0 – 20 mA signal, as 0 –mA usually refers to a n open-circuit or error. Having 4 mA rather than 0 mA as the lower limit allows the 0 mA condition to be used as the error condition.

5. An RTD is a "resistive temperature device "and is used to measure temperature. It is a transducer which converts changes in temperature to changes in resistance (which can be converted to changes in voltage). The general operating range of an RTD is -200 to 700 degrees Celsius If the temperature of the process is in the range of -200 to 500 degrees Celsius, an RTD is the preferred option. For extremely high temperatures, a thermocouple is usually used.

Some other types of temperature sensors are :

The thermocouple which converts changes in temperature to small changes in voltage. Thermocouples have a higher range than R.T.Ds: a K – type thermocouple has a short term range of -180 to 3000 degrees Celsius.

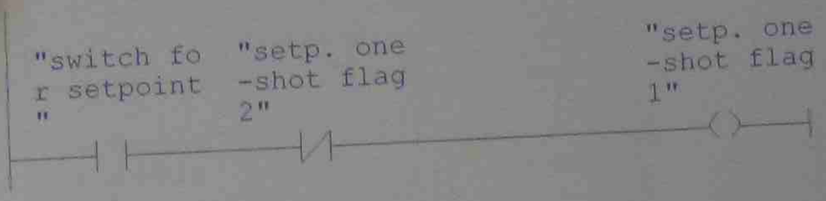
The thermistor (thermally sensitive resistor) which converts changes in temperature to changes in resistance and has an operating range of -200 to +1000 degrees Celsius. They achieve a higher precision over a small temperature range : from -90 to 130 degrees Celsius.

The thermostat which converts changes in temperature to changes in resistance, or current flow. Wax pellet thermostats have an operating range of 70 to 90 degrees Celsius.

The bi-metallic strip converts temperature change to mechanical displacement. Operating Range unknown.

etwork: 2

one shot for setpoint

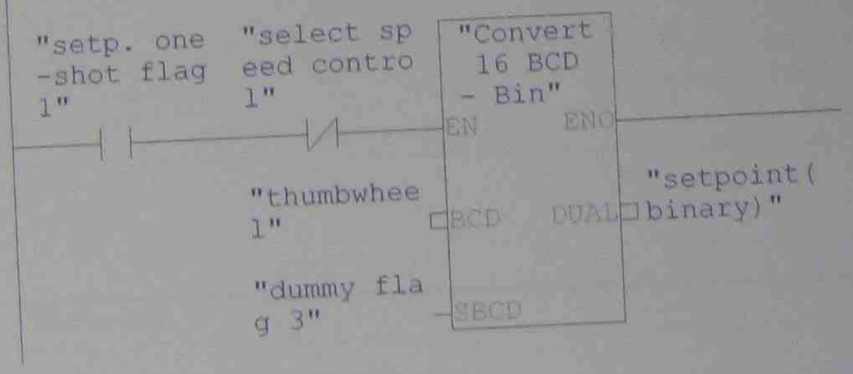


Network: 3

one shot for setpoint

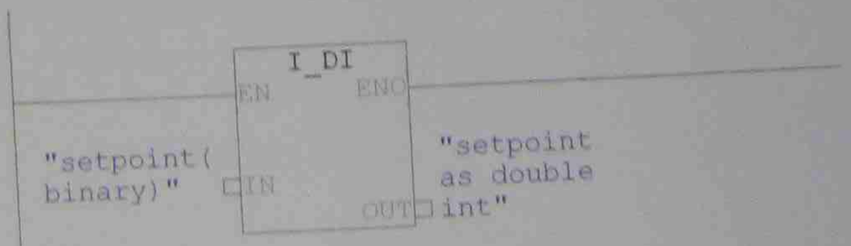
Network: 4

convert BCD setpoint to binary if pushbutton speed control is used, to not load thumbwhel setpoint



