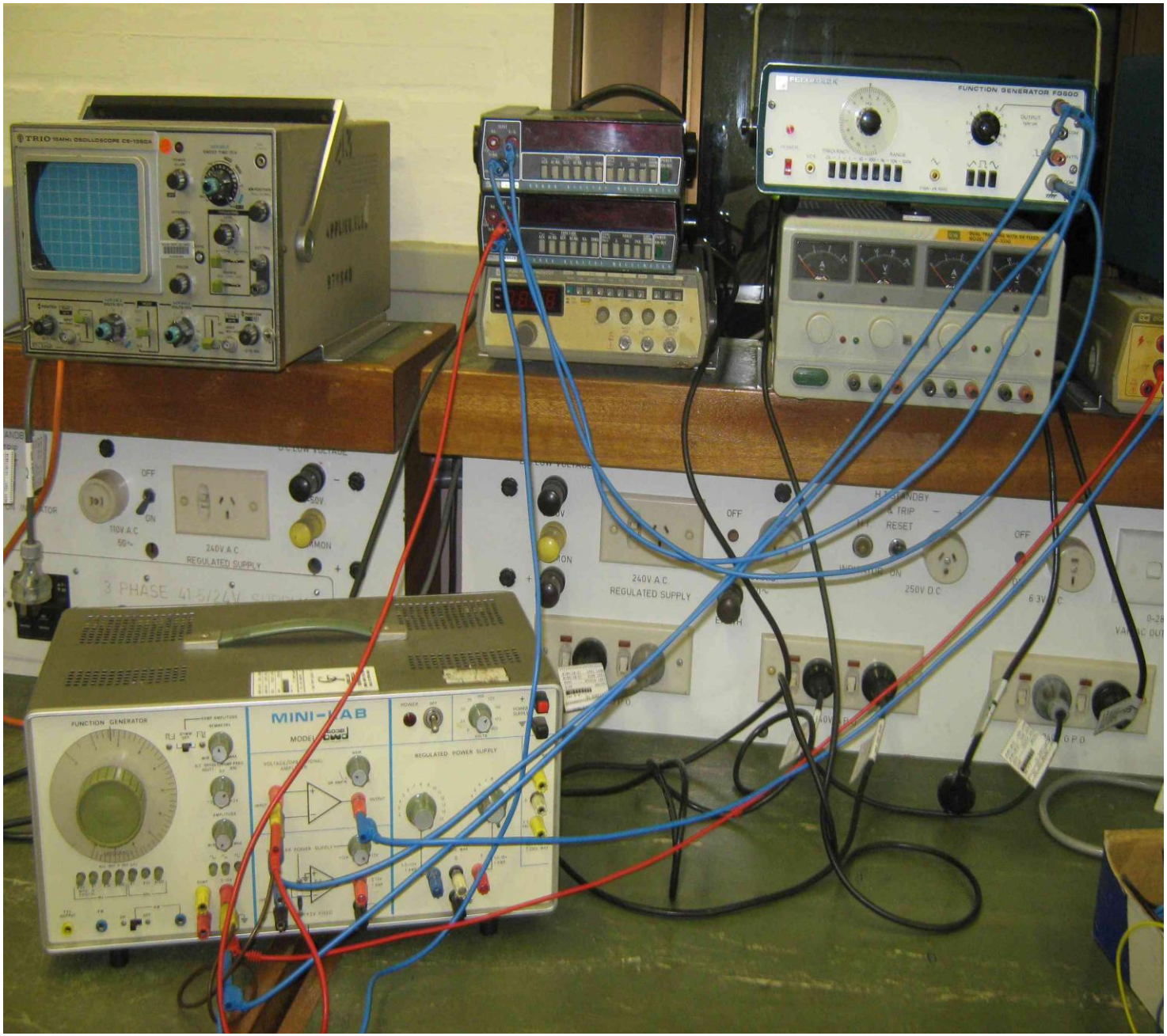
PROCEDURE

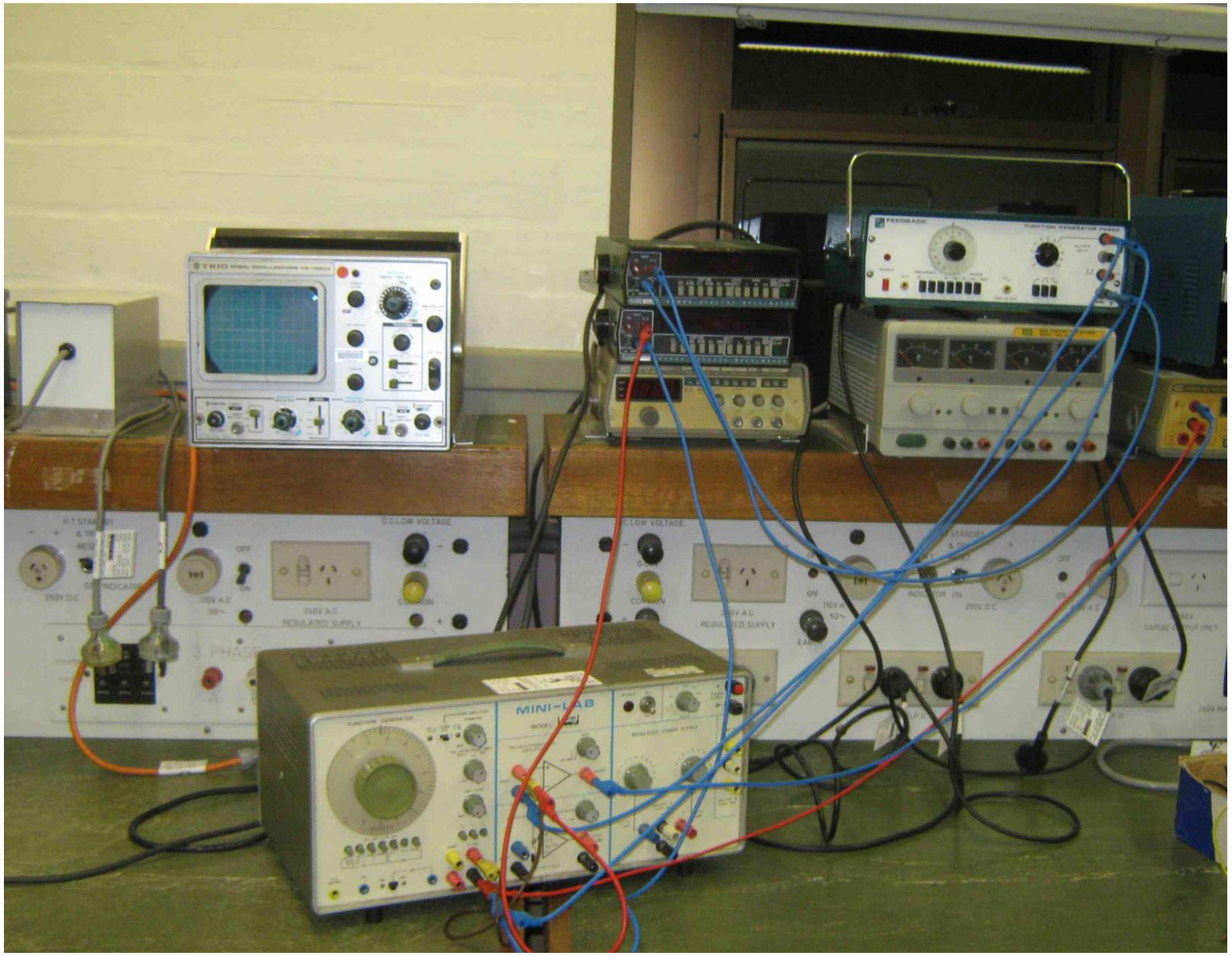
CONNECT THE GIVEN CIRCUIT

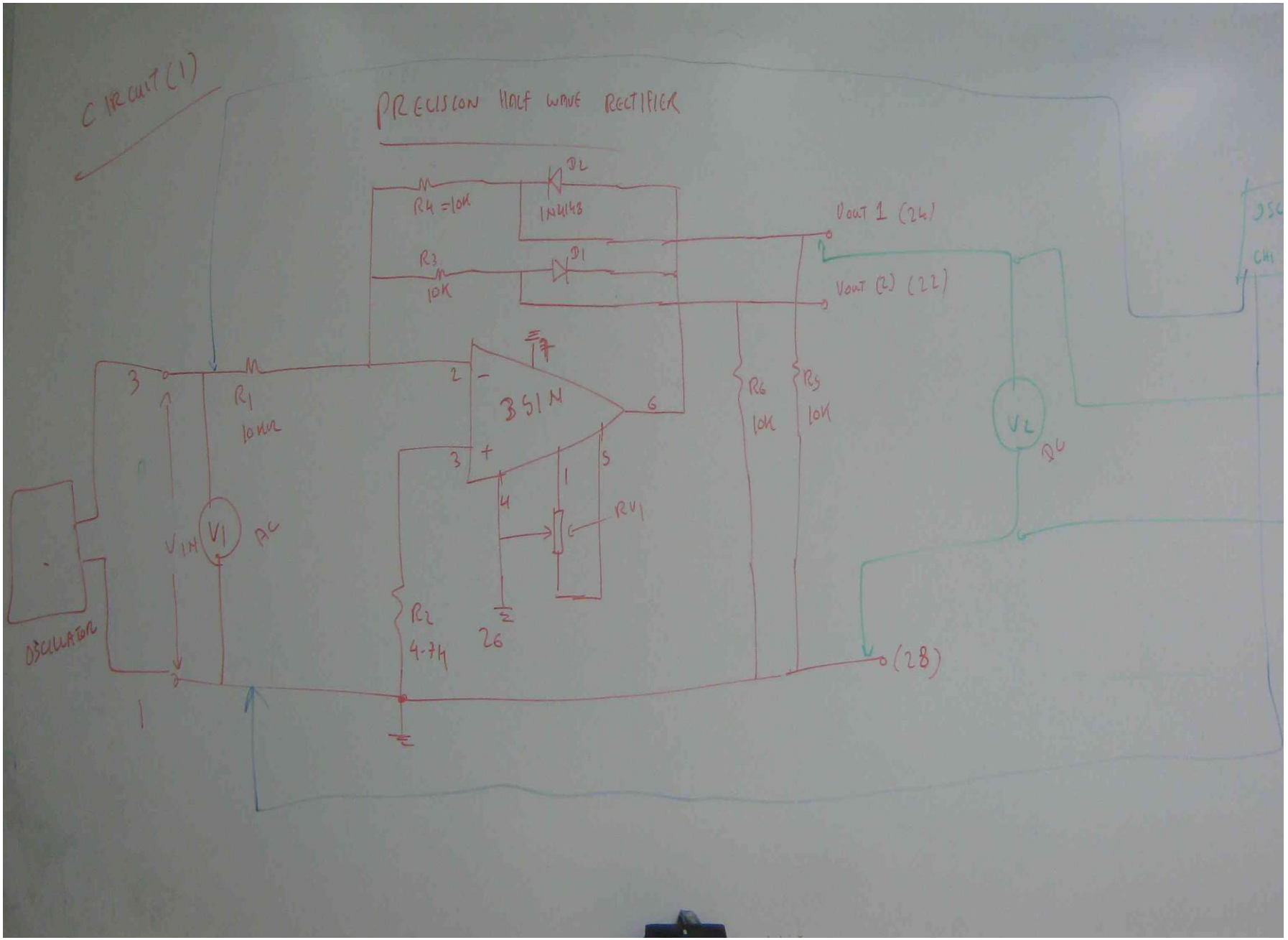


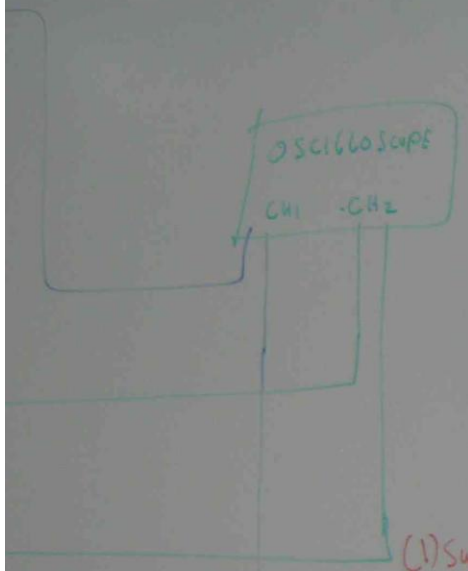
- GIVE 3.6V, 35 Hz TO V_{IN2} AS REFERENCE
- INCREASE V_{IN1} AND NOTE THE V_{OUT}

$V_{IN\,1}$ (V_1)	$V_{IN(2)}$ (V_2)	$V_2 - V_1$	V_{OUT}
0	3.6 V		
↓	CONSTANT		
MAXIMUM			

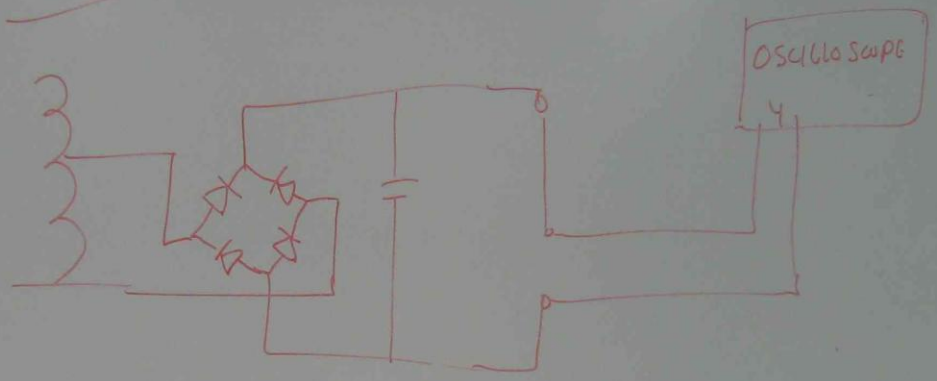

TRIO
FUNCTION GENERATOR FG500
240V A.C.
REGULATED SUPPLY
MINI-LAB
FUNCTION GENERATOR
REGULATED POWER SUPPLY


TRIO
FEEDBACK
FUNCTION GENERATOR
MINI-LAB
D.C LOW VOLTAGE
240V A.C
REGULATED SUPPLY
110V A.C
50~
250V D.C
3 PHASE


CIRCUIT (1)
PRECISION HALF WAVE RECTIFIER
R4 =10K
D2
IN4148
R3
10K
D1
VOUT 1 (24)
VOUT (2) (22)
R1
10KΩ
3
2
7
3
6
4
1
5
351N
RV1
R6
10K
R5
10K
V2
D6
VIN
V1
AC
OSCILLATOR
R2
4.7K
26
1
(28)
CH1



CIRCUIT (2)

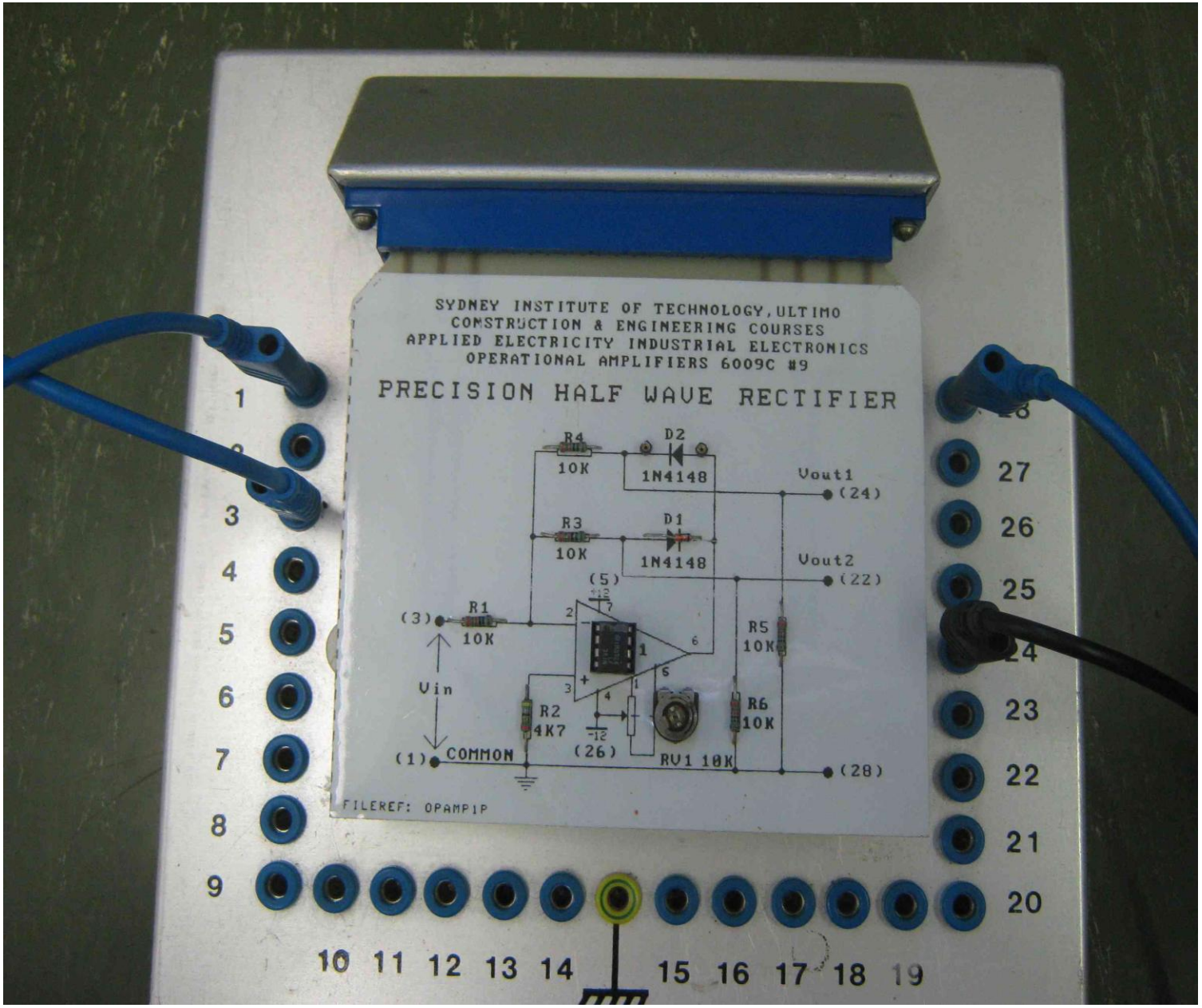


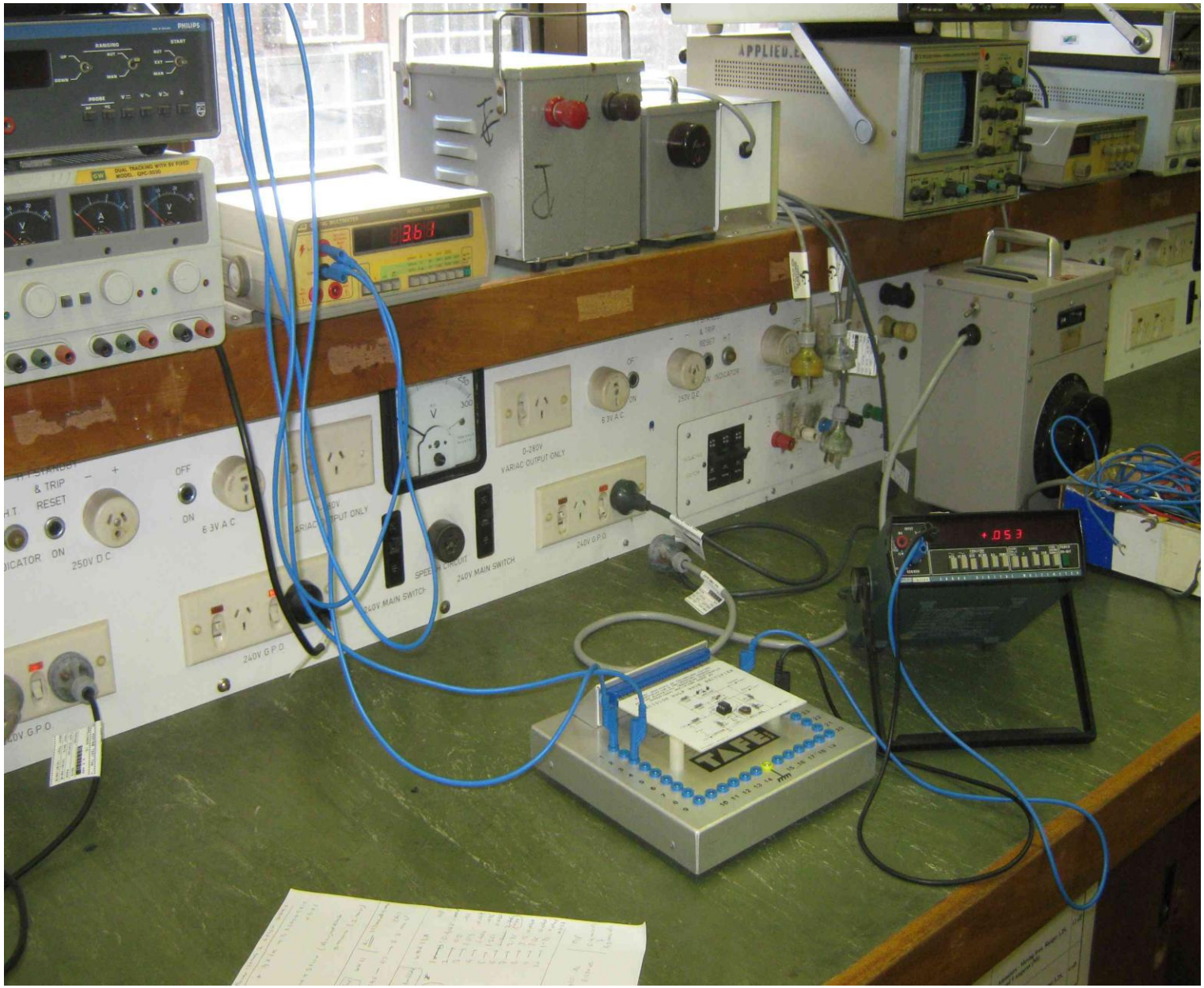
(1) SWITCH ON CIRCUIT (1) & (2)

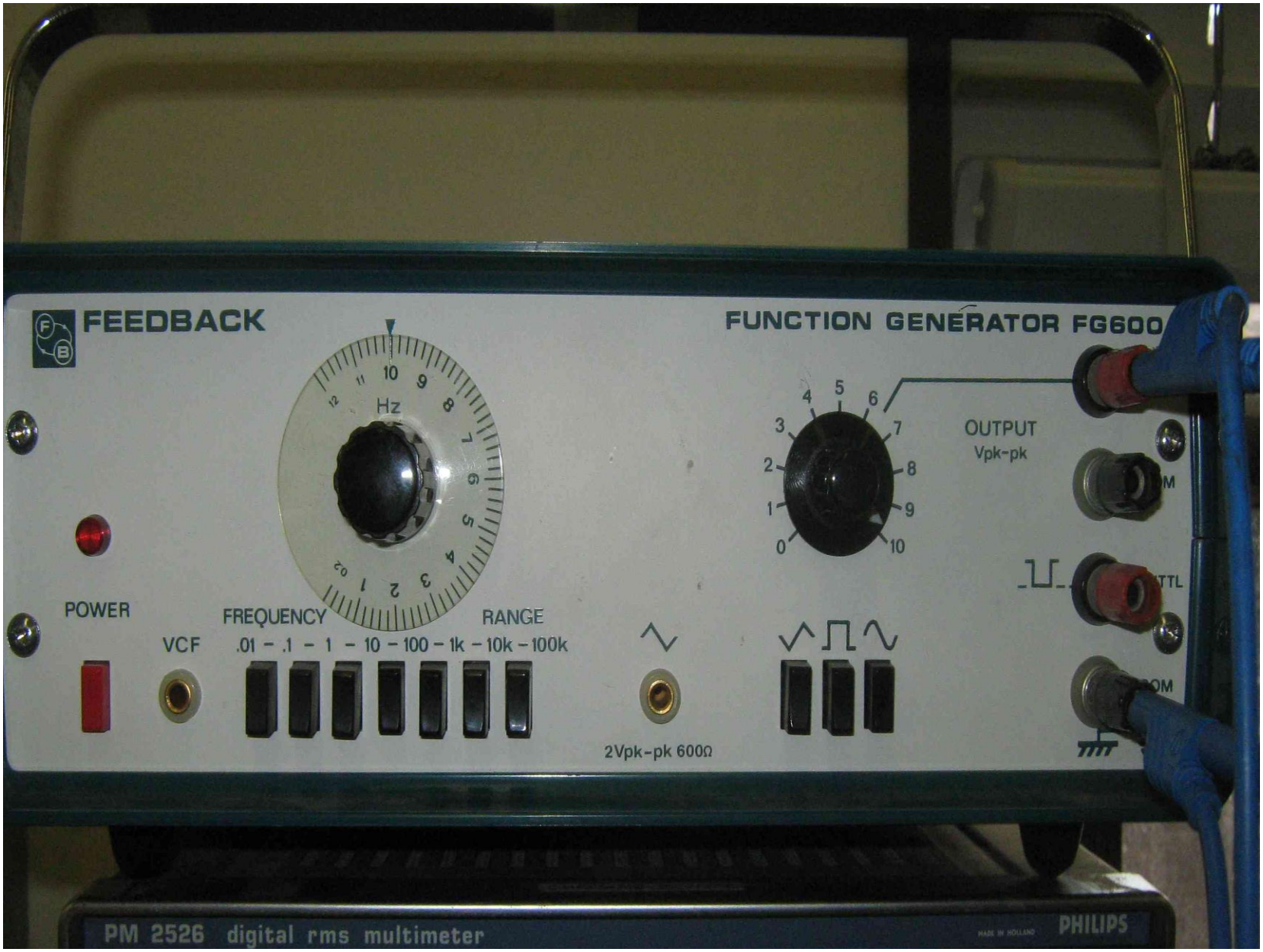
(2) INCREASE OSCILLATOR SETTINGS AND NOTE INPUT & OUTPUT VOLTAGES.