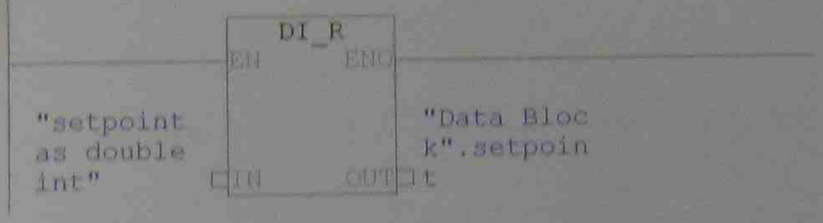
Network: 5

convert binary setpoint to double int



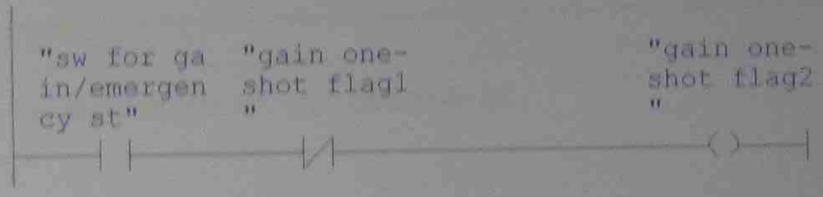
Network: 6

double int to real: setpoint to real no.