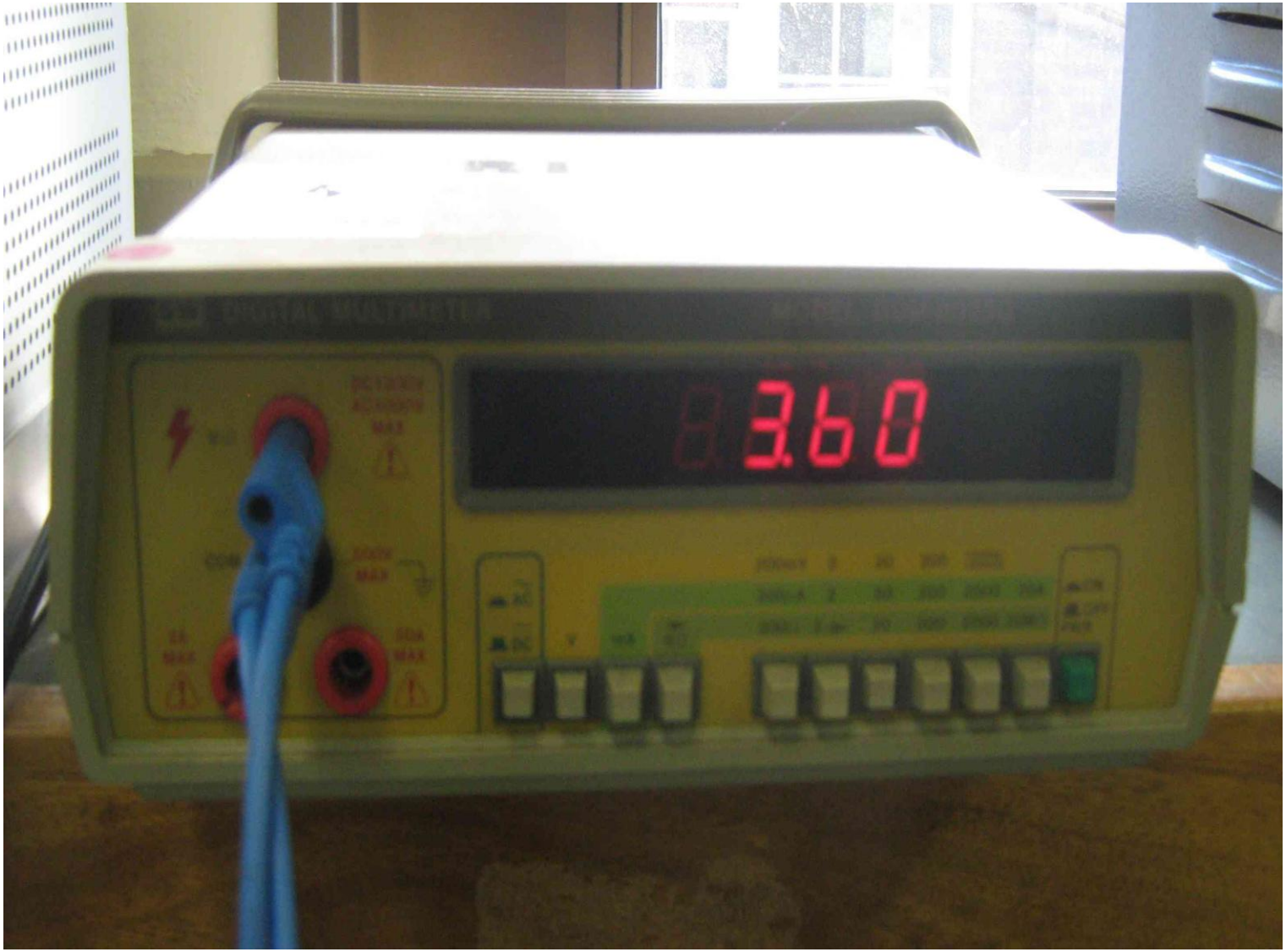
OSCILLATOR	V_1 (INPUT)	V_2 (OUTPUT)
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

(3) OBSERVE THE OUTPUT WAVE FORM (2C) AND COMPARE IT WITH CIRCUIT (2) OUTPUT WAVE ON OSCILLOSCOPE

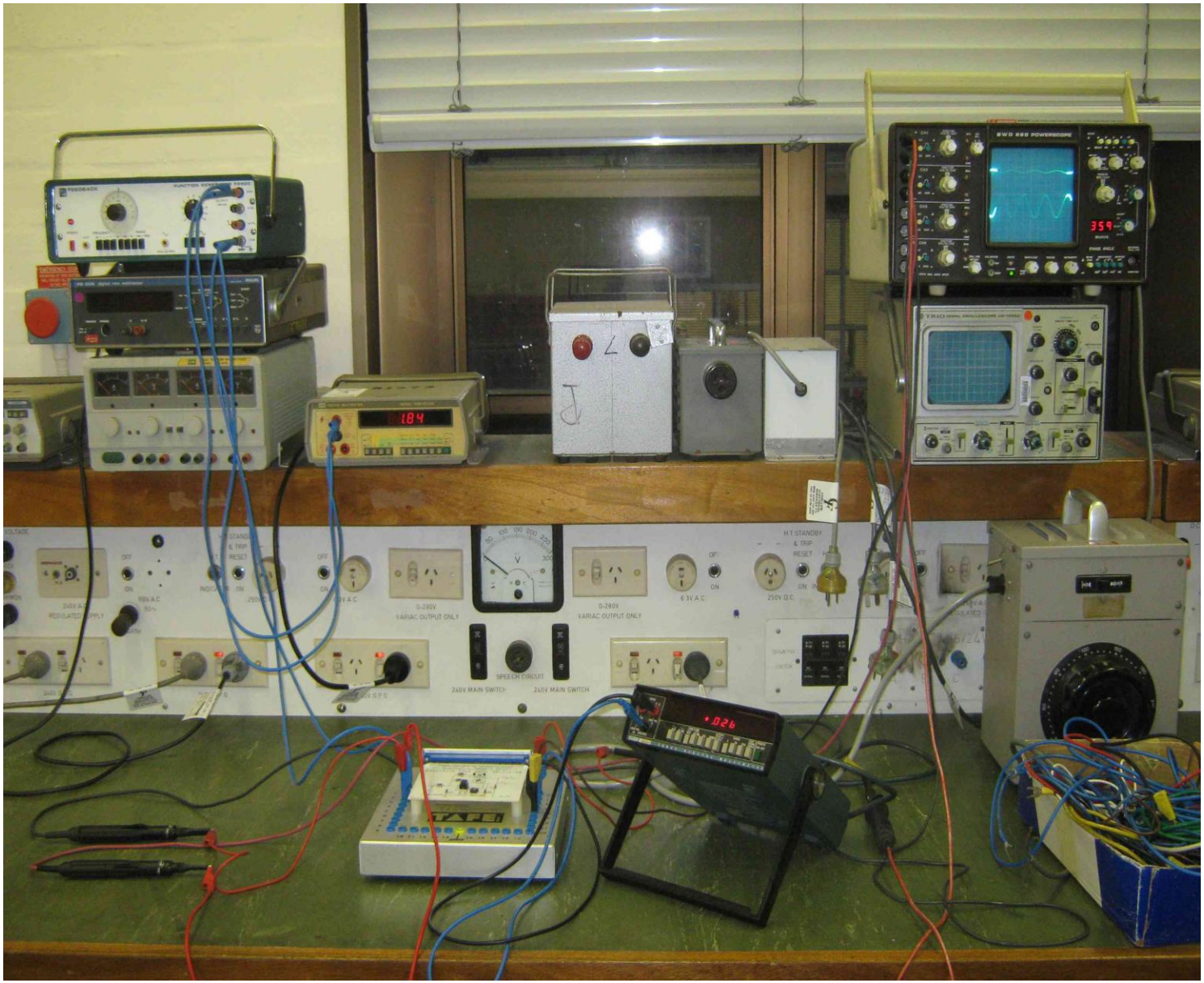

SYDNEY INSTITUTE OF TECHNOLOGY, ULTIMO
CONSTRUCTION & ENGINEERING COURSES
APPLIED ELECTRICITY INDUSTRIAL ELECTRONICS
OPERATIONAL AMPLIFIERS 6009C #9
PRECISION HALF WAVE RECTIFIER
R4
10K
D2
1N4148
Vout1
(24)
R3
10K
D1
1N4148
Vout2
(22)
(5)
(3)
R1
10K
R5
10K
Vin
R2
4K7
R6
10K
(1)
COMMON
(26)
RV1 10K
(28)
FILEREF: OPAMP1P
1
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

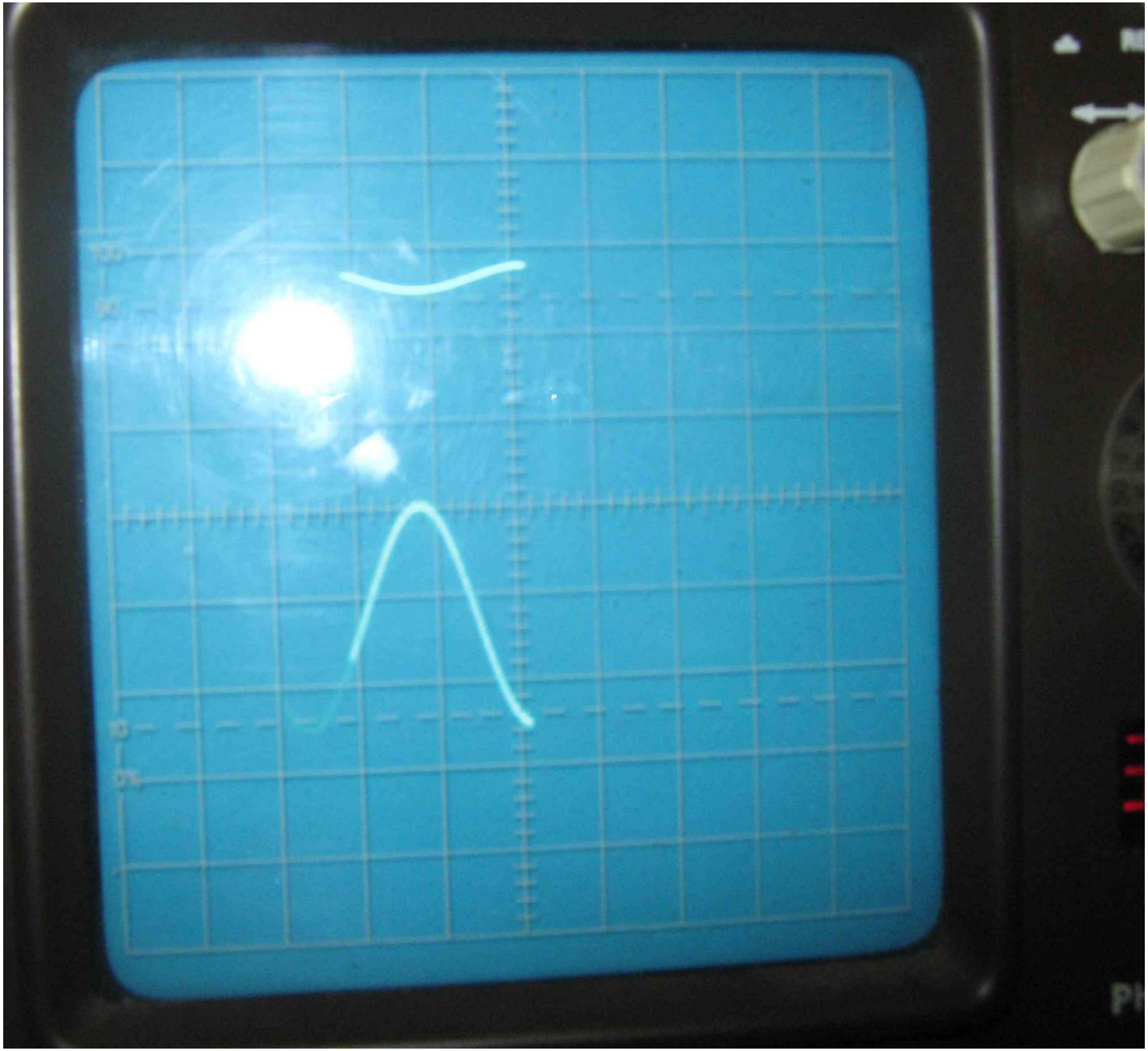

PHILIPS
APPLIED
& TRIP
RESET
OFF
ON
6.3V A.C
250V D.C
240V G.P.O
240V MAIN SWITCH
VARIAC OUTPUT ONLY
0-280V

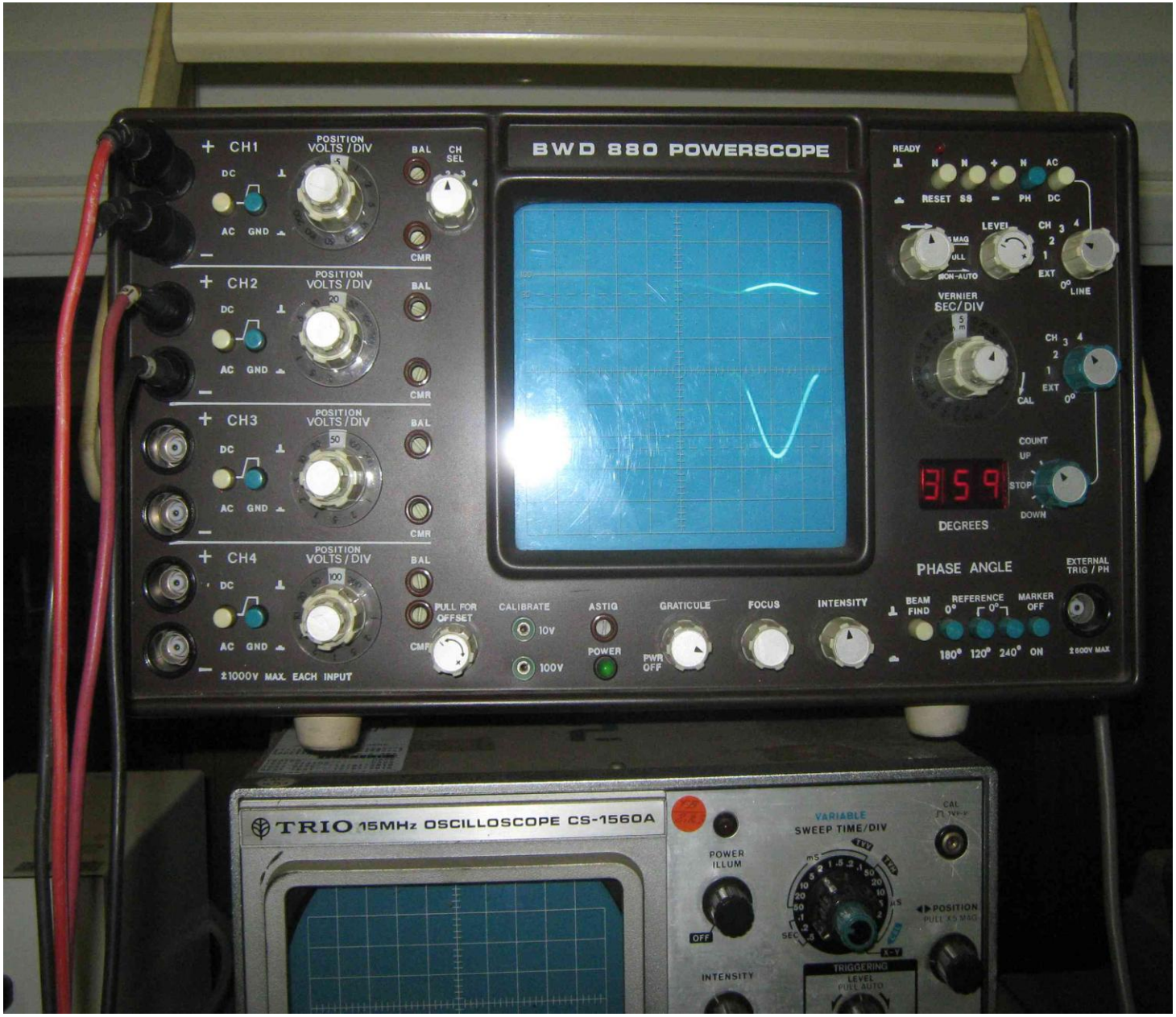

FEEDBACK
FUNCTION GENERATOR FG600
Hz
OUTPUT
Vpk-pk
POWER
VCF
FREQUENCY
RANGE
.01 - .1 - 1 - 10 - 100 - 1k - 10k - 100k
2Vpk-pk 600Ω
PM 2526 digital rms multimeter
PHILIPS


3.60


INPUT
MA
V-Ω
+.053
FUNCTION
ACV
AC MA
DCV
DC MA
KΩ
20MΩ
200MV
200μA
200Ω
RANGE
2
20
200
1200V
2000MA
2000KΩ
POWER
ON-OFF
FLUKE
84503
8000A DIGITAL MULTIMETER


359
184
0-280V
VARIAC OUTPUT ONLY
240V MAIN SWITCH
SPEECH CIRCUIT




BWD 880 POWERSCOPE
CH1
CH2
CH3
CH4
POSITION
VOLTS / DIV
DC
AC
GND
BAL
CMR
CH SEL
±1000V MAX. EACH INPUT
READY
RESET
LEVEL
VERNIER
SEC/DIV
CAL
EXT
LINE
COUNT
UP
STOP
DOWN
DEGREES
PHASE ANGLE
EXTERNAL TRIG / PH
BEAM FIND
REFERENCE
MARKER OFF
180° 120° 240° ON
±600V MAX
PULL FOR OFFSET
CALIBRATE
10V
100V
ASTIG
POWER
GRATICULE
PWR OFF
FOCUS
INTENSITY
TRIO 15MHz OSCILLOSCOPE CS-1560A
POWER ILLUM
OFF
VARIABLE
SWEEP TIME/DIV
POSITION
INTENSITY
TRIGGERING
LEVEL
PULL AUTO


TRIO
2

$$N = \frac{120f}{P}$$

N = MOTOR SPEED

f = FREQUENCY

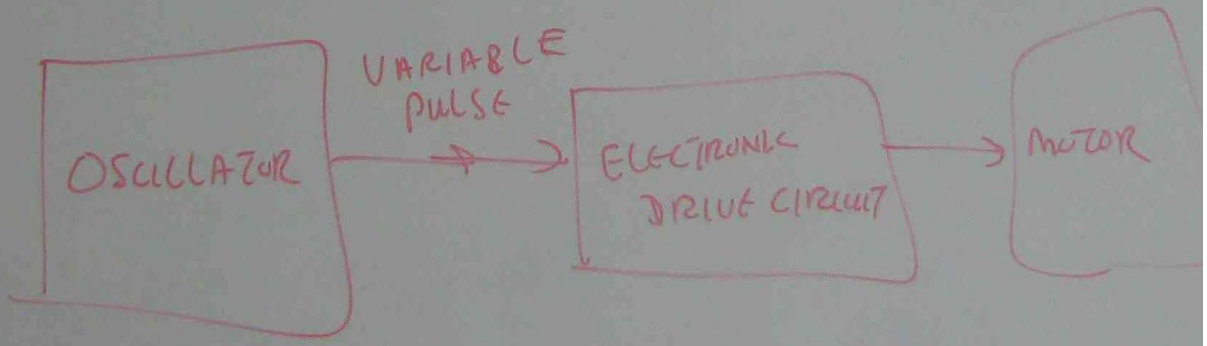
P = NO. OF POLES.

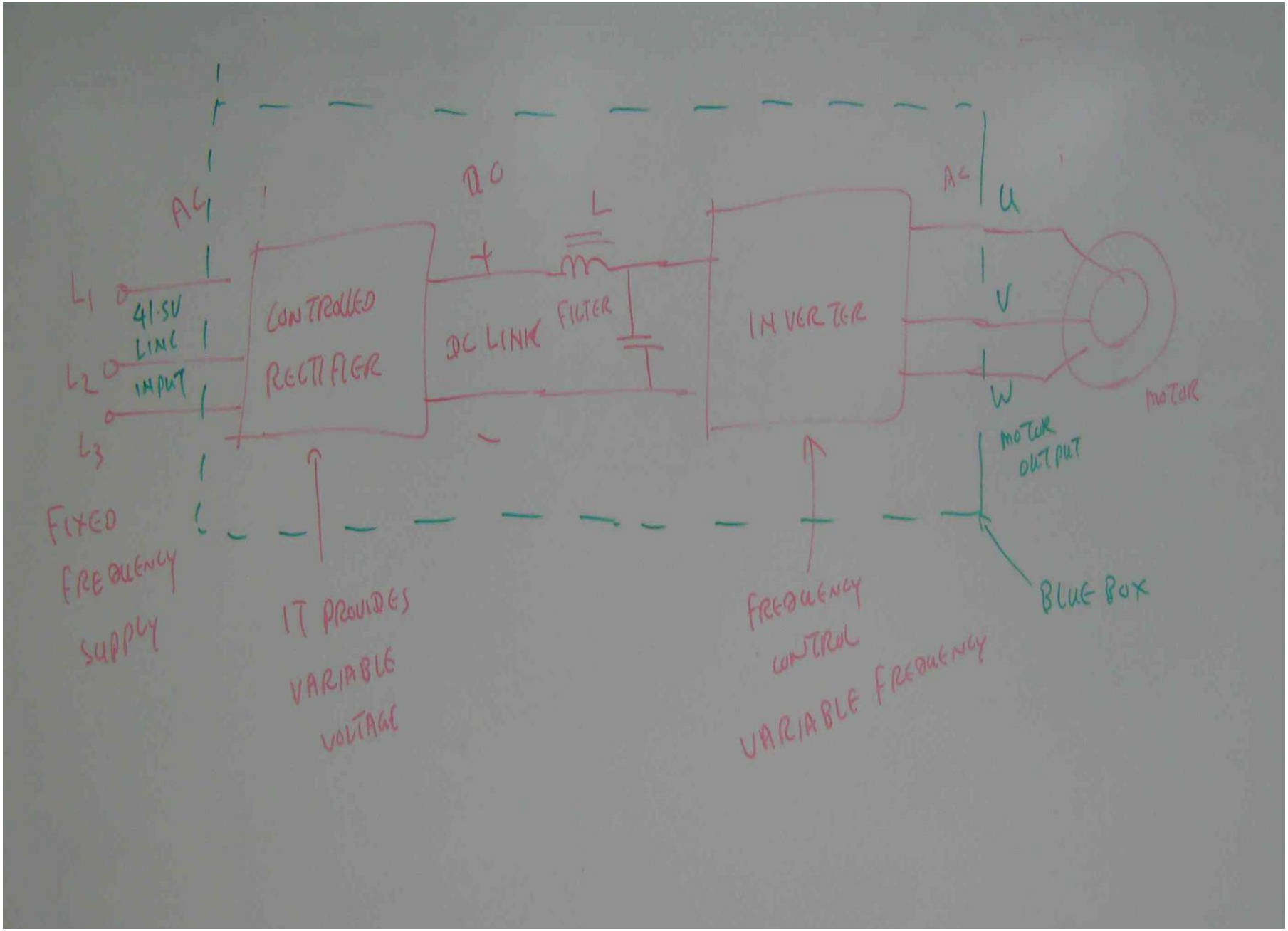
If $f = 50Hz$

$P = 4$ poles

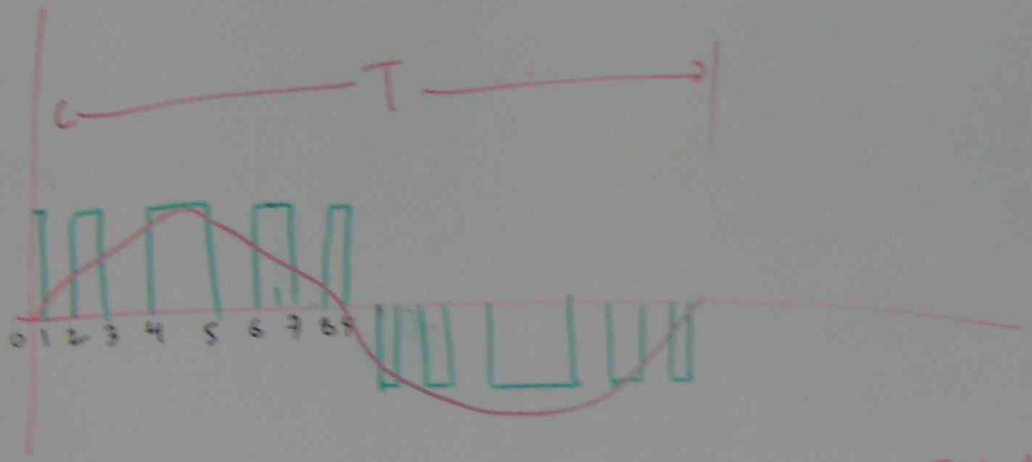
$$N = \frac{120 \times 50}{4}$$

$= 1500$ RPM




AC
DC
AC
L1
L2
L3
415V
LINE
INPUT
CONTROLLED
RECTIFIER
DC LINK
FILTER
L
INVERTER
U
V
W
MOTOR
MOTOR
OUTPUT
FIXED
FREQUENCY
SUPPLY
IT PROVIDES
VARIABLE
VOLTAGE
FREQUENCY
CONTROL
VARIABLE FREQUENCY
BLUE BOX