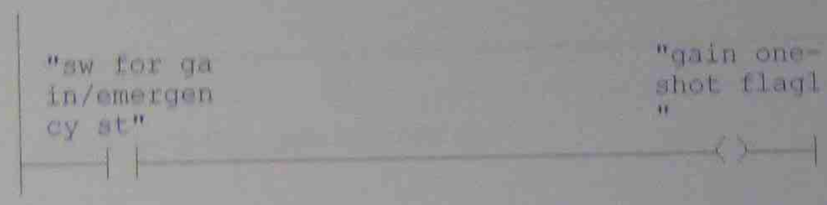
Network: 7

one shot for gain



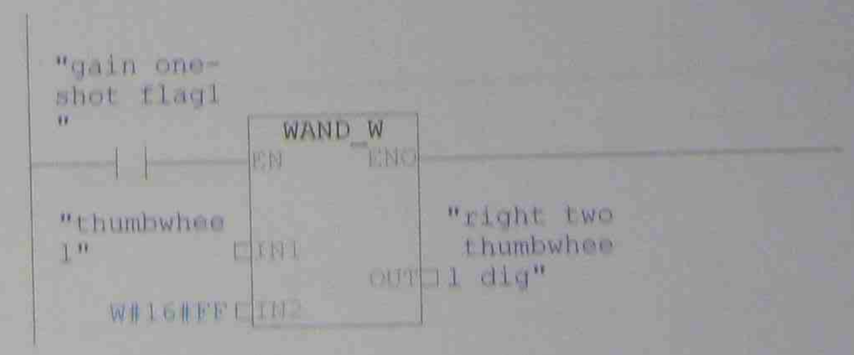
Network: 8

one shot for gain



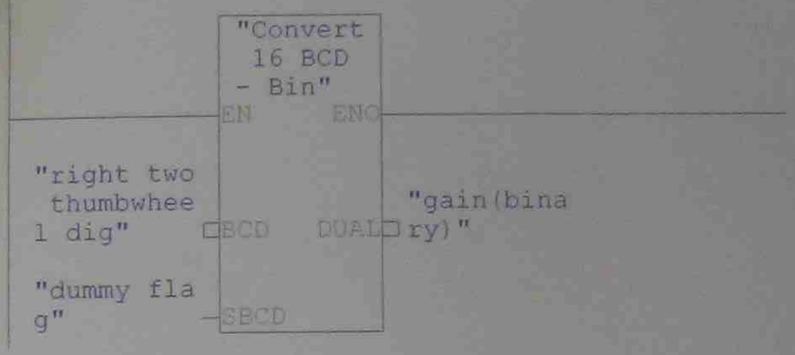
Network: 9

get thumbwheel data for gain



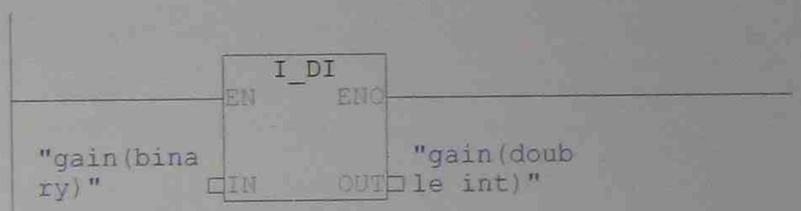
Network: 10

convert BCD gain to binary



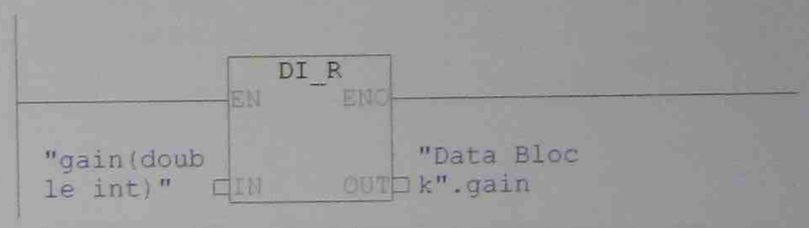
Network: 11

binary to double int : gain as double int



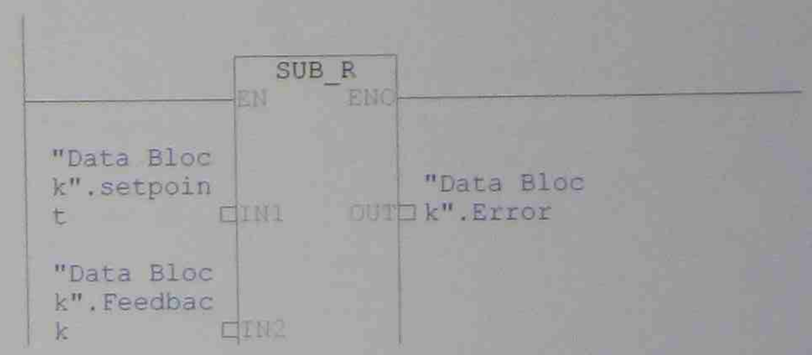
Network: 12

double int to real : gain as a real no.