Pulse width modulation system → frequency / output voltage control



switching circuits. Their switching rates are different

0 → 1 = 1 μs

2 → 3 = 2 μs

4 → 5 = 3 μs

6 → 7 = 2 μs

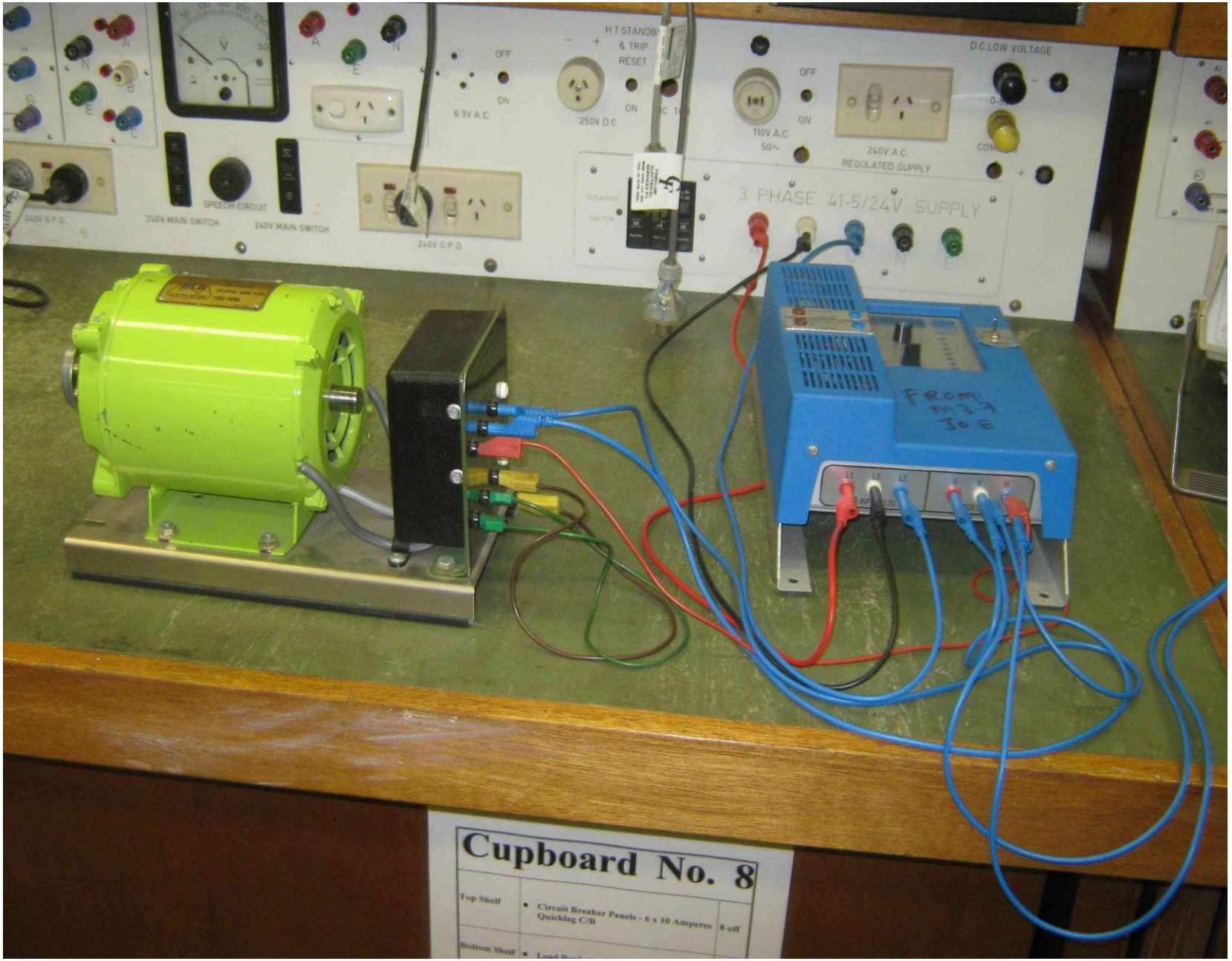
8 → 9 = 1 μs

By switching rate, sinusoidal output voltage is created.

T depends on total switching times of electronic devices

$$f = \frac{1}{T}$$

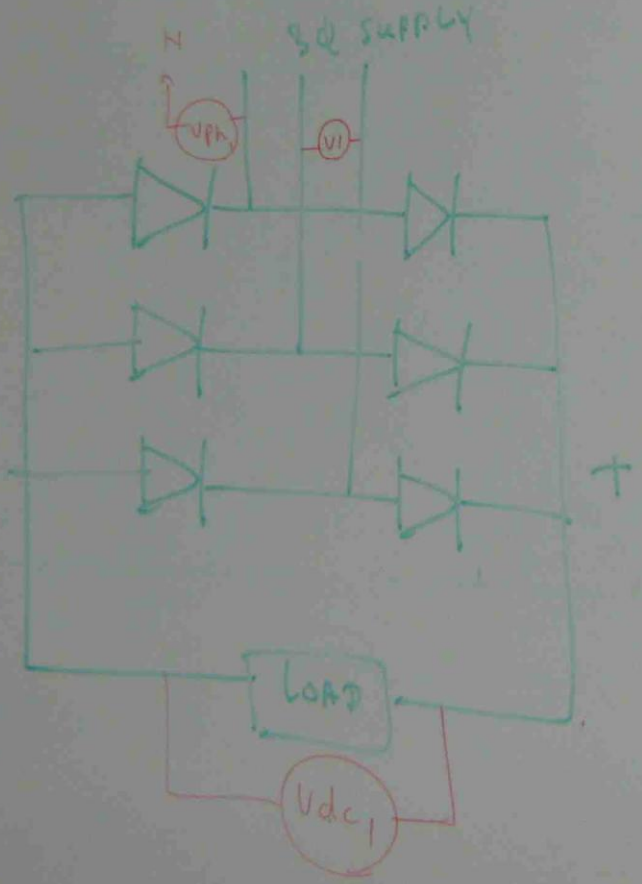
By adjusting the switching rates of electronic devices, the output frequency can be changed.


H.T STANDBY & TRIP RESET
OFF
ON
6.3V A.C
250V D.C
110V A.C 50~
240V A.C REGULATED SUPPLY
D.C LOW VOLTAGE
3 PHASE 41-5/24V SUPPLY
240V MAIN SWITCH
SPEECH CIRCUIT
240V G.P.O
From M33 Joe
Cupboard No. 8
Top Shelf
Circuit Breaker Panels - 6 x 10 Amperes Quickdag C/B
8 off
Bottom Shelf


OFF
CAL
STROBOTAC
TYPE 1531-A
OUTPUT
TRIGGER
INPUT

3ϕ RECTIFIER

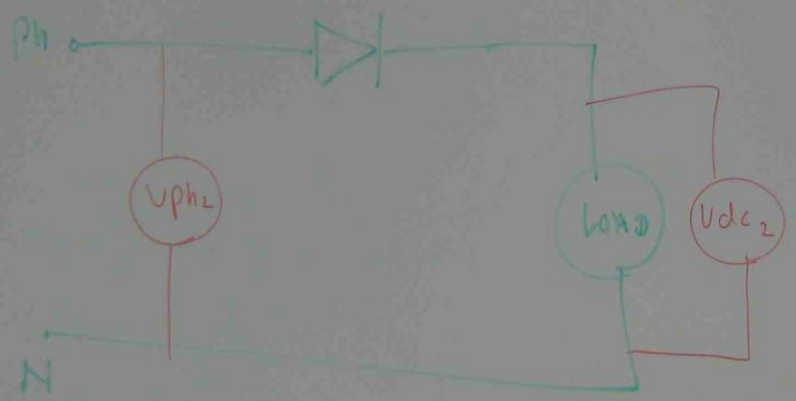
CONNECT THE GIVEN CIRCUIT



MEASURE V_l, V_{ph}, V_{dc}

II

CONNECT THE GIVEN CIRCUIT



MEASURE U_{ph_2}, V_{dc_2}

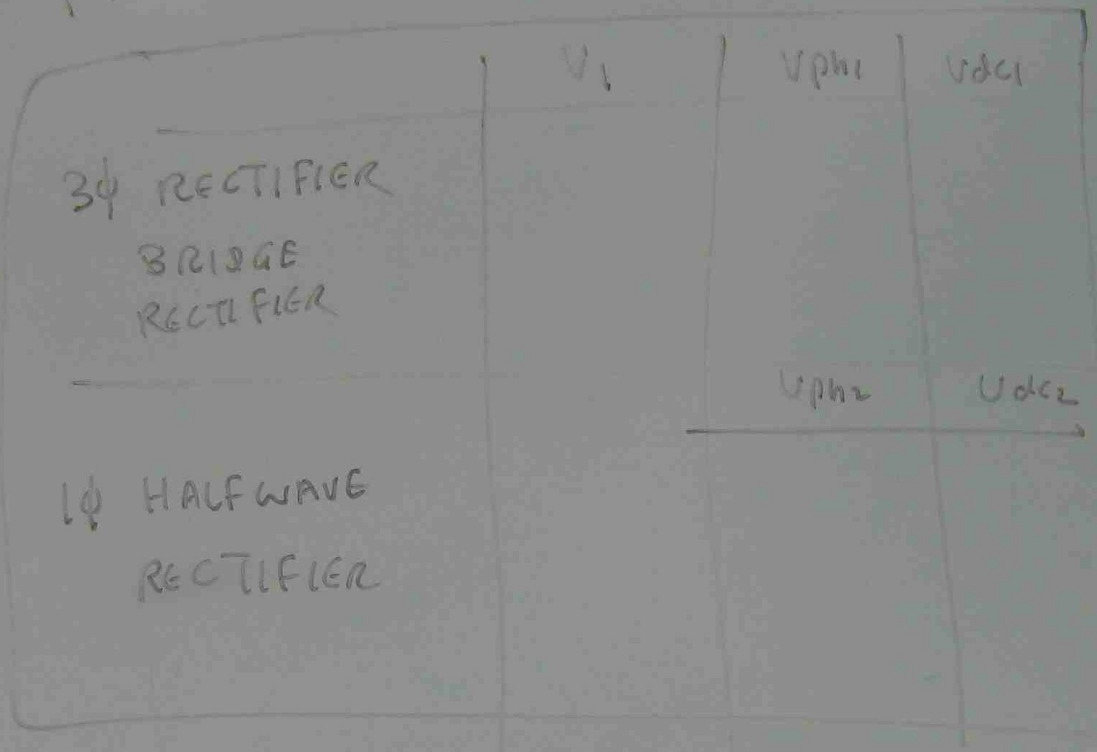
FILL IN TABLE

3φ RECTIFIER
BRIDGE
RECTIFIER

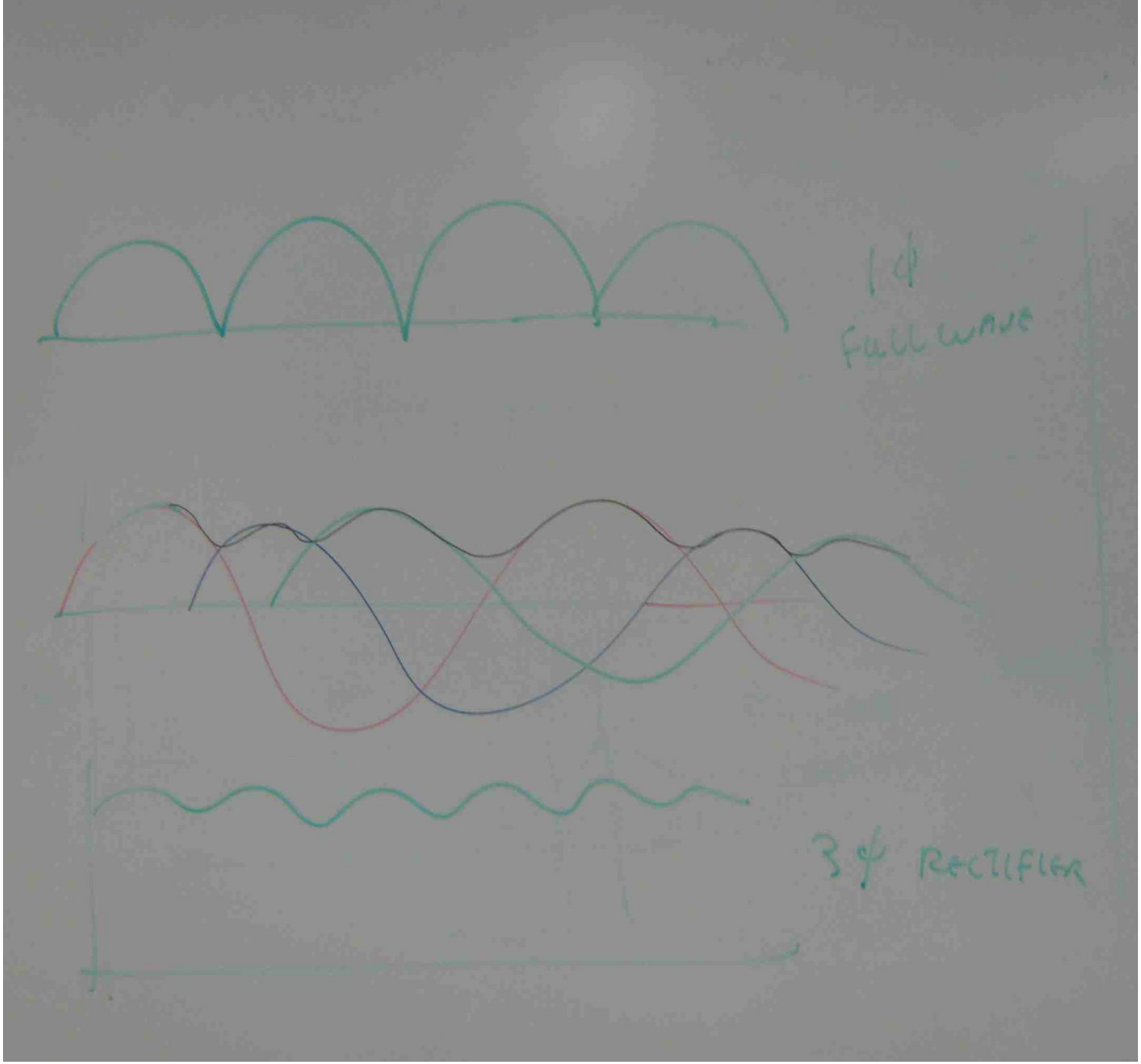
1φ HALFWAVE
RECTIFIER

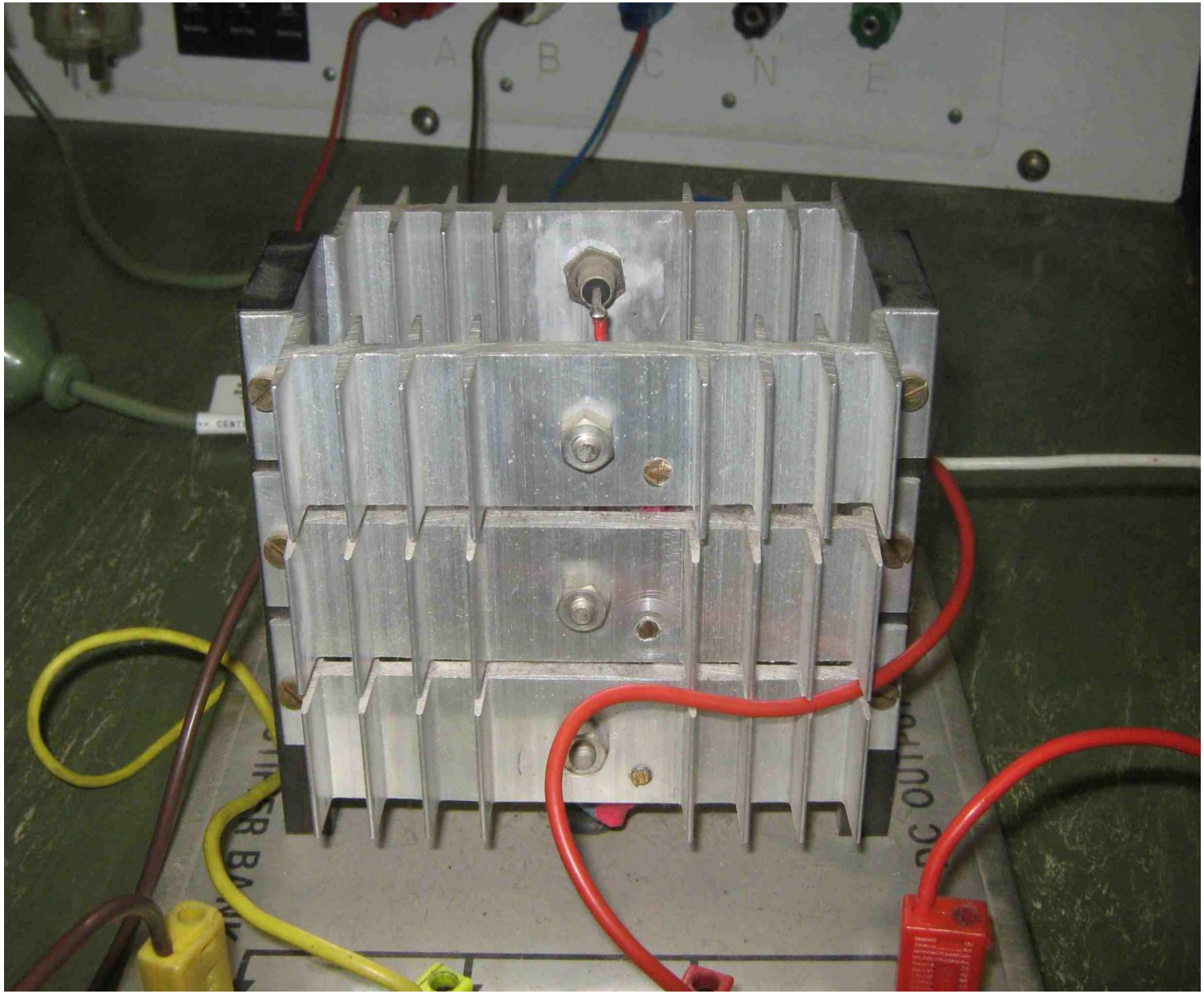
WHICH SY
DC VOLTAG

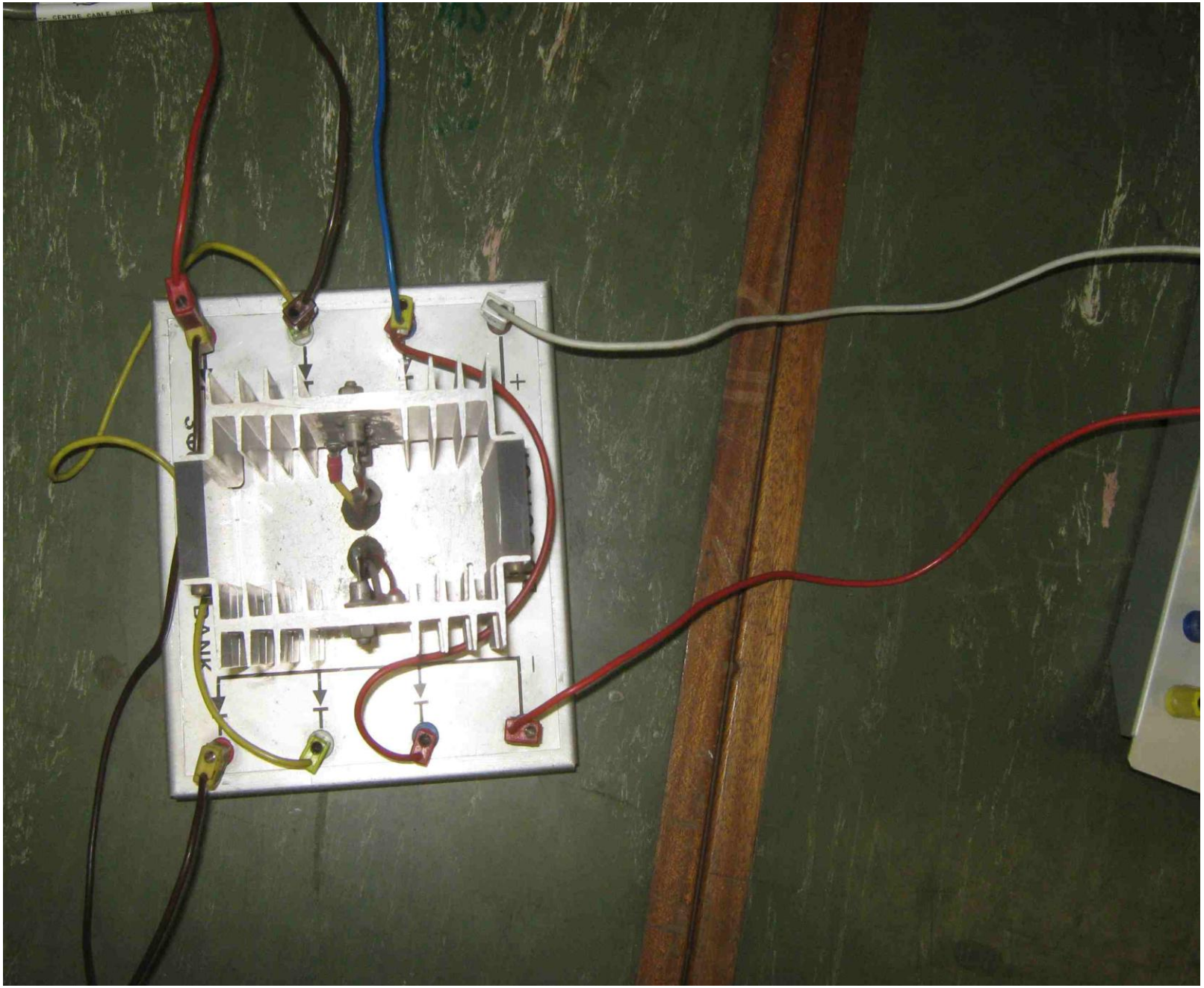
FILL IN TABLE

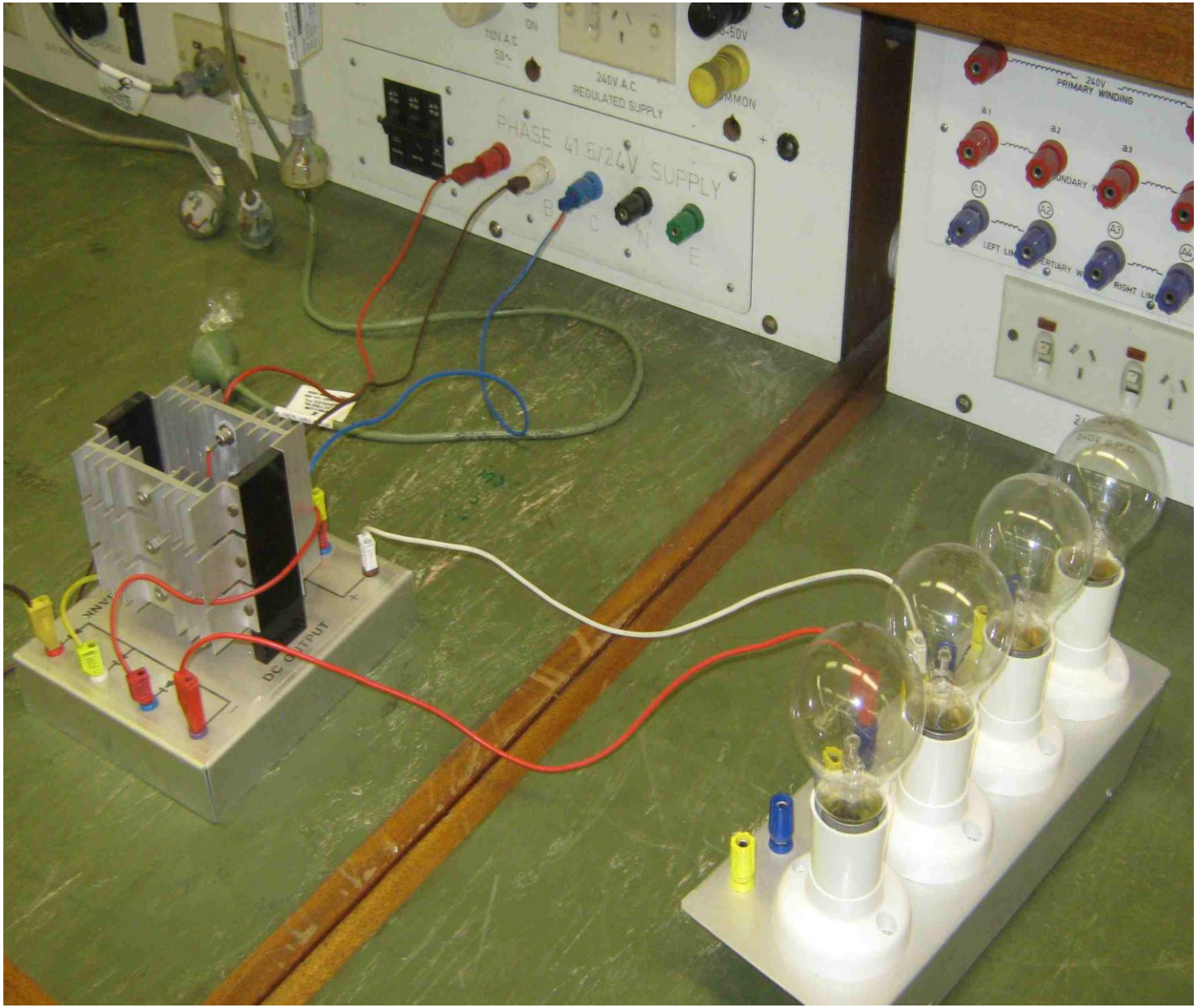


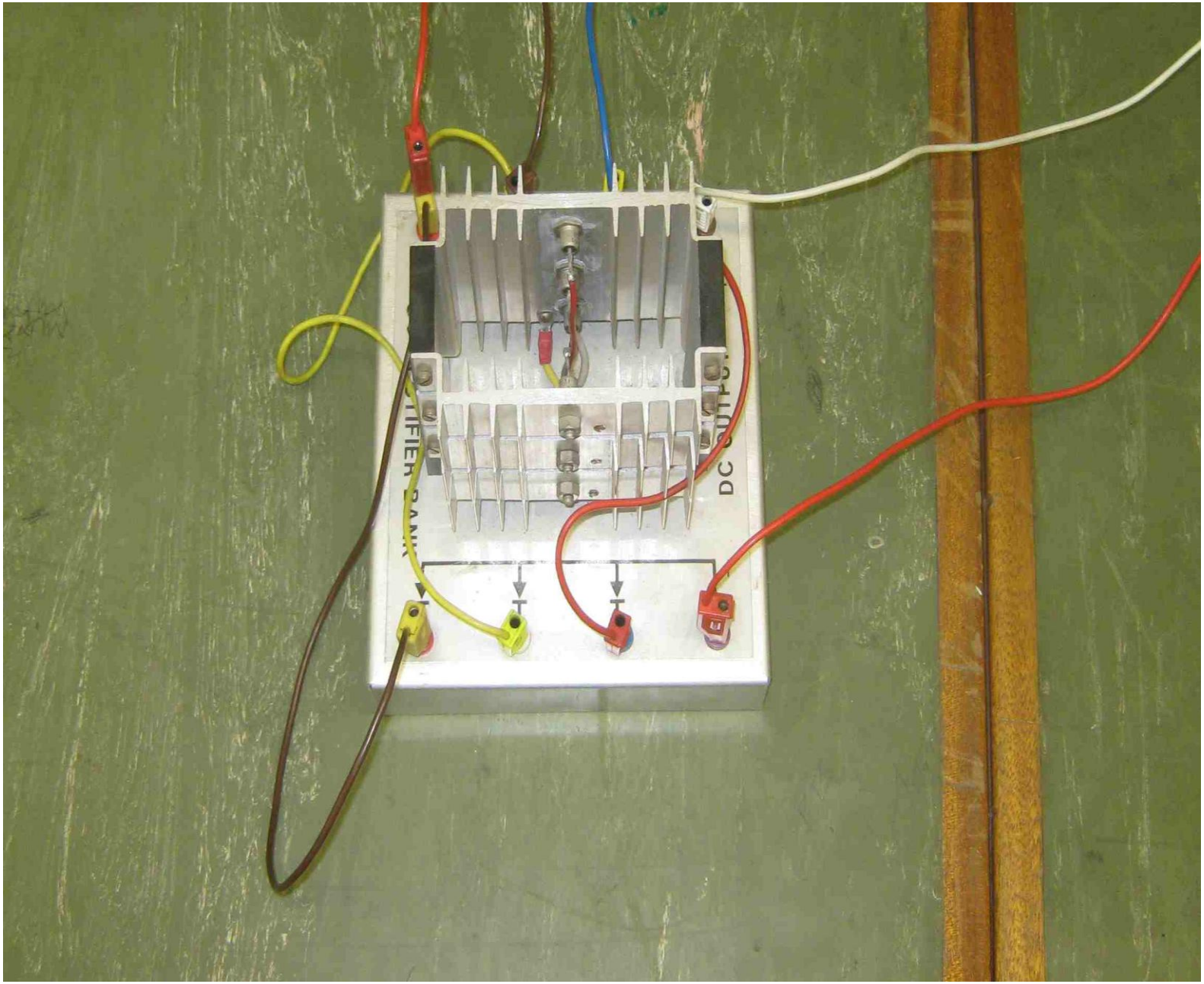
WHICH SYSTEM CAN GIVE HIGHER DC VOLTAGE, WHY?