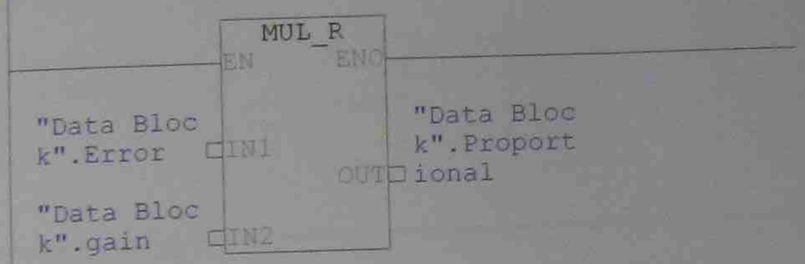
Network: 13

calculate error



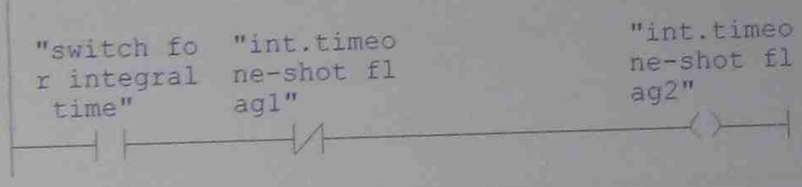
Network: 14

proportional action



Network: 15

one shot for integral time

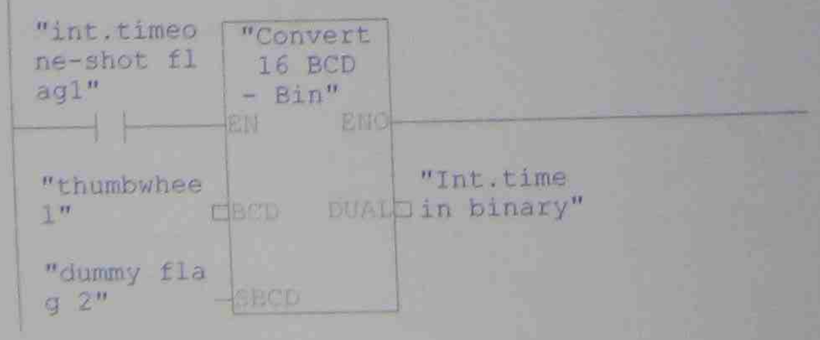


Network: 16

one shot for integral time

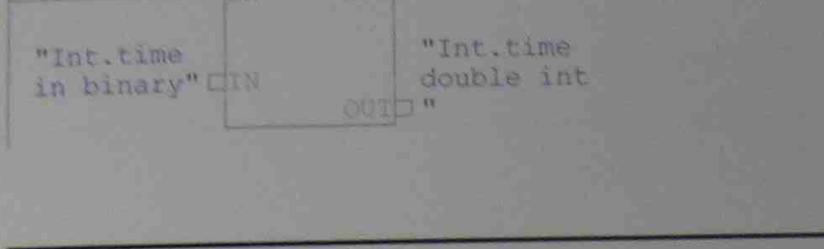
Network: 17

get Integral time from thumbwheel: convert BCD integral time to binary



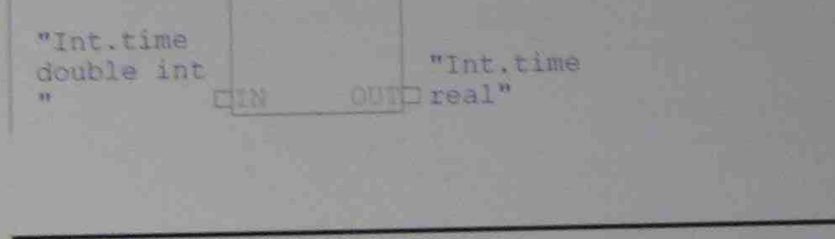
Network: 18

convert binary integral time to double int



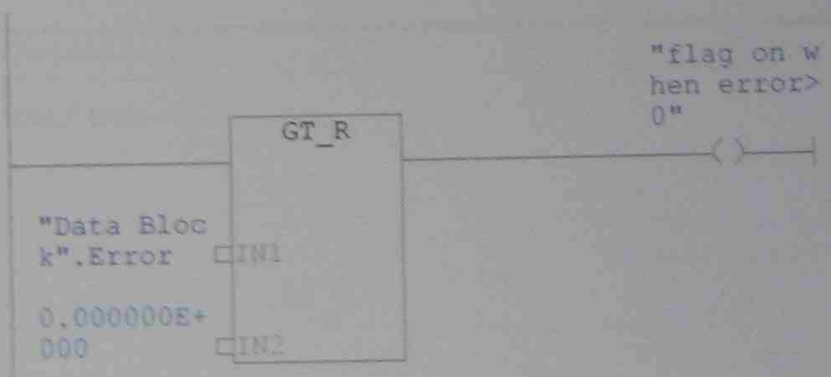
Network: 19

convert double int integral time to real no.



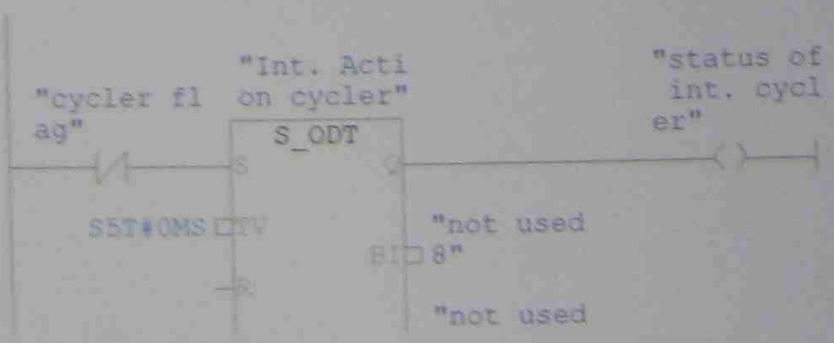
Network: 20

test error > 0



Network: 21

cycler for integral action



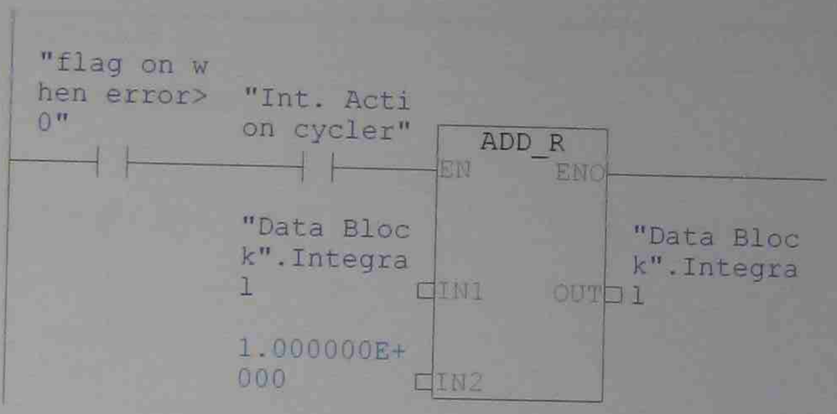
BCD 9"