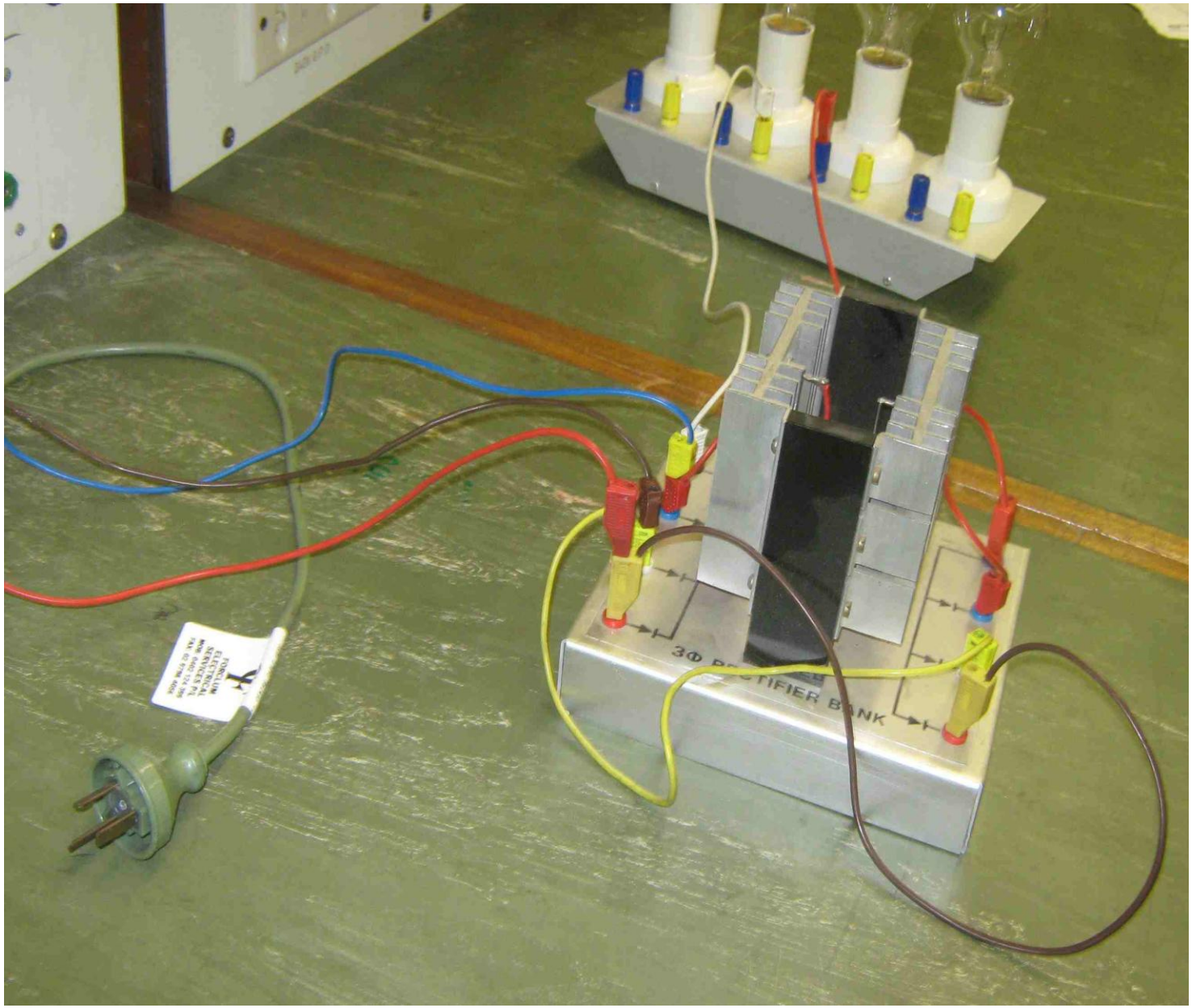

1 ϕ
Full wave
3 ϕ RECTIFIER

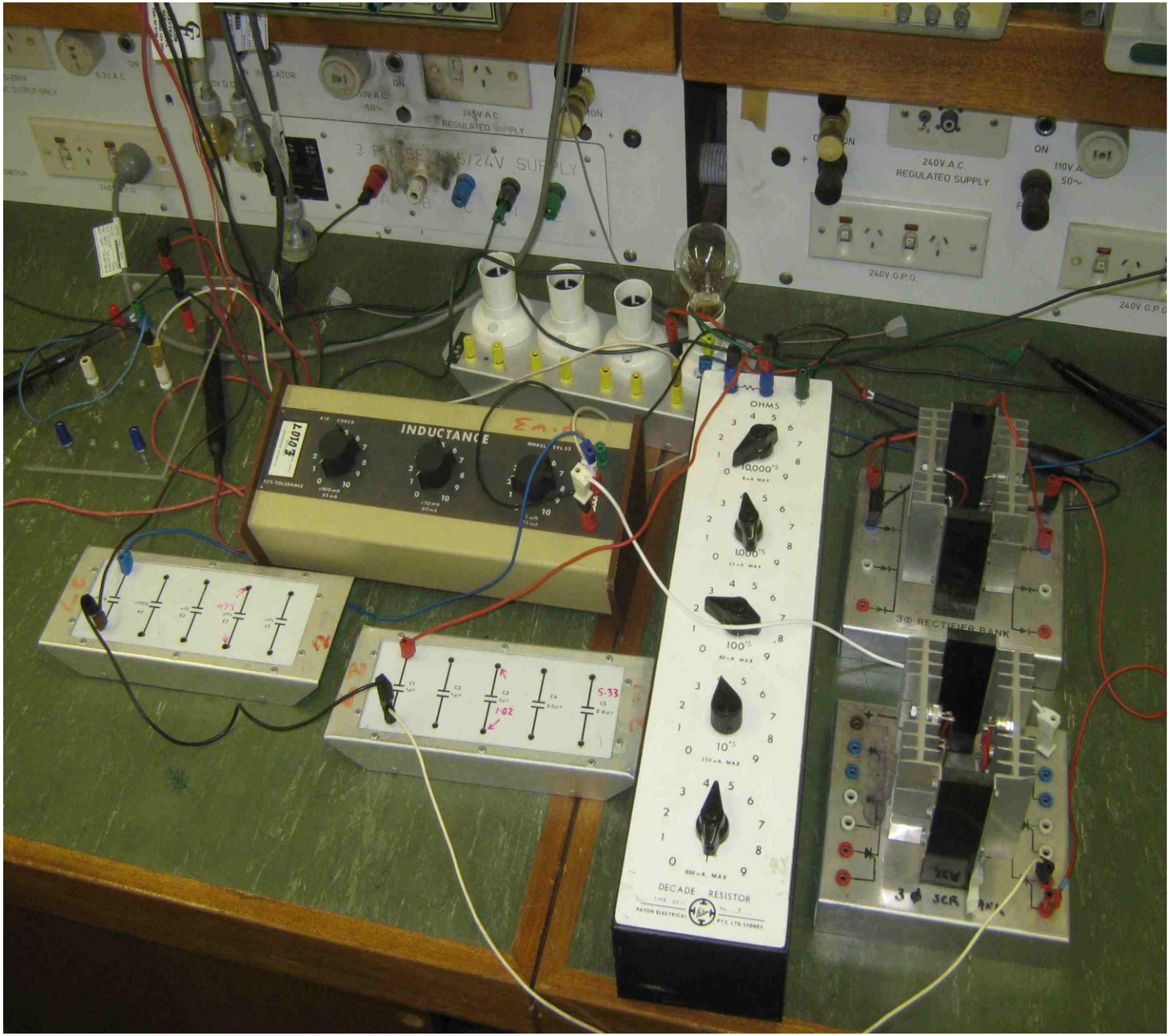

A
B
C
N
E

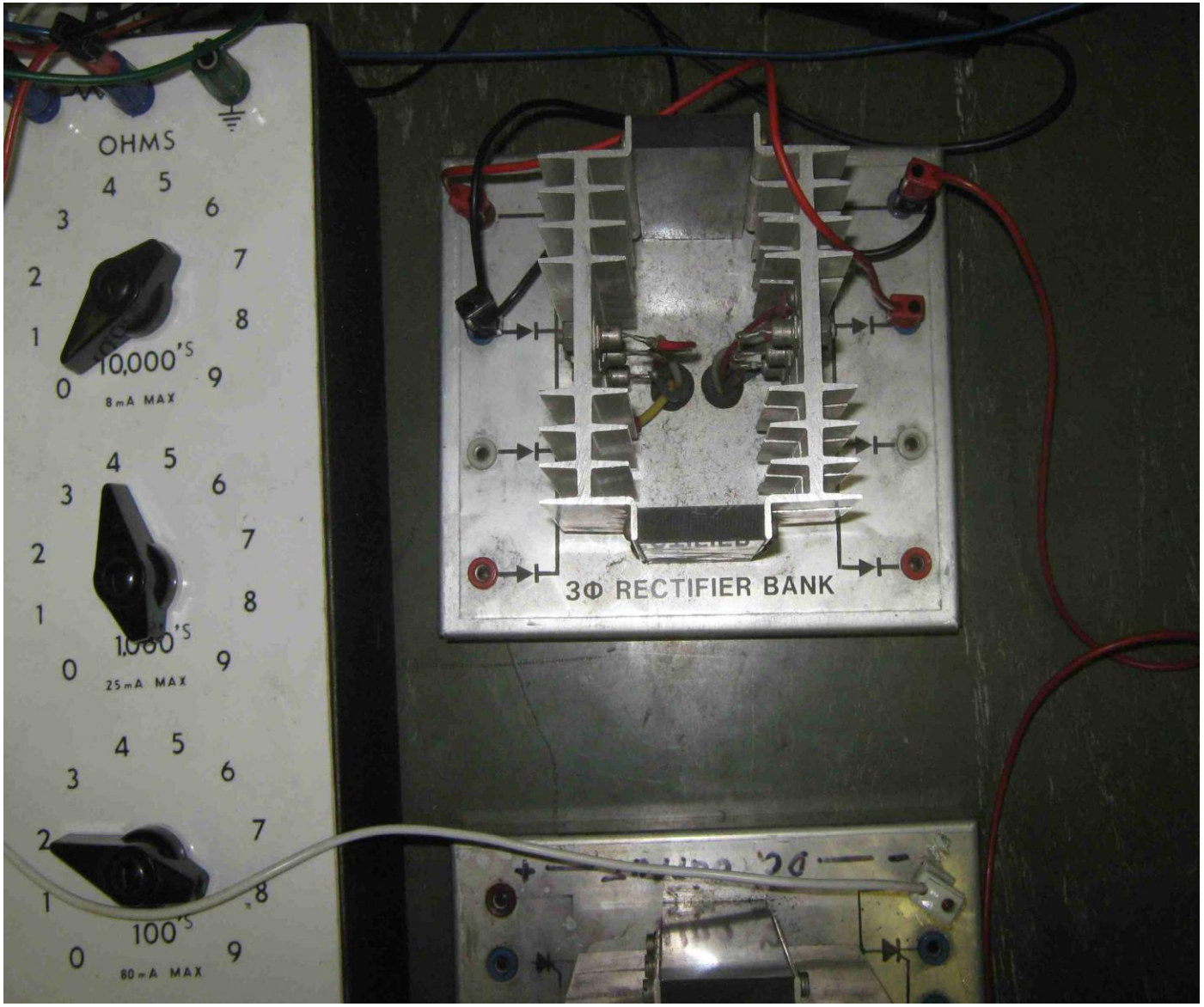

CENTRE CABLE HERE
+
BLANK

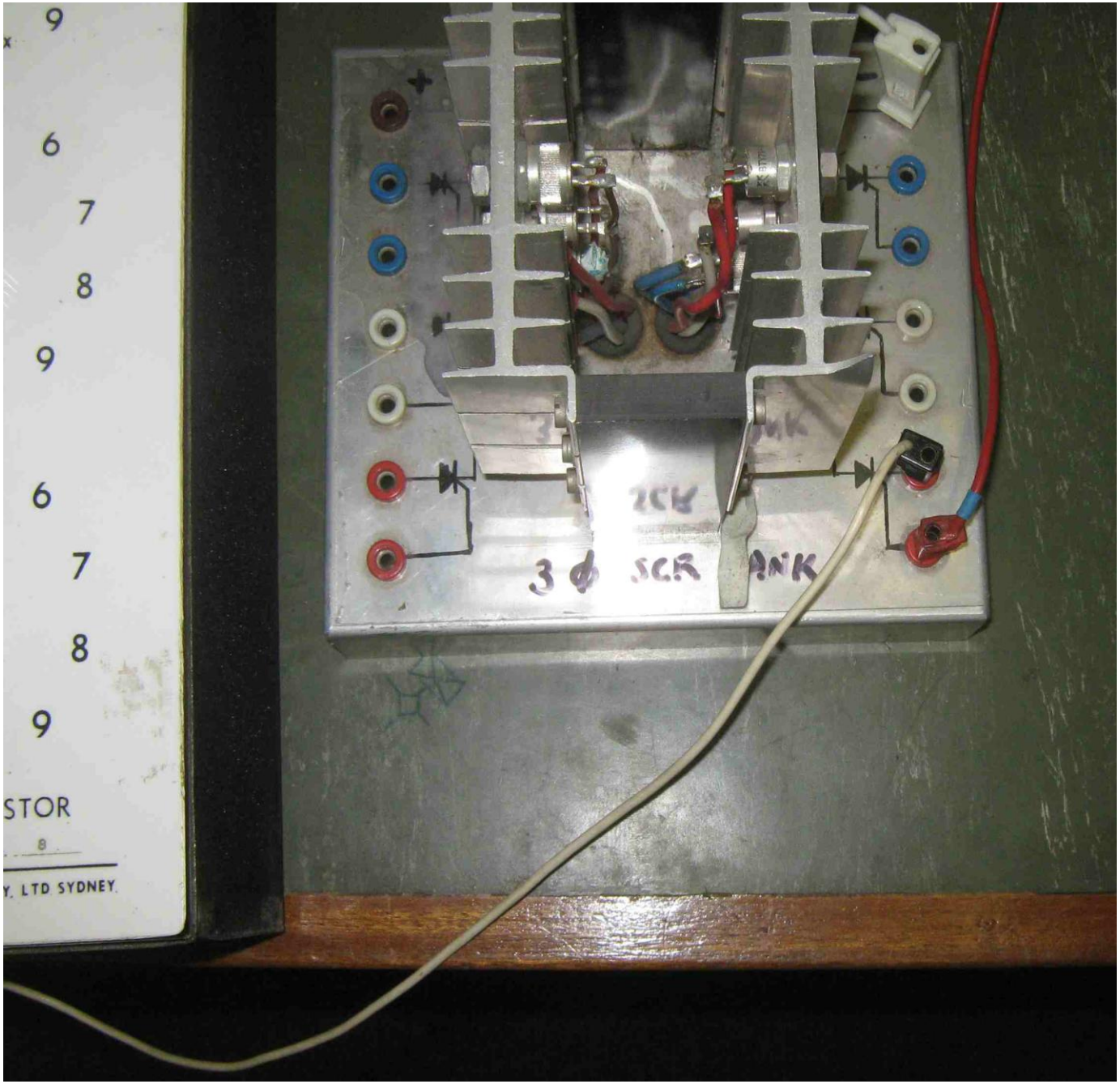

110V A.C.
50~
ON
240V A.C.
REGULATED SUPPLY
COMMON
PHASE 415/240V SUPPLY
240V
PRIMARY WINDING
DC OUTPUT

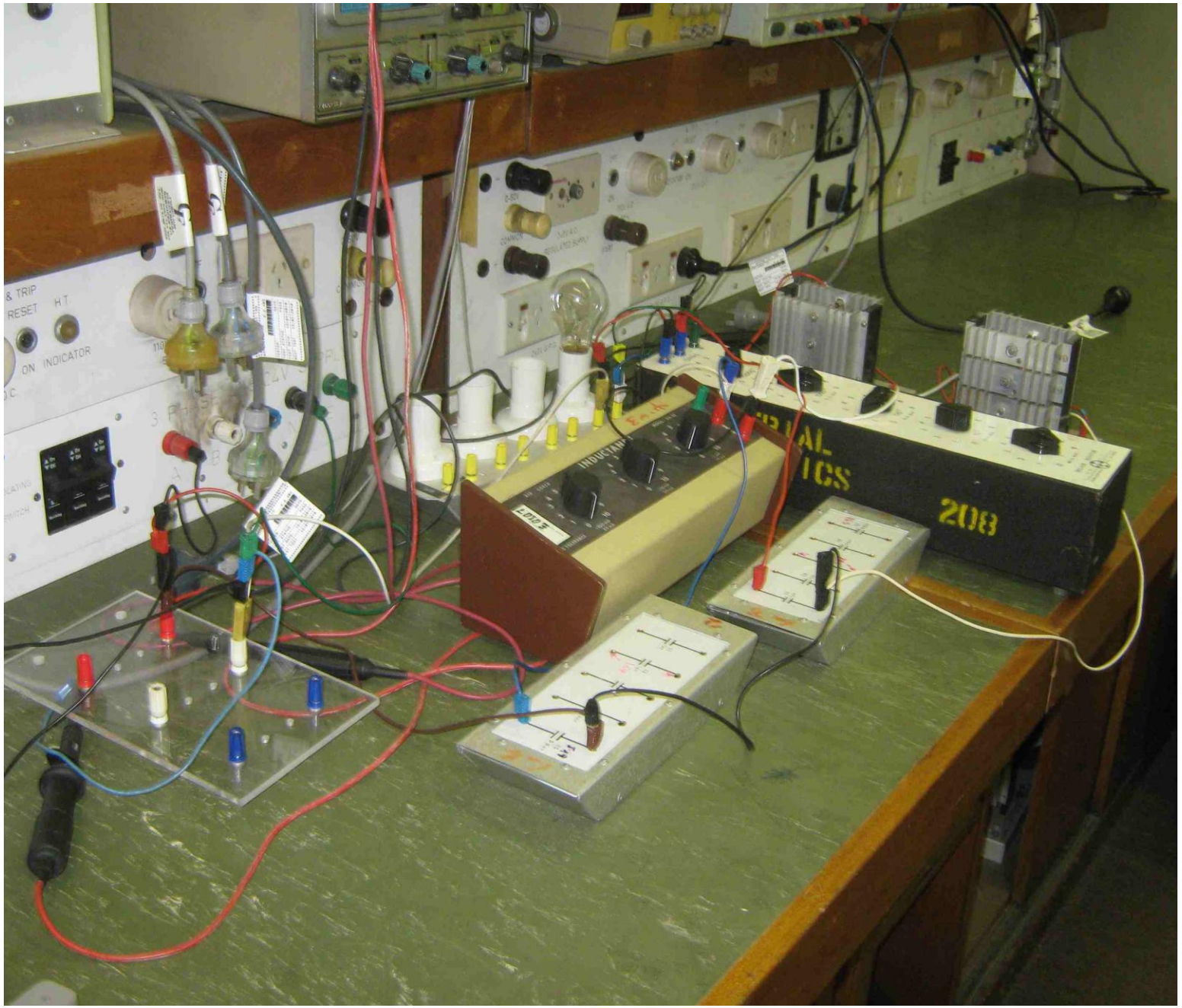


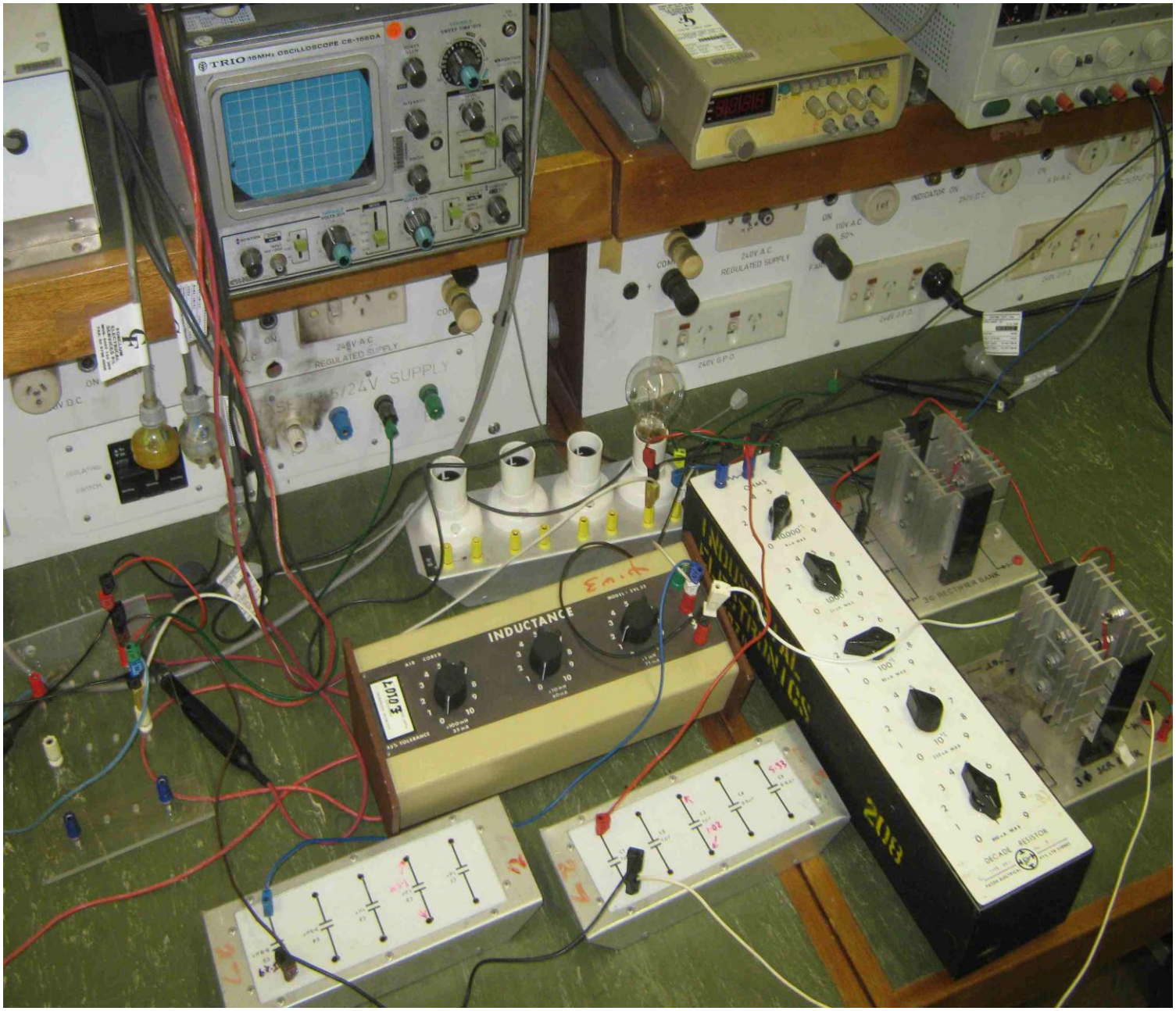

3Φ


240V A.C
REGULATED SUPPLY
240V G.P.O.
INDUCTANCE
OHMS
DECADE RESISTOR
3Φ RECTIFIER BANK
3Φ SCR

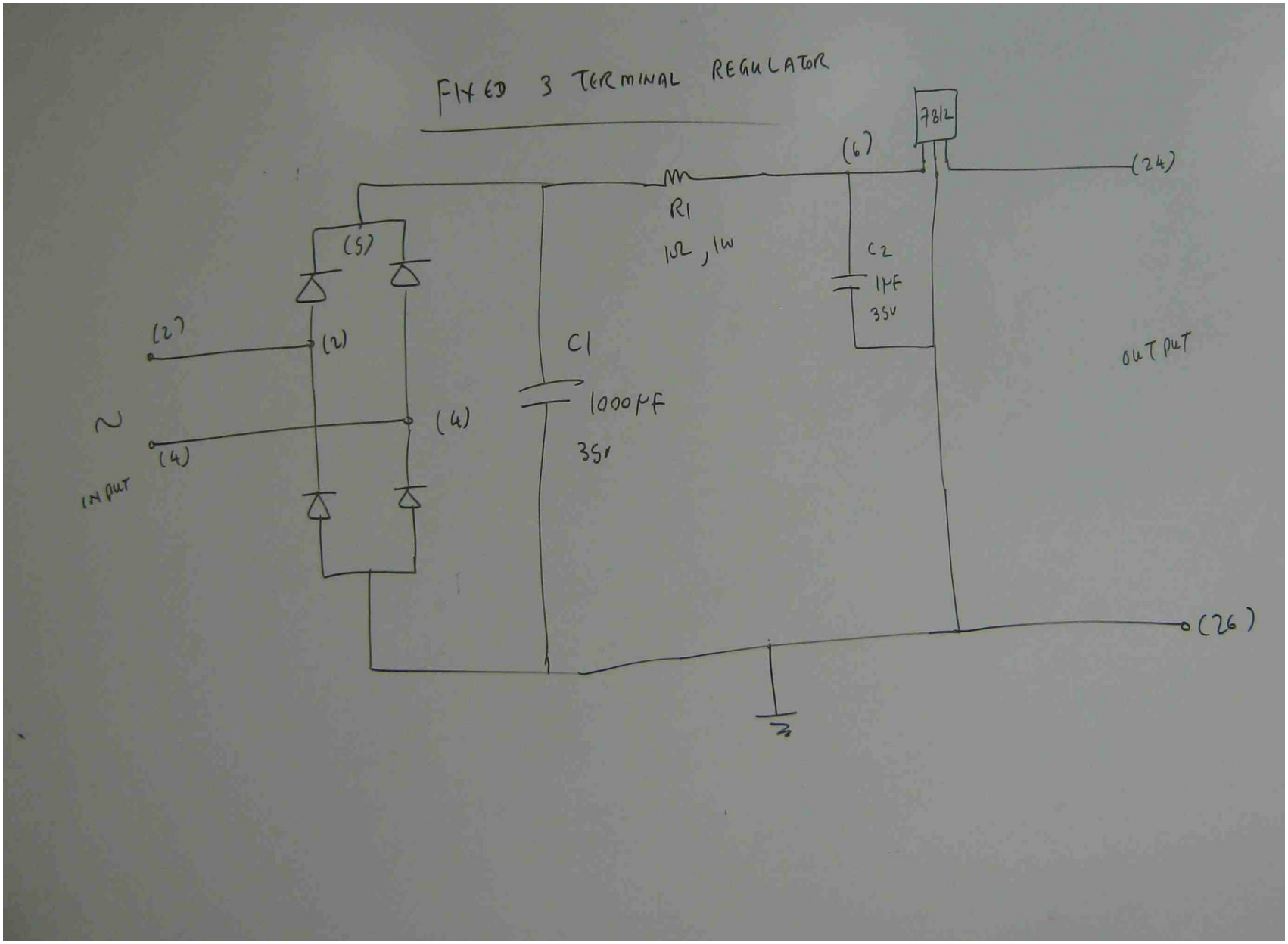

OHMS
10,000'S
8mA MAX
1,000'S
25mA MAX
100'S
80mA MAX
3Φ RECTIFIER BANK


9
6
7
8
9
6
7
8
9
STOR
8
Y. LTD SYDNEY
3 ϕ SCR

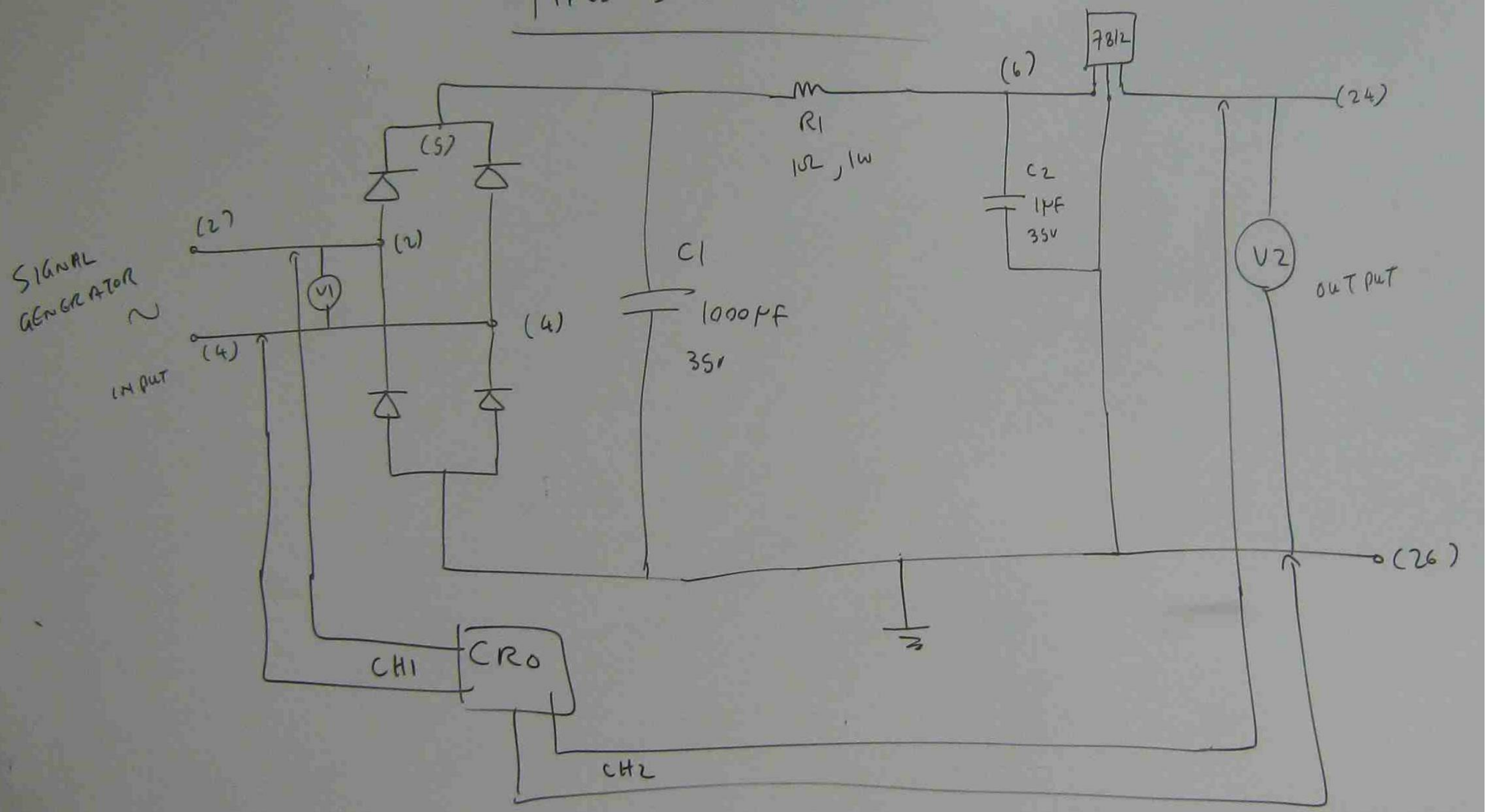

& TRIP
RESET
H.T.
ON
INDICATOR
208


TRIO 15MHz OSCILLOSCOPE CS-1560A
INDUCTANCE
DECADE RESISTOR


Cupboard No. 4


FIXED 3 TERMINAL REGULATOR
7812
(6)
(24)
R1
1Ω, 1W
C2
1μF
35V
(5)
(2)
(2)
C1
1000μF
35V
OUTPUT
~
(4)
(4)
INPUT
(26)

FIXED 3 TERMINAL REGULATOR



CONNECT THE GIVEN CIRCUIT

GIVE THE INPUT AC (SINUSOIDAL, SQUARE, SAW TOOTH)

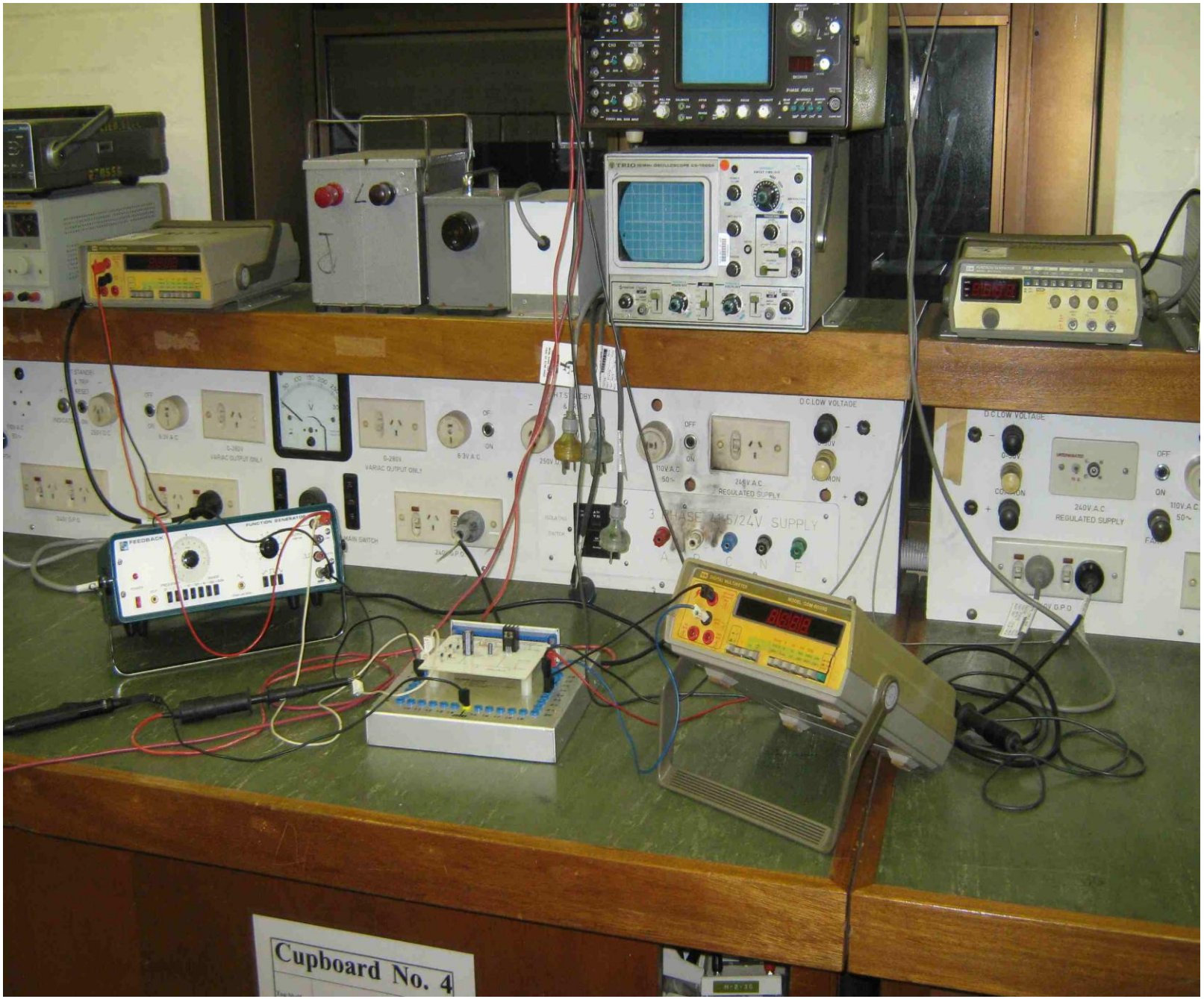
MEASURE OUT PUT VOLTAGE & NOTE WAVE FORM

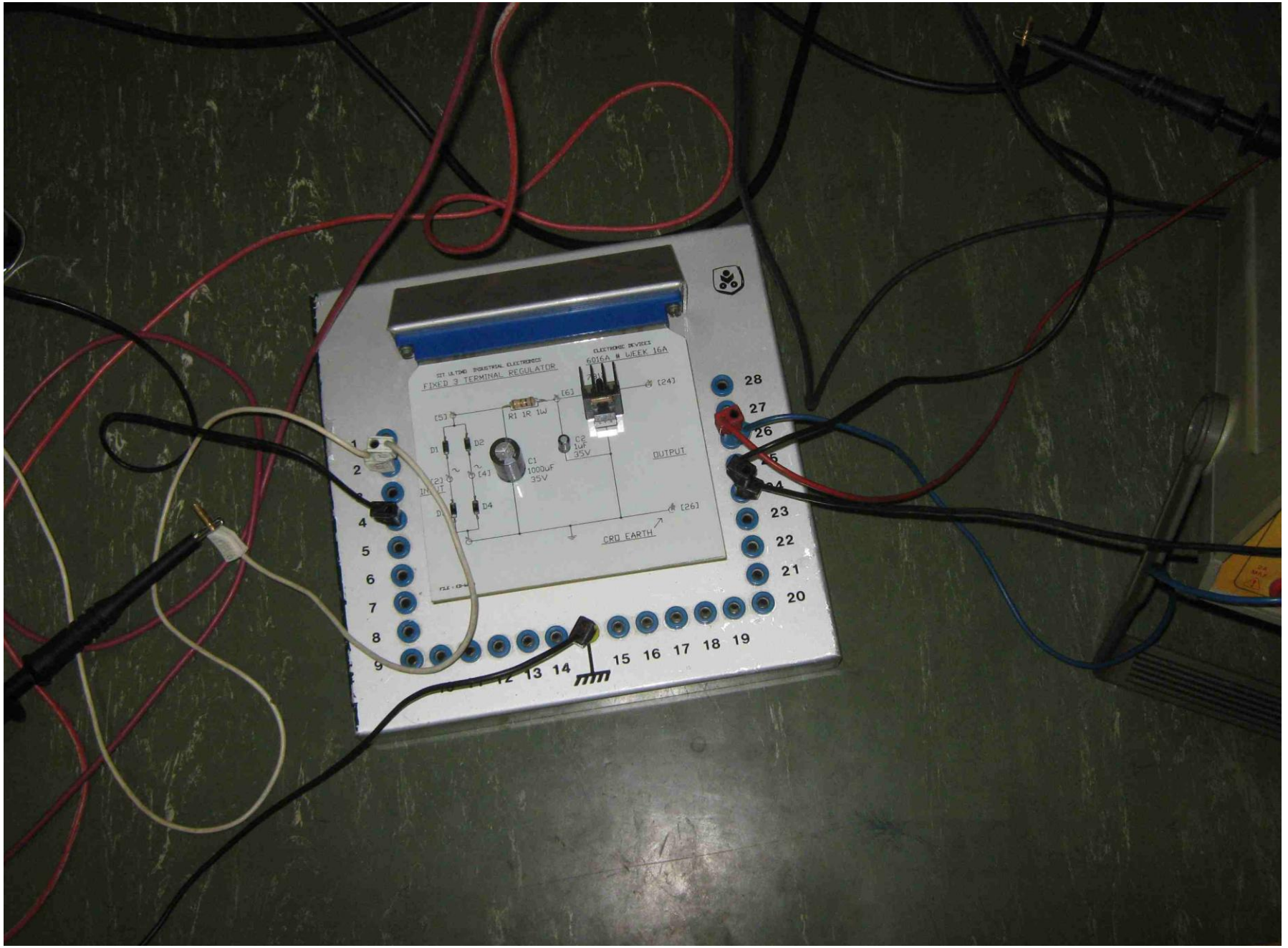
INPUT SINUSOIDAL VOLTAGE V_1	V_2	WAVE FORM

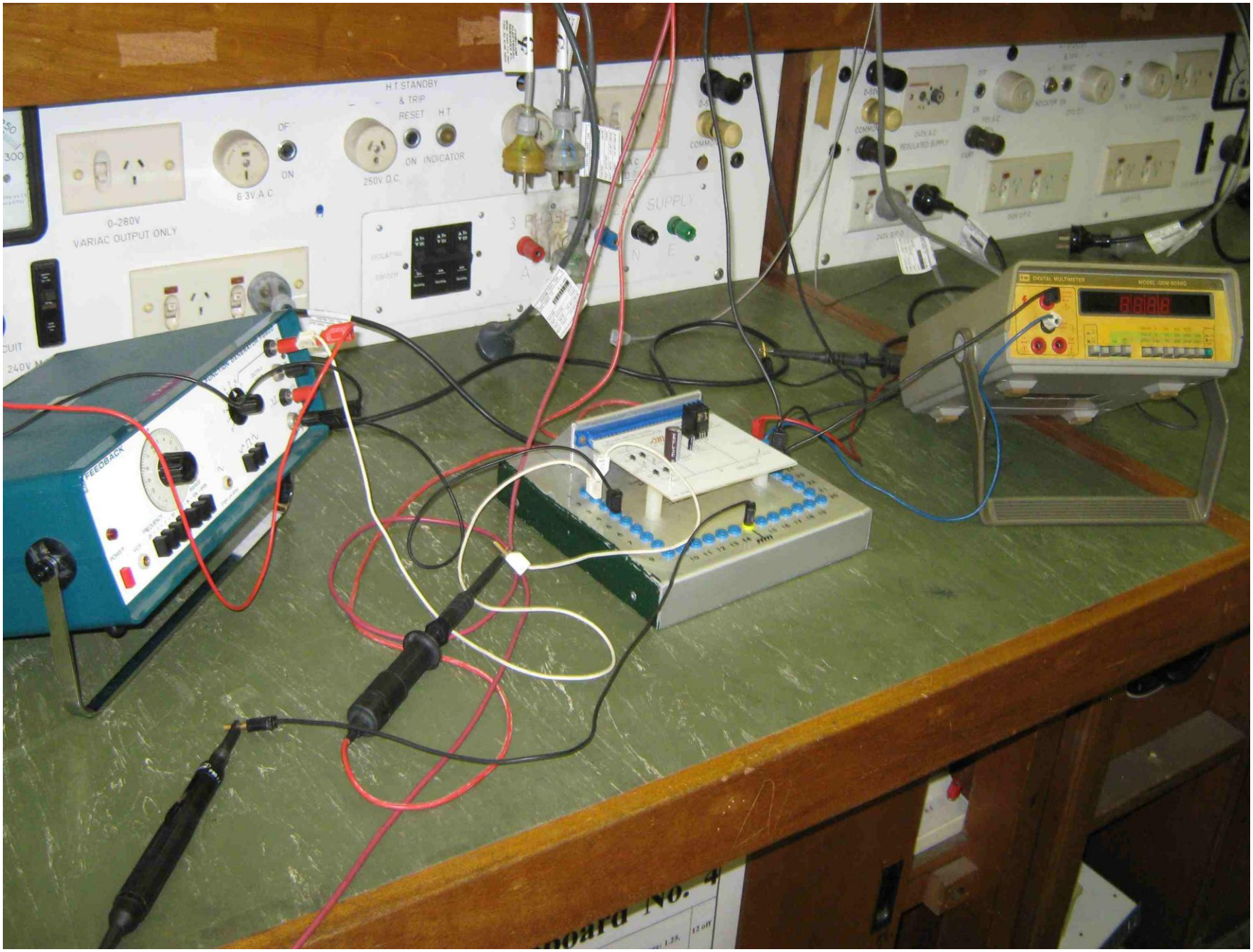
INPUT SINUSOIDAL FREQUENCY	V_1	V_2
–		
–		
–		
–		
–		