Network: 22

cycler for integral action

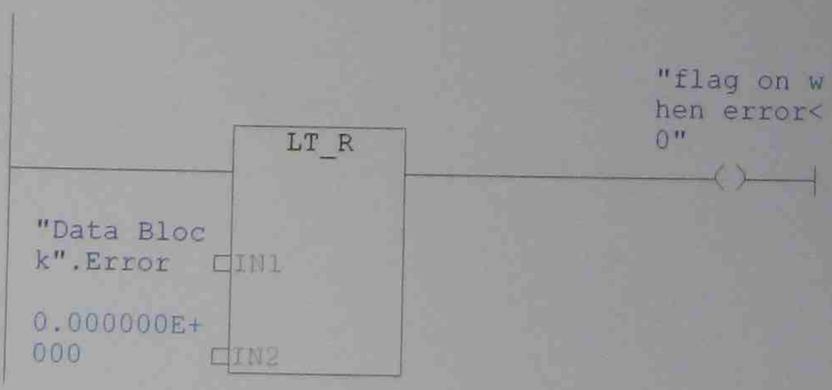
Network: 23

Increment Integral action



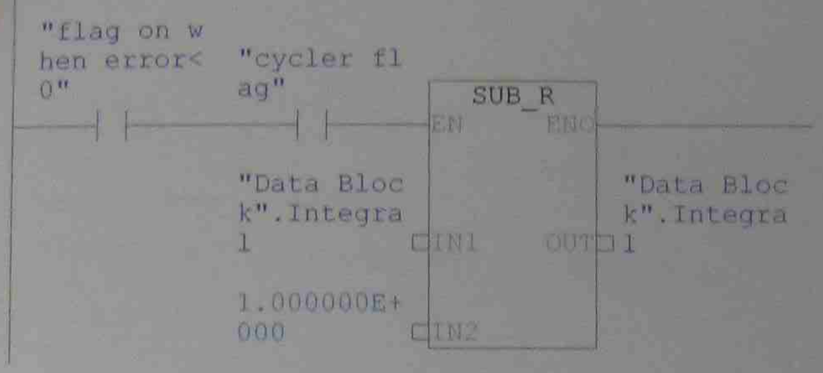
Network: 24

test if error < 0



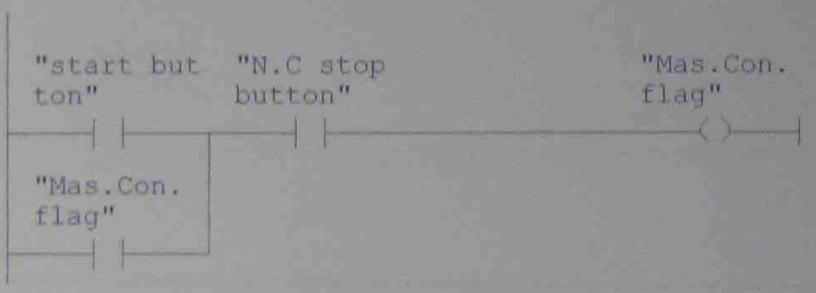
Network: 25

decrement integral action



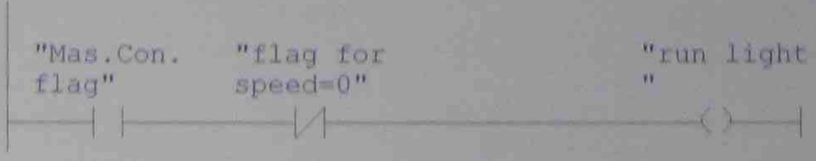
Network: 26

start stop station



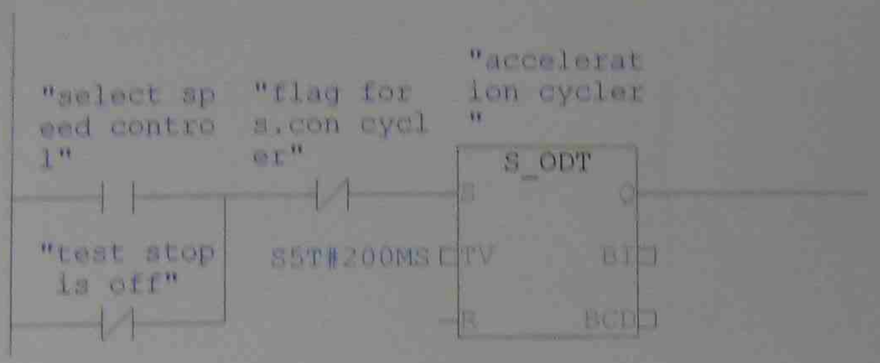
Network: 27

turn on run light



Network: 28

cycler for pushbutton speed control: if pushbutton speed control selected, start
cycling, if stop button pressed, start cycling.