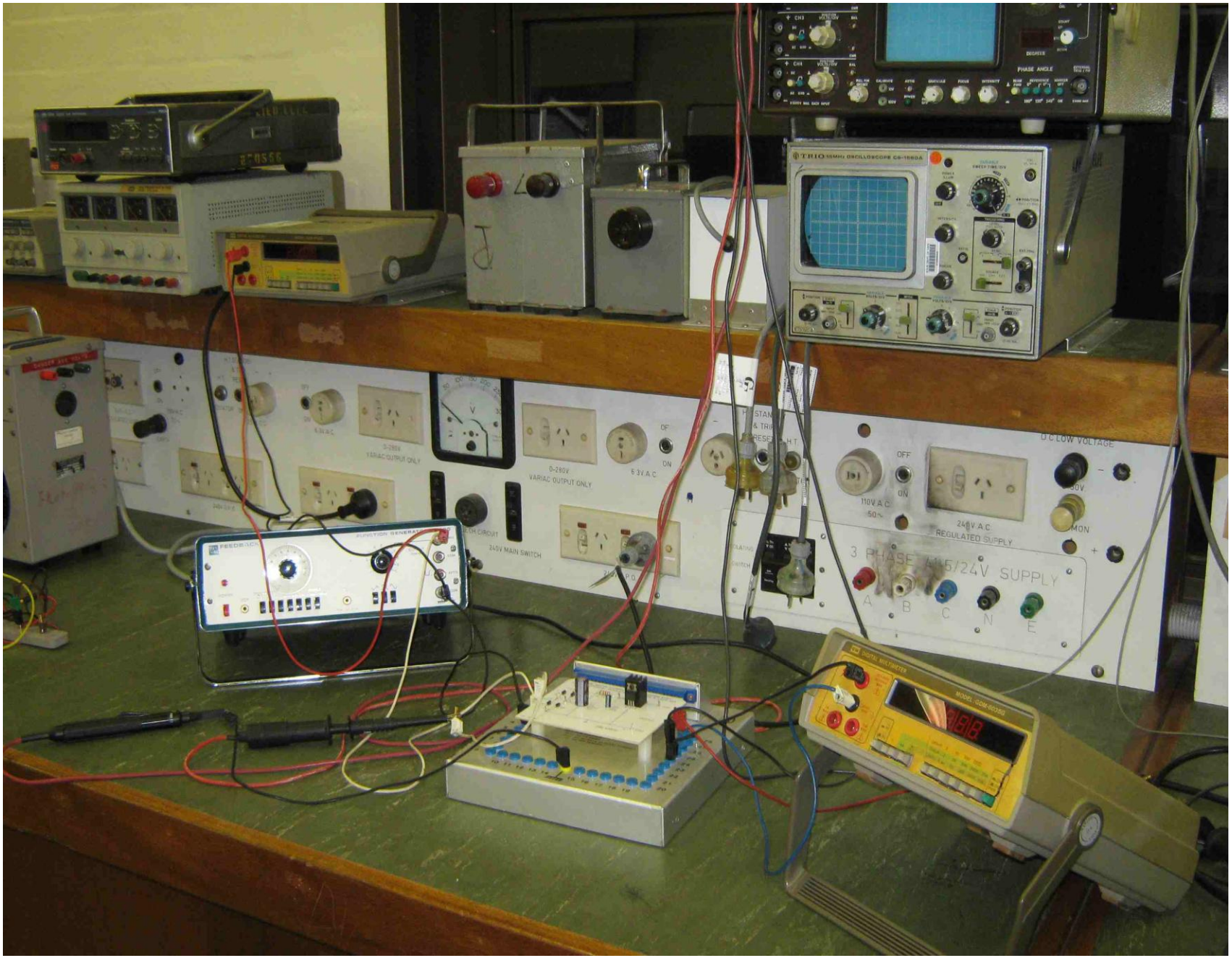
INPUT SQUARE WAVE FREQUENCY	V_1	V_2

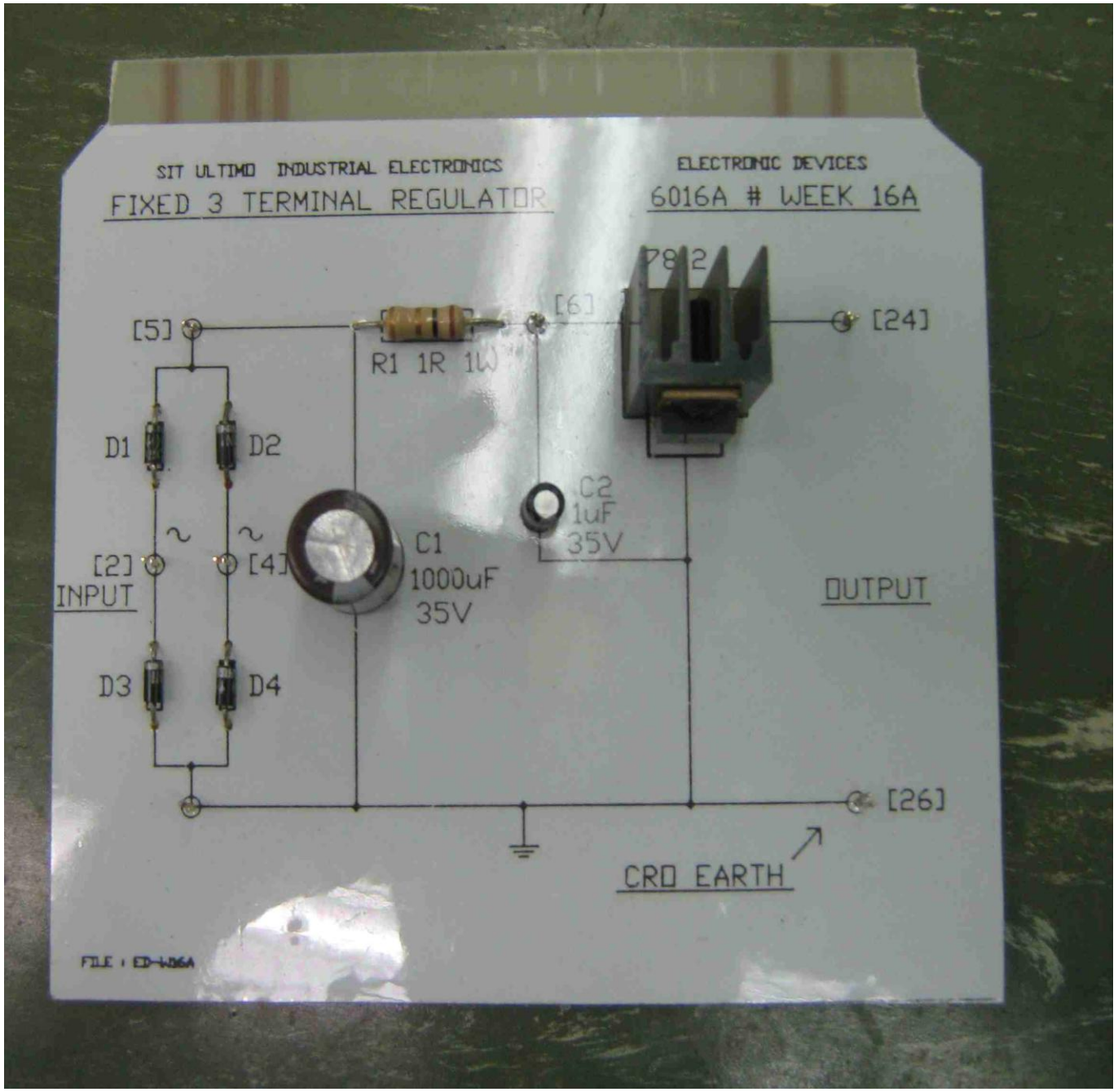
INPUT SAW TOOTH WAVE FREQUENCY	V_1	V_2
–		
–		
–		

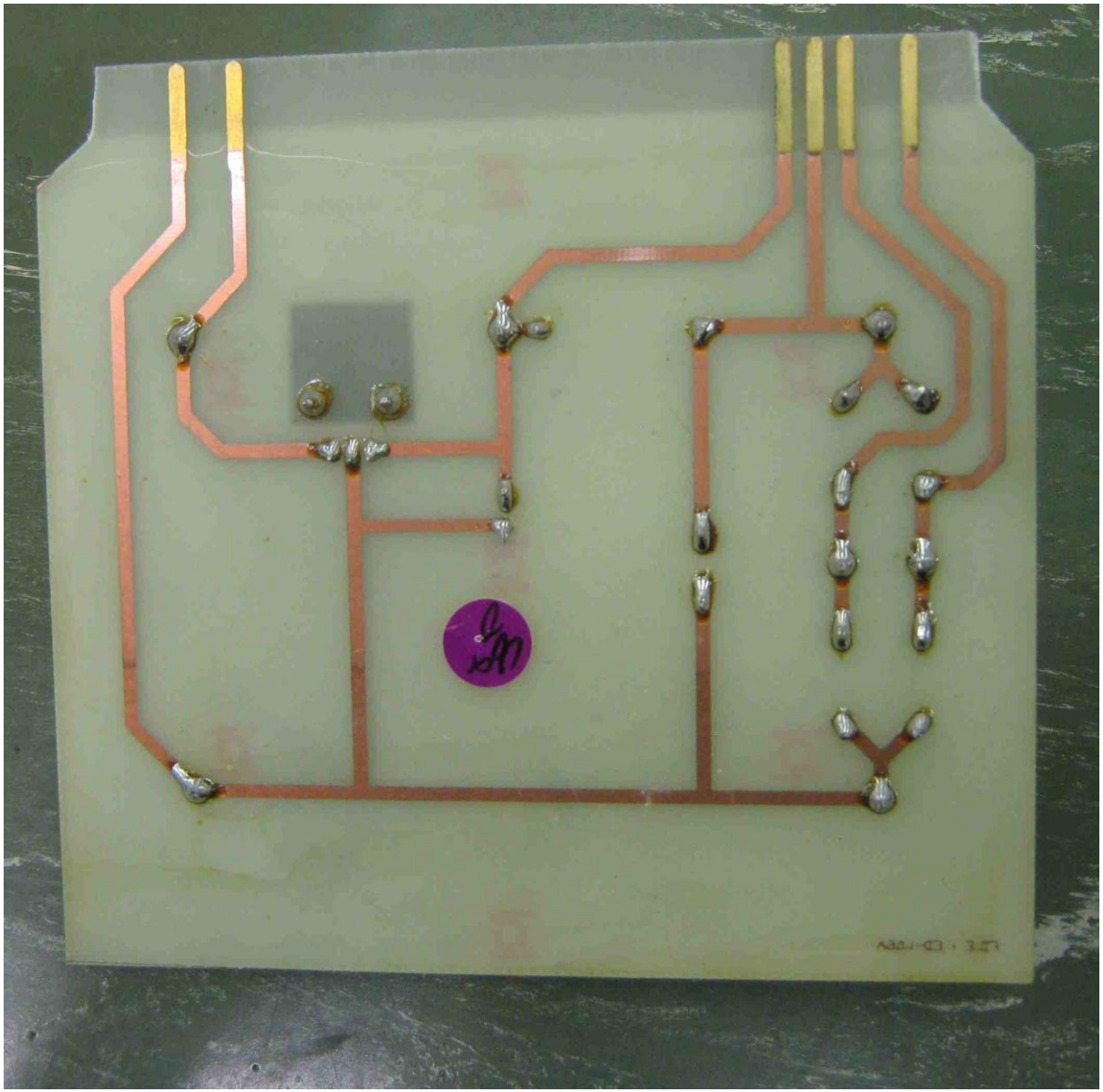

TRIO
FEEDBACK
D.C.LOW VOLTAGE
240V.A.C
REGULATED SUPPLY
110V.A.C
50~
Cupboard No. 4

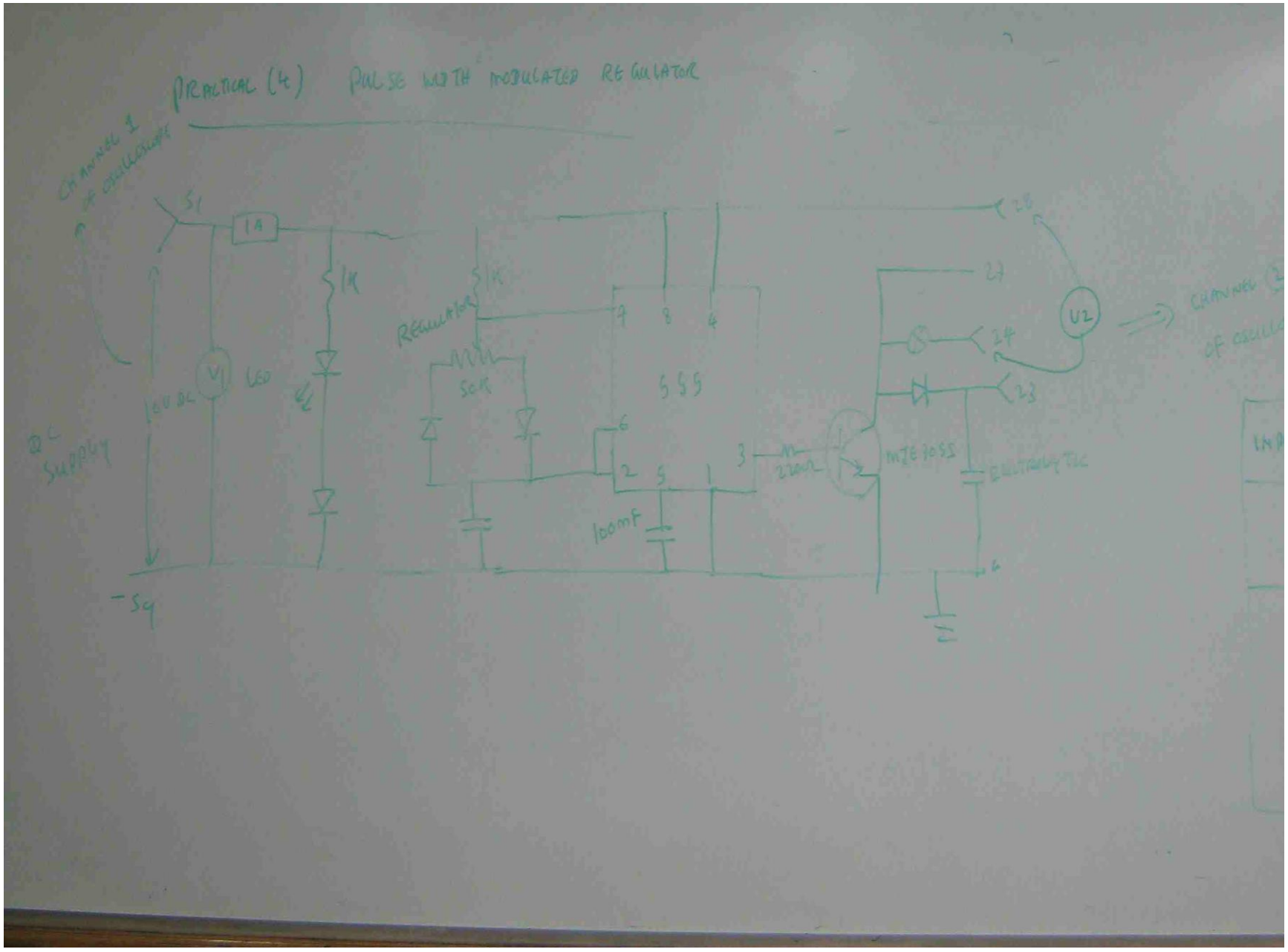

ELECTRONIC DEVICES
6016A & WEEK 16A
FIXED 3 TERMINAL REGULATOR
R1 1R 1W
C1
1000uF
35V
C2
1uF
35V
D1
D2
D4
[2]
[4]
[5]
[6]
[24]
[26]
OUTPUT
CRO EARTH
1
2
4
5
6
7
8
9
13
14
15
16
17
18
19
20
21
22
23
26
27
28

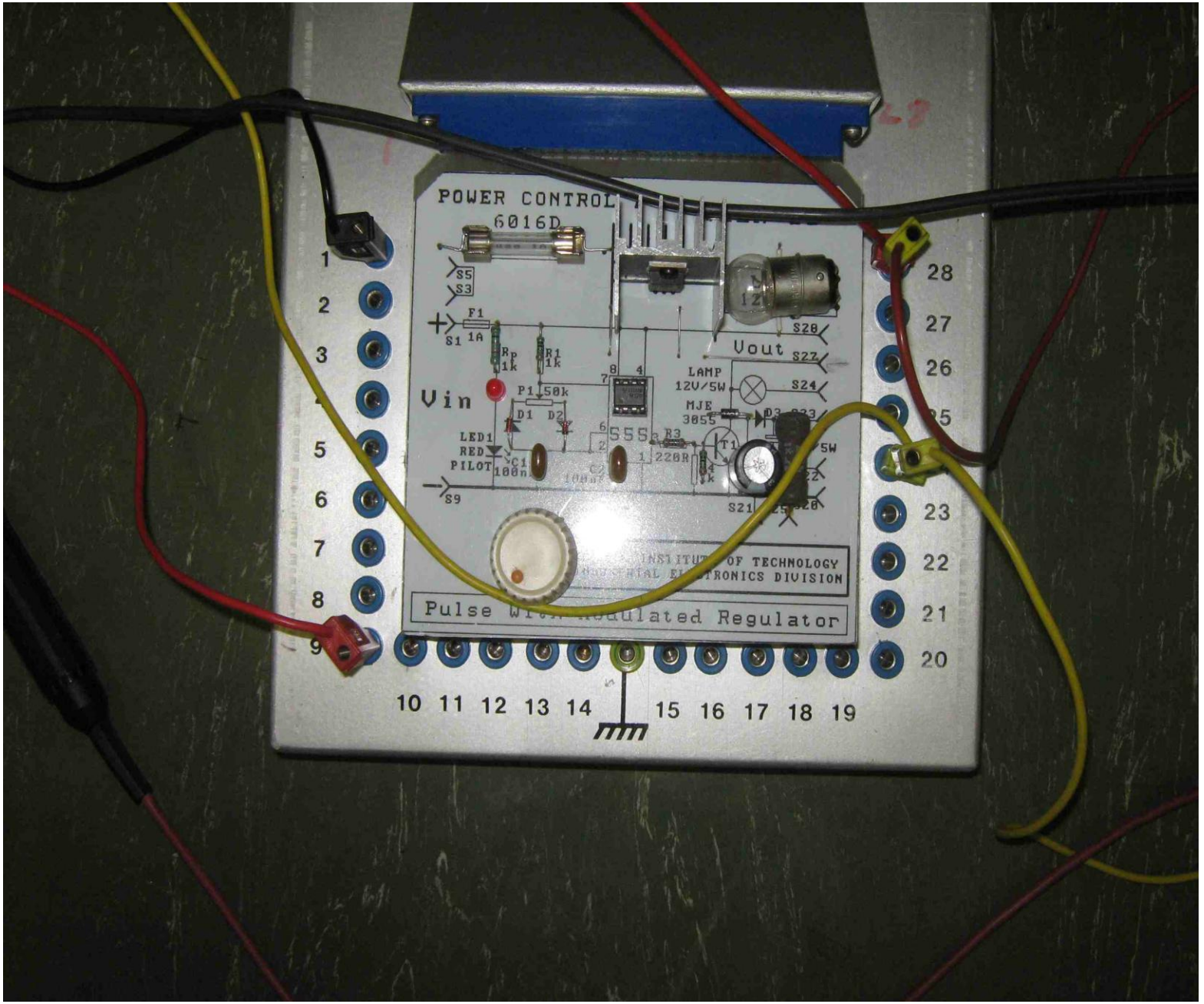

H.T. STANDBY
& TRIP
RESET
H.T
OFF
ON
ON
INDICATOR
6.3V A.C
250V D.C.
0-280V
VARIAC OUTPUT ONLY
FEEDBACK
DIGITAL MULTIMETER
MODEL GDM-9035G

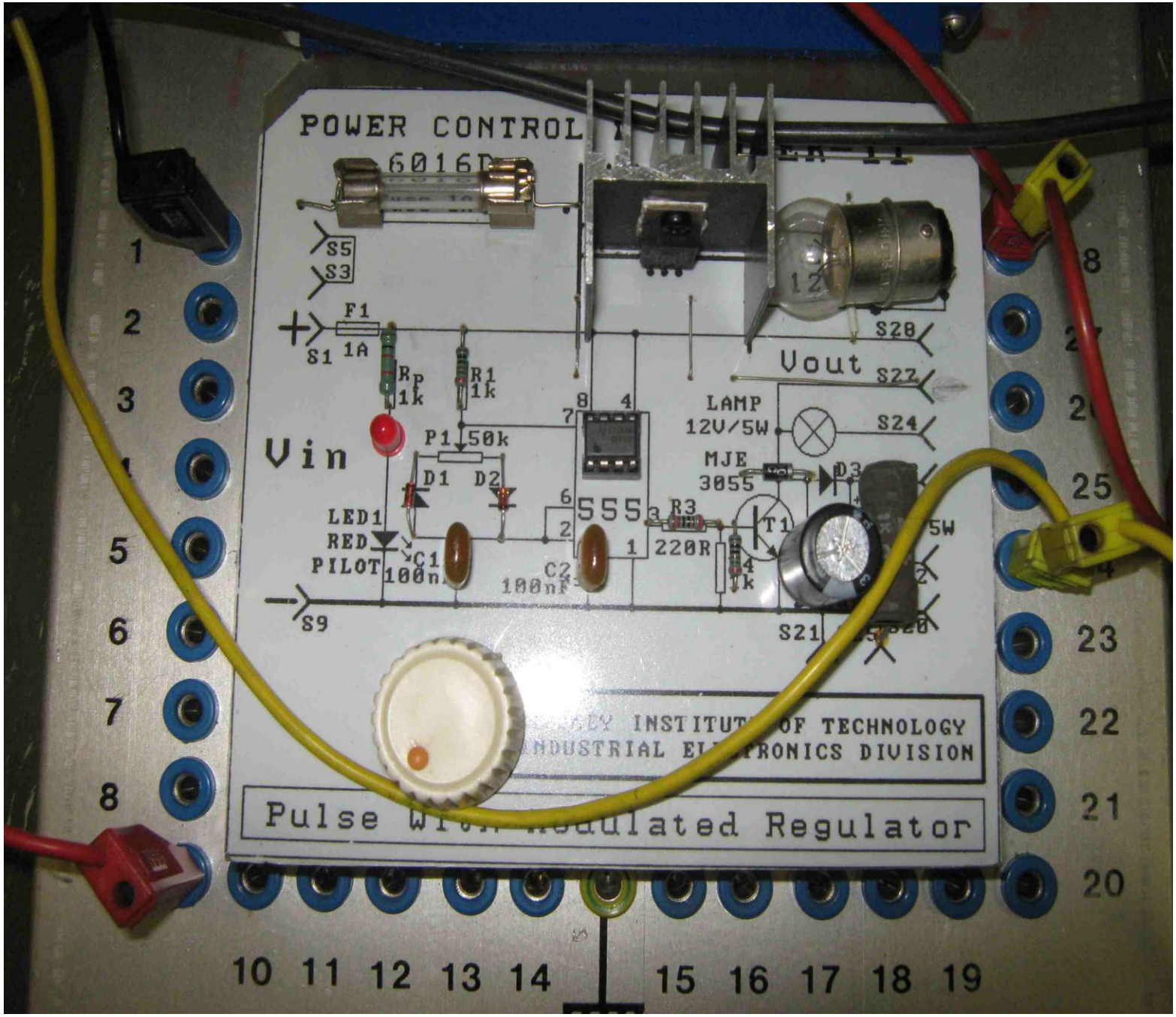

TRIO 15MHz OSCILLOSCOPE CS-1560A
PHASE ANGLE
D-28SV
VARIAC OUTPUT ONLY
6·3V.A.C.
OFF
ON
110V.A.C.
50~
240V.A.C.
REGULATED SUPPLY
DC LOW VOLTAGE
3 PHASE 415/24V SUPPLY
A B C N E
240V MAIN SWITCH