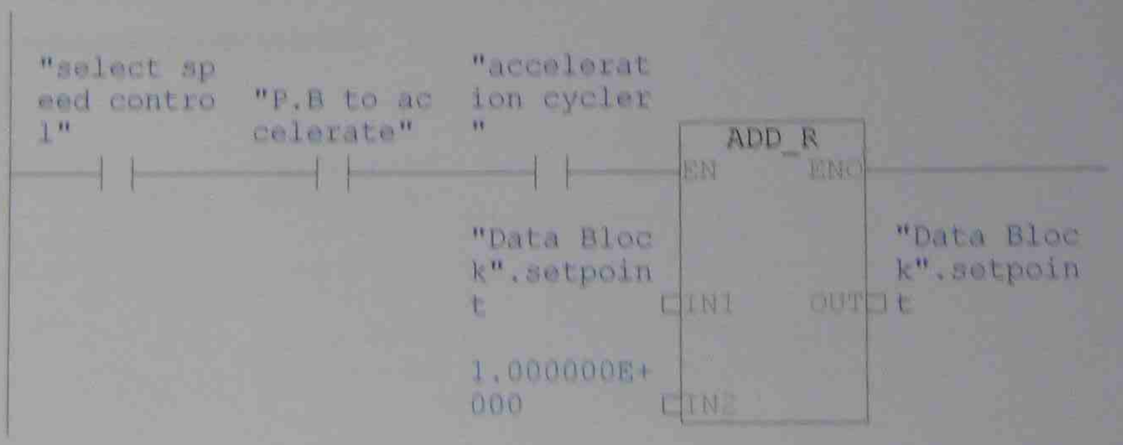


Network: 29

cycler for pushbutton speed control

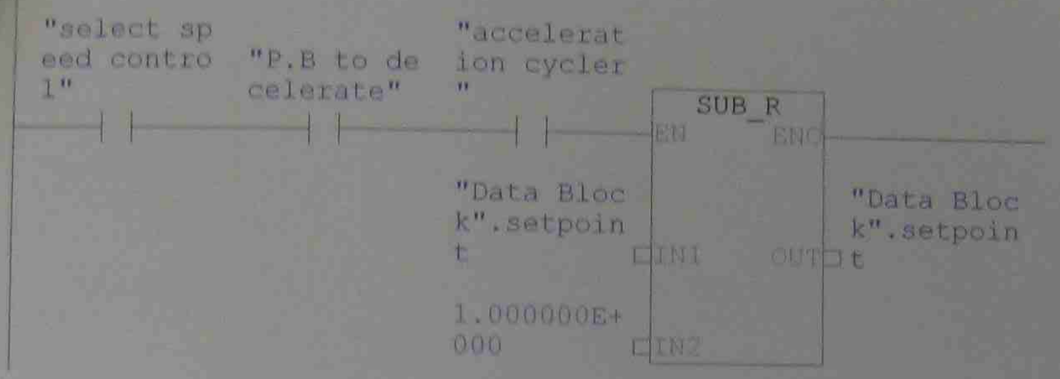
Network: 30

accelerate motor



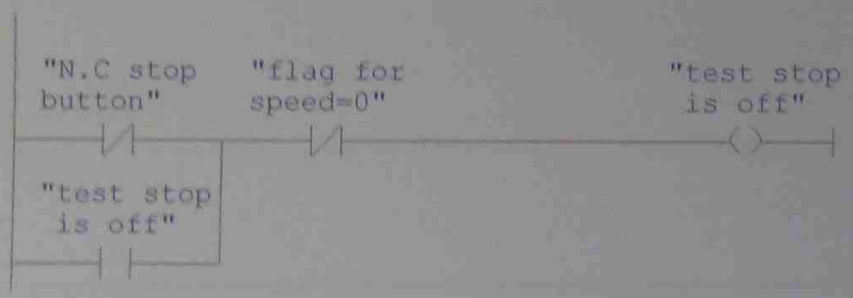
Network: 31

decelerate motor



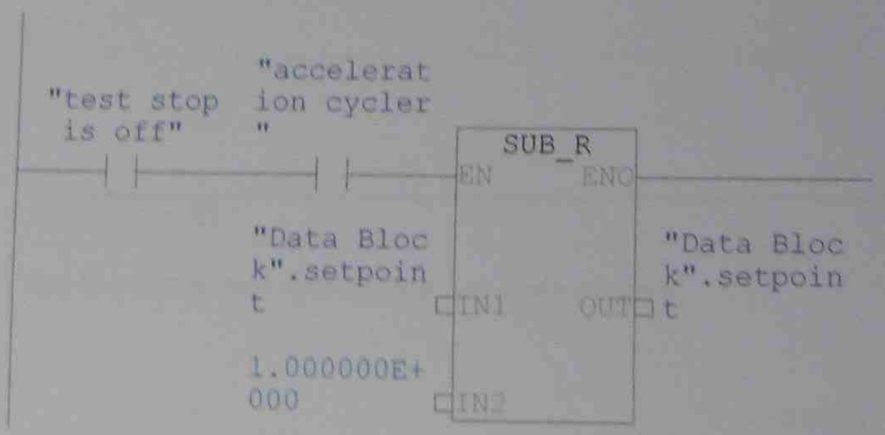
Network: 32

stop button control



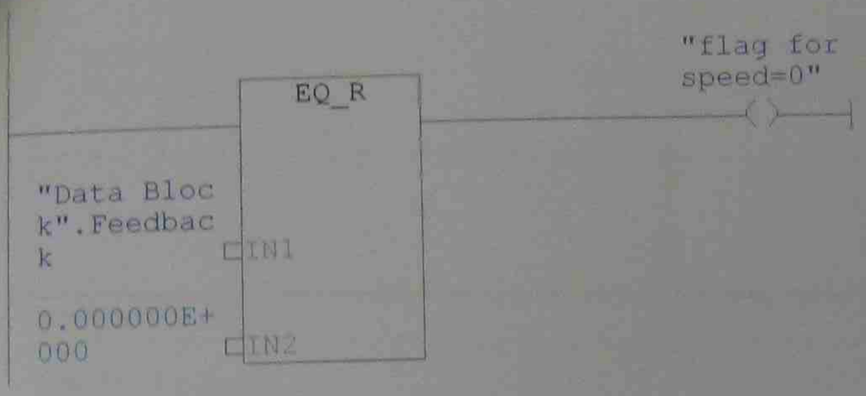
Network: 33