SIT ULTIMO INDUSTRIAL ELECTRONICS
ELECTRONIC DEVICES
FIXED 3 TERMINAL REGULATOR
6016A # WEEK 16A
7812
[5]
[6]
[24]
R1 1R 1W
D1
D2
C2
1uF
35V
C1
1000uF
35V
[2]
INPUT
[4]
OUTPUT
D3
D4
[26]
CRO EARTH



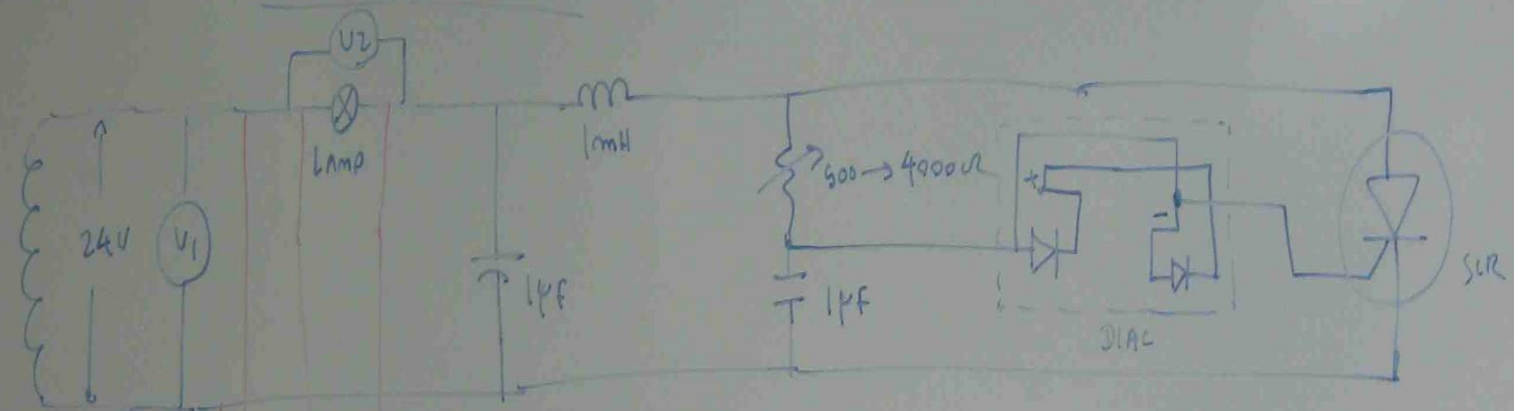

PRACTICAL (4) PULSE WIDTH MODULATED REGULATOR
CHANNEL 1 OF OSCILLOSCOPE
S1
1A
1K
LED
V1
12V DC
DC SUPPLY
-Sq
REGULATOR
SCR
555
100mf
MJE 1055
U2
CHANNEL 2 OF OSCILLOSCOPE


POWER CONTROL
6016D
Vin
Vout
LAMP
12V/5W
MJE
3055
LED1
RED
PILOT
555
Pulse Width Modulated Regulator
1 2 3 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 25 26 27 28


POWER CONTROL
6016D
S5
S3
F1
1A
S1
Rp
1k
R1
1k
P1
50k
Vin
D1
D2
LED1
RED
PILOT
C1
100n
C2
100nF
555
8
4
7
6
2
1
3
R3
220R
4
k
T1
MJE
3055
D3
LAMP
12V/5W
Vout
S28
S22
S24
S21
S9
INSTITUTE OF TECHNOLOGY
INDUSTRIAL ELECTRONICS DIVISION
Pulse Width Modulated Regulator
1
2
3
5
6
7
8
10
11
12
13
14
15
16
17
18
19
20
21
22
23
25

PRACTICAL

SCR DRIVE SYSTEM

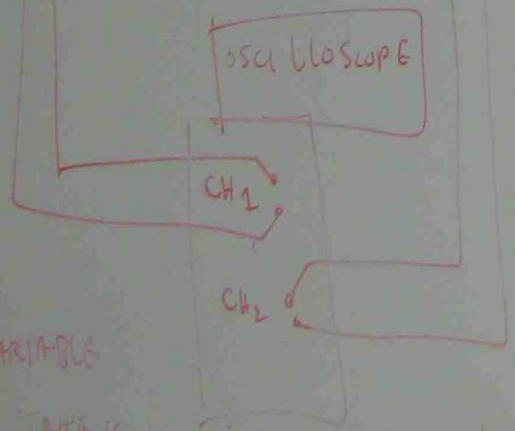


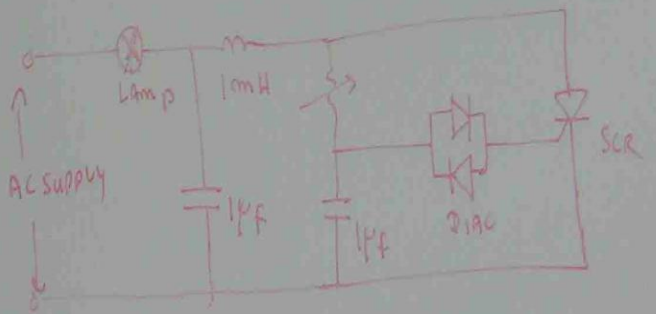
(1) CONNECT THE CIRCUIT

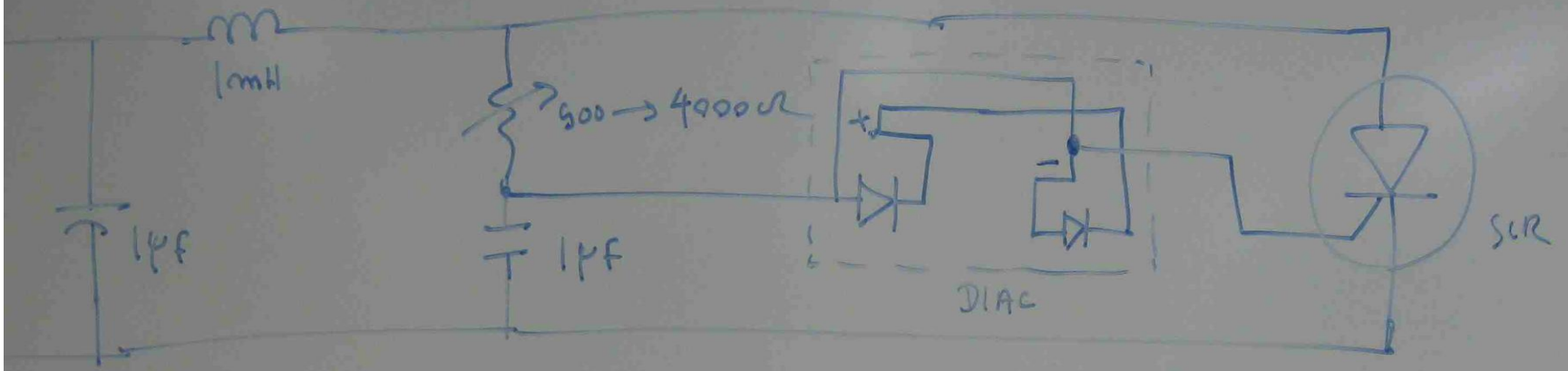
(2) TAKE READING OF V_1, V_2

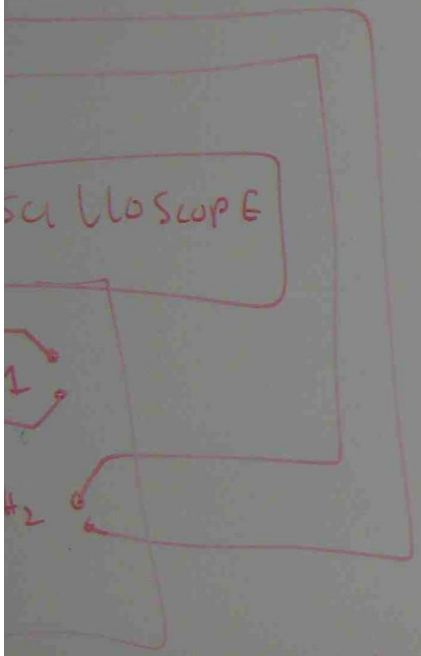
(3) OBSERVE LAMP

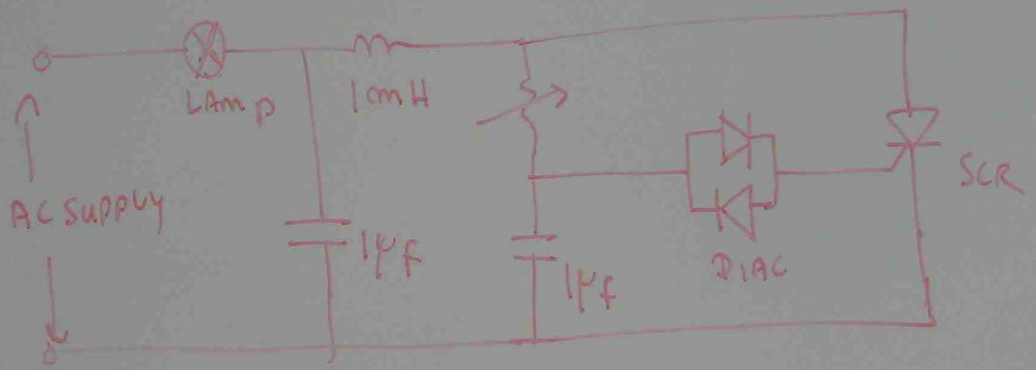
(4) CHANGE VARIABLE RESISTANCE, TAKE V_1, V_2, SKETCH THE INPUT OUTPUT WAVEFORMS.











SKETCH THE INPUT / OUT PUT WAVEFORMS.

SCR

VARIABLE RESISTANCE	V_1	V_2	LAMP DARK OR/ BRIGHT	SKETCH OF V_2 WAVE FORM
700Ω				
1700Ω				
2700Ω				
3700Ω				
4700Ω				

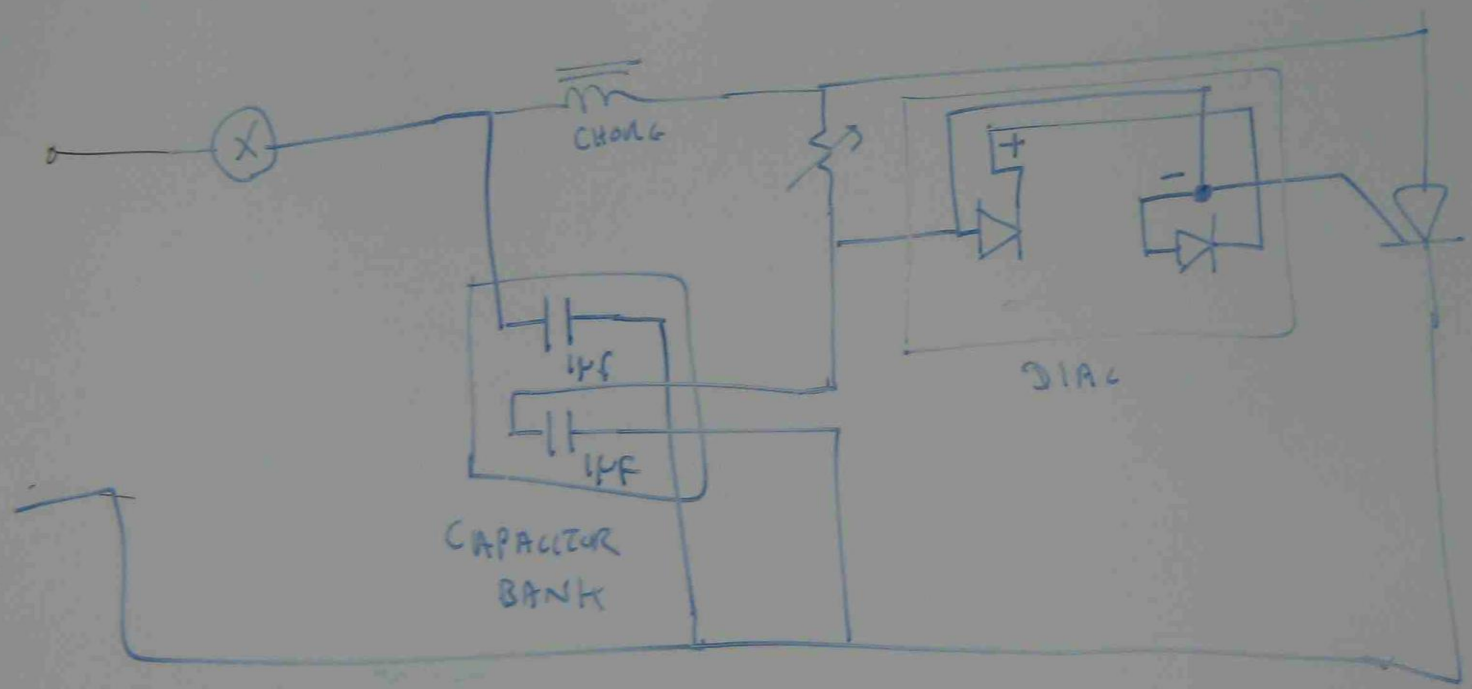
QUESTION

WHAT IS CUT OFF POINT RESISTANCE FOR LAMP?

WHEN THE RESISTANCE IS REDUCED BACK, WHAT HAPPENS TO LAMP?

HAND ON PRACTICAL WORK

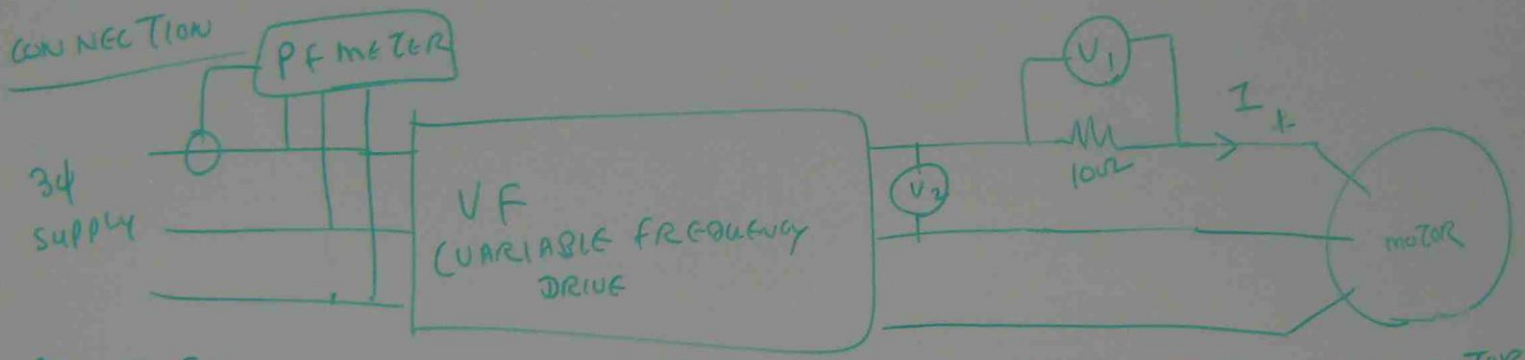
- CONNECT THE GIVEN CIRCUIT
- SHOW IT TO TEACHER
- DISCONNECT THE CIRCUIT
- PUT EQUIPMENTS BACK.


CHOKE
+
-
1μf
1μF
DIAC
CAPACITOR
BANK

VARIABLE FREQUENCY DRIVE SYSTEM

AIM TO INVESTIGATE POWER, CURRENT DRAWN BY 3ϕ MOTOR DRIVEN BY VARIABLE FREQUENCY / VARIABLE DRIVE SYSTEM

CONNECTION



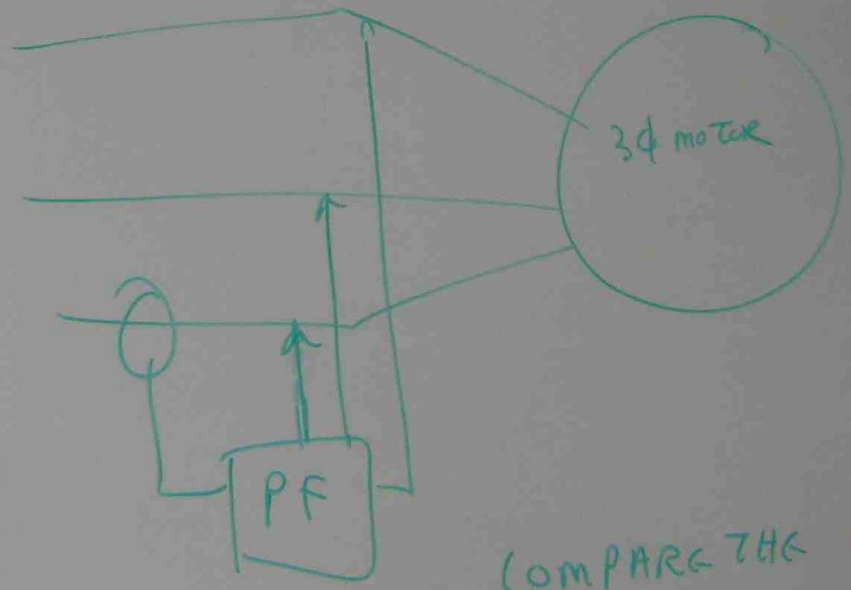
ADJUST FREQUENCY AND MEASURE THE CURRENT AND VOLTAGE ACROSS MOTOR

TAKE 3 → 4 READINGS

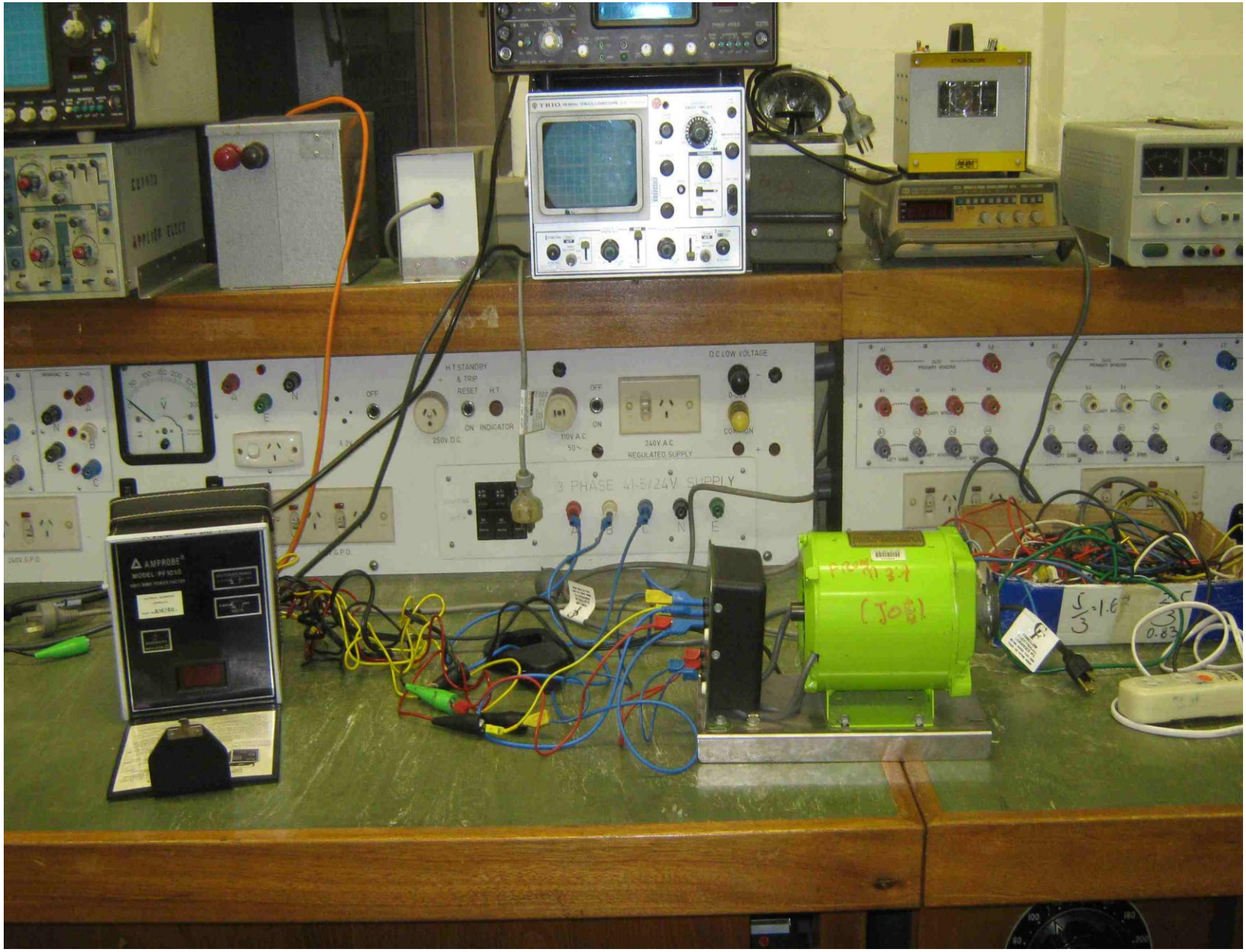
FREQUENCY	V_2	V_1	$I_1 = \frac{V1}{10\Omega}$	POWER (V.A) $= \sqrt{3} V_2 I_1$

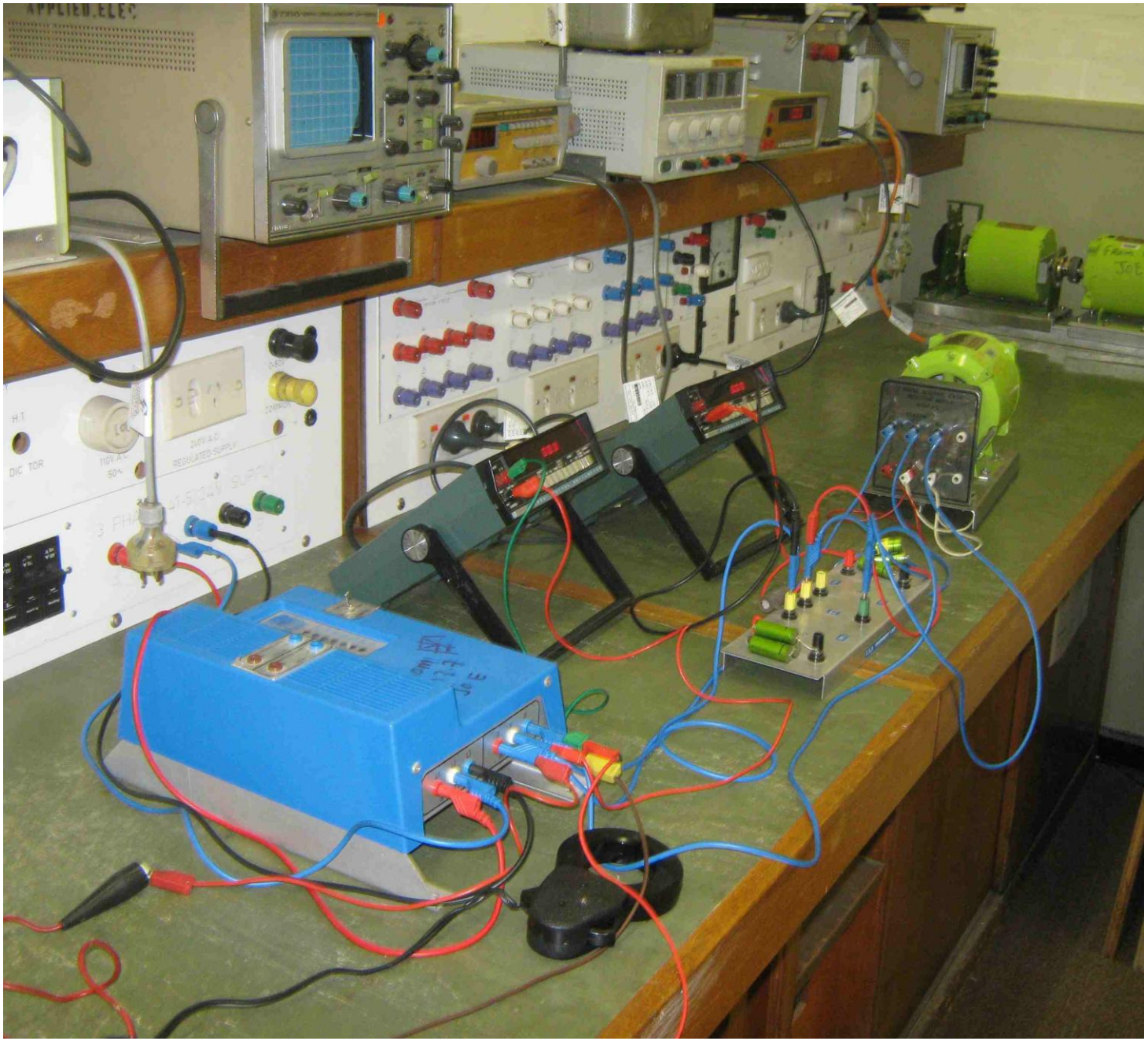
NOTE P.F METER READING AT SUPPLY TO V.F DRIVE

MEASURE P.F DIRECTLY ACROSS THE MOTOR



COMPARE THE READINGS.

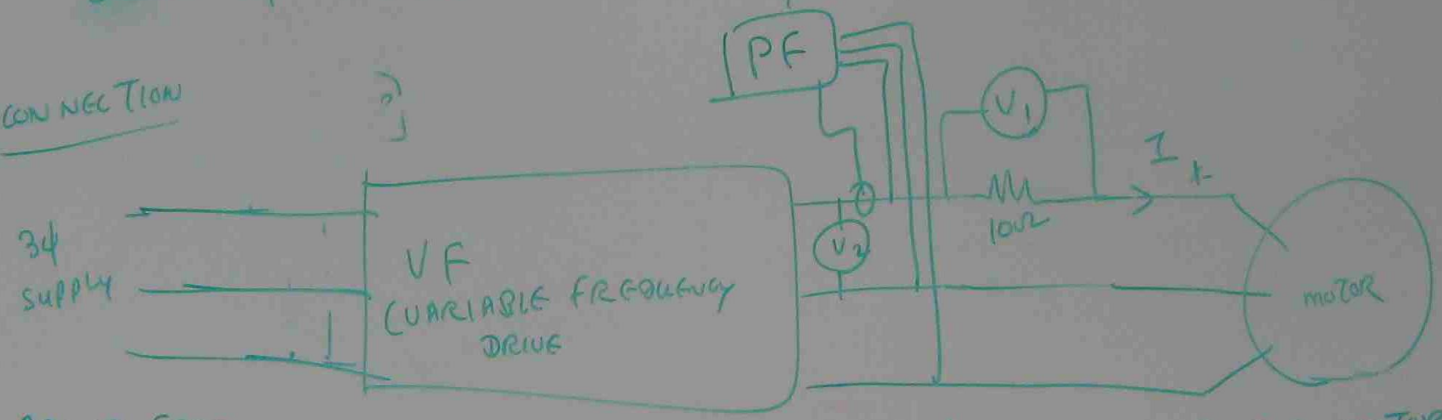

TRIO
H.T STANDBY
& TRIP
RESET
H.T
ON INDICATOR
250V D.C
110V A.C
50~
240V A.C
REGULATED SUPPLY
D.C LOW VOLTAGE
3 PHASE 415/24V SUPPLY
AMPROBE
MODEL PF 10 D
[Joe]



VARIABLE FREQUENCY DRIVE SYSTEM

AIM TO INVESTIGATE POWER, CURRENT DRAWN BY 3ϕ MOTOR DRIVEN BY VARIABLE FREQUENCY / VARIABLE DRIVE SYSTEM

CONNECTION



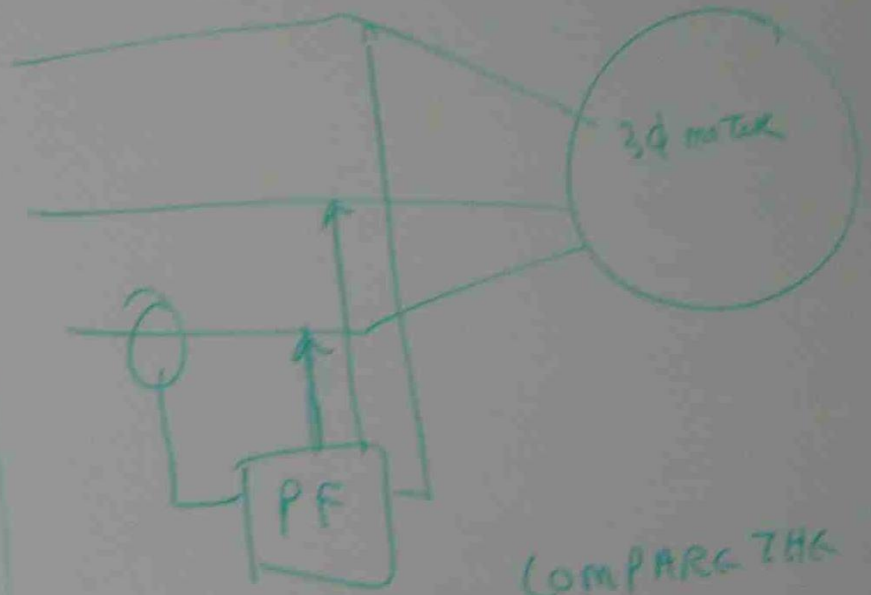
ADJUST FREQUENCY AND MEASURE THE CURRENT AND VOLTAGE ACROSS MOTOR

TAKE 3 → 4 READINGS

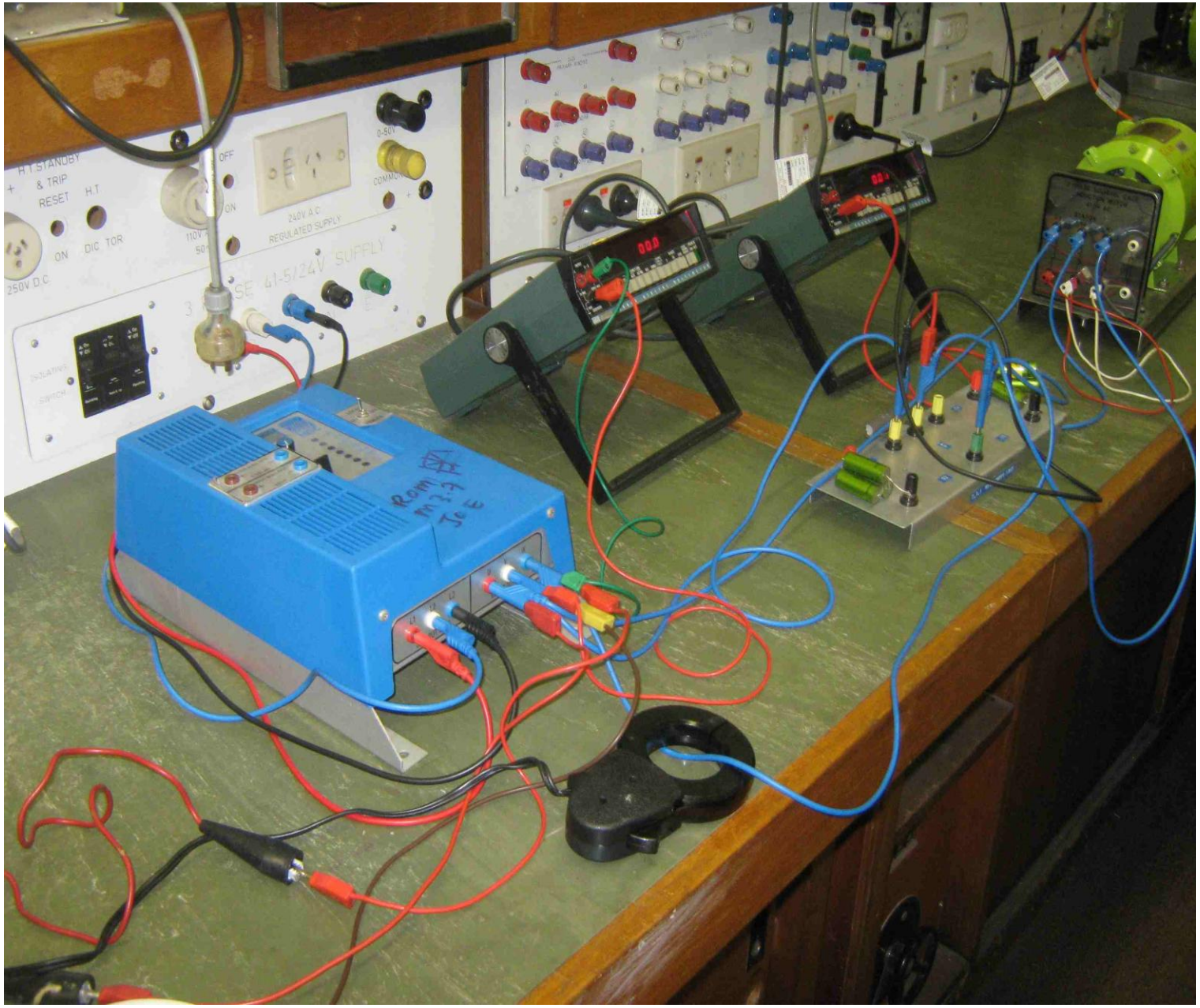
FREQUENCY	V_2	V_1	$I_1 = \frac{V1}{10\Omega}$	POWER (U.A) $= \sqrt{3} V_2 I_1$

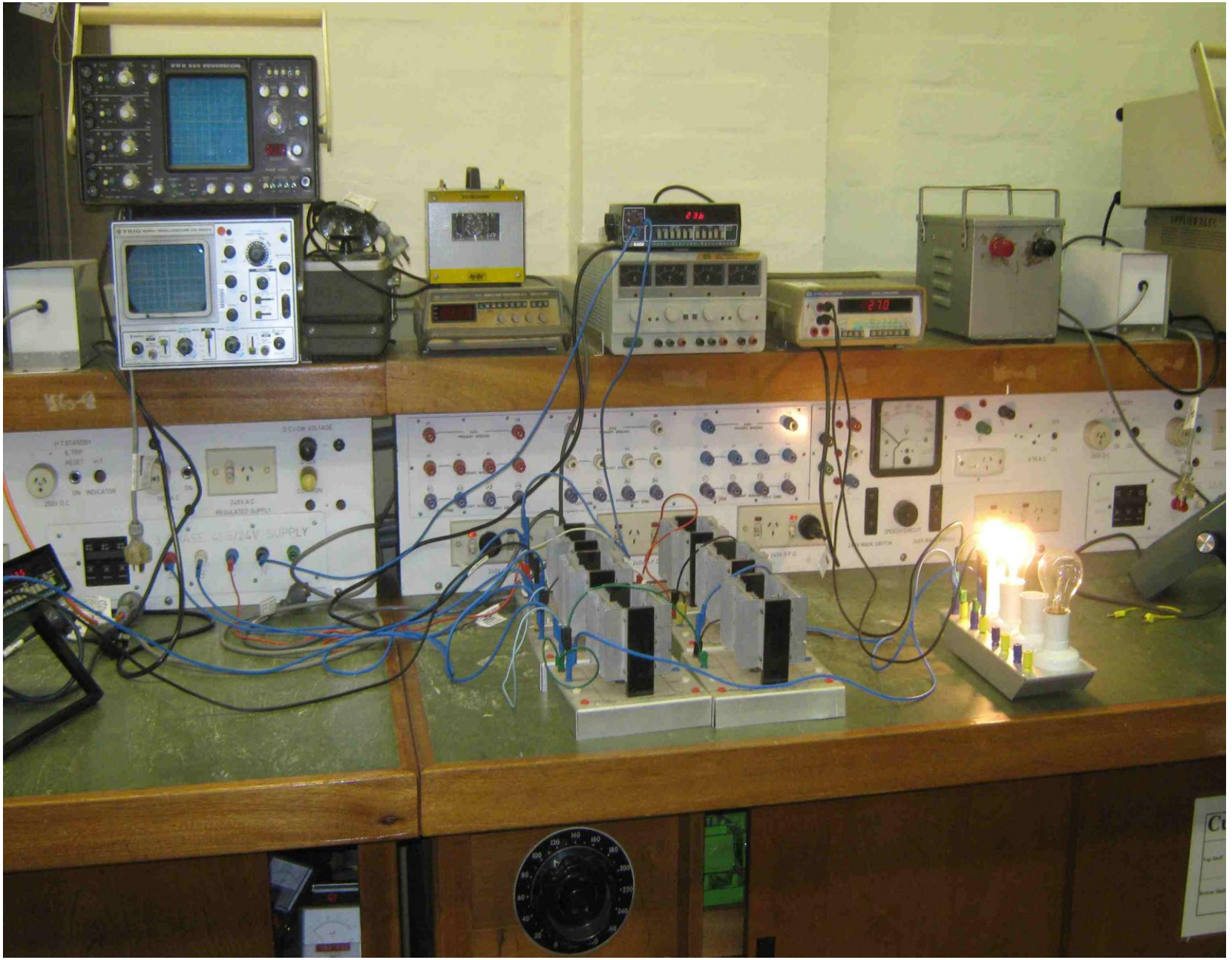
NOTE P.F METER READING AT SUPPLY TO V.F DRIVE

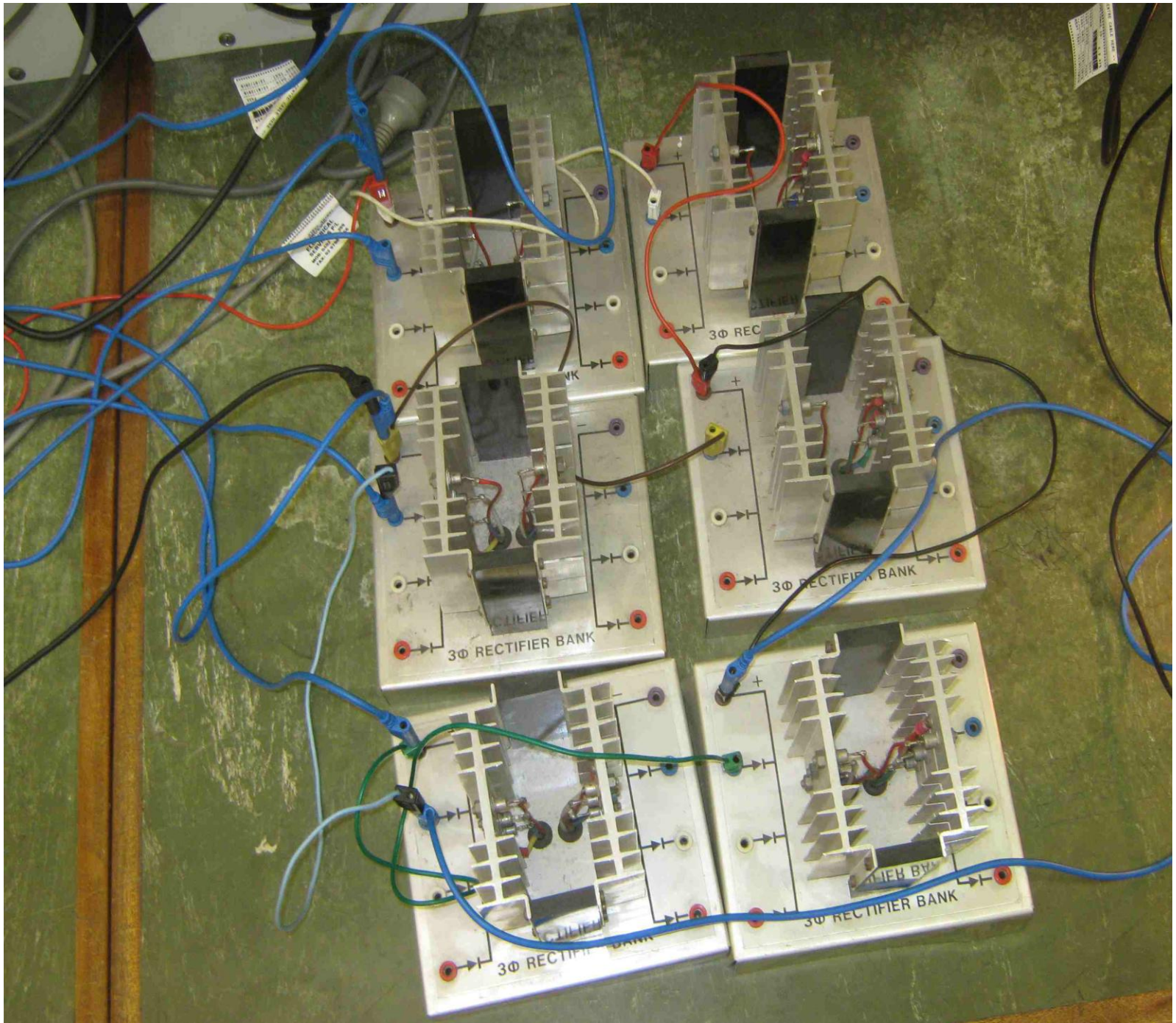
MEASURE P.F DIRECTLY ACROSS THE MOTOR

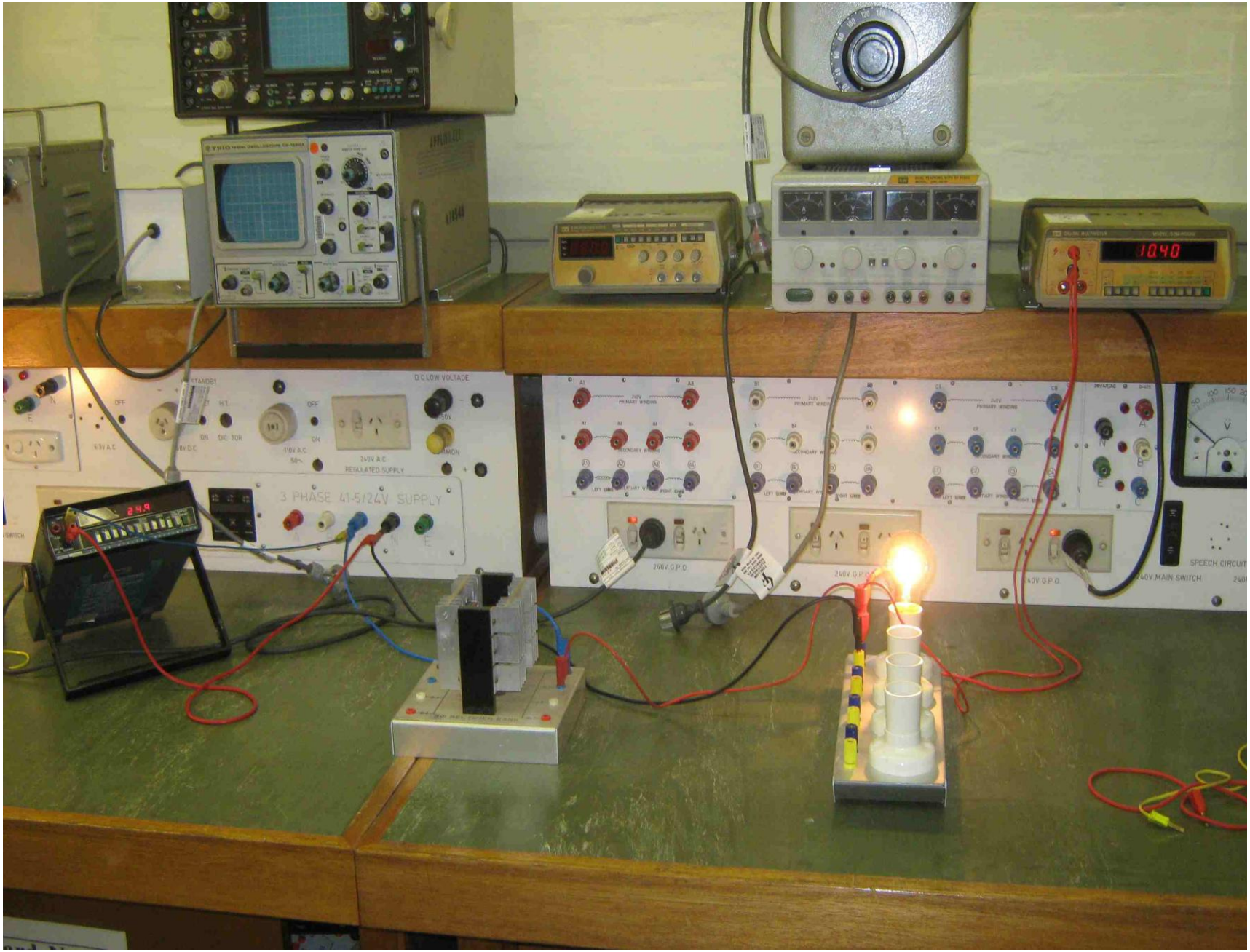


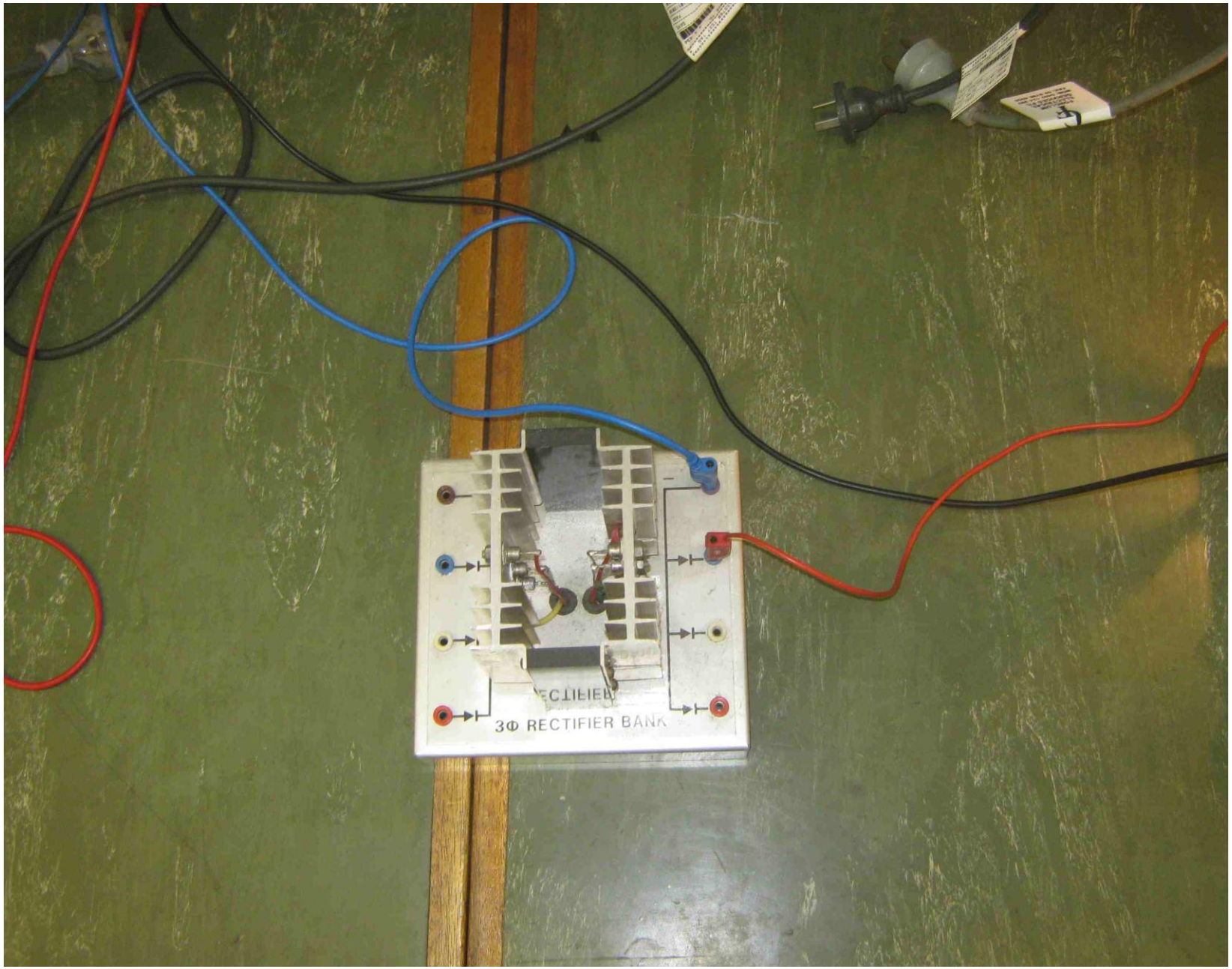
COMPARE THE READINGS.


H.T STANDBY
& TRIP
RESET
H.T
ON
OFF
240V A.C.
REGULATED SUPPLY
250V D.C.




3Φ RECTIFIER BANK
3Φ RECTIFIER BANK


D.C LOW VOLTAGE
240V A.C
REGULATED SUPPLY
3 PHASE 41-5/24V SUPPLY
240V G.P.O
240V MAIN SWITCH
SPEECH CIRCUIT


3Φ RECTIFIER BANK