decererate motor to a halt



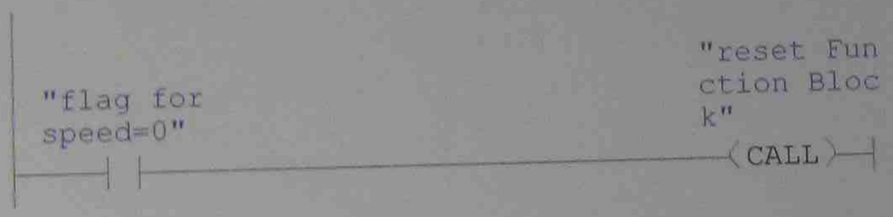
Network: 34

test to see if speed is 0



Network: 35

jump to reset function block

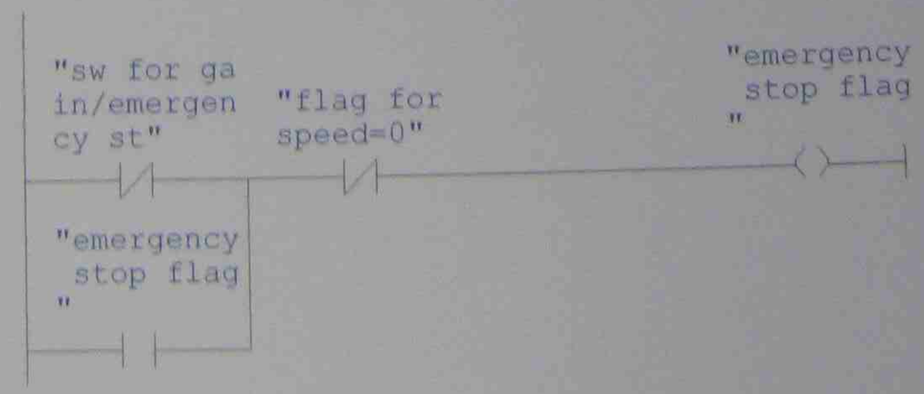


Network: 36

jump to reset function block switch for gain shared with emergency stop

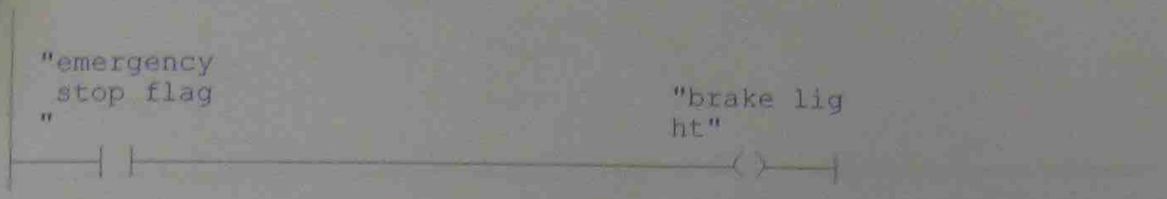
Network: 37

flag to indicate emergency stop has been pressed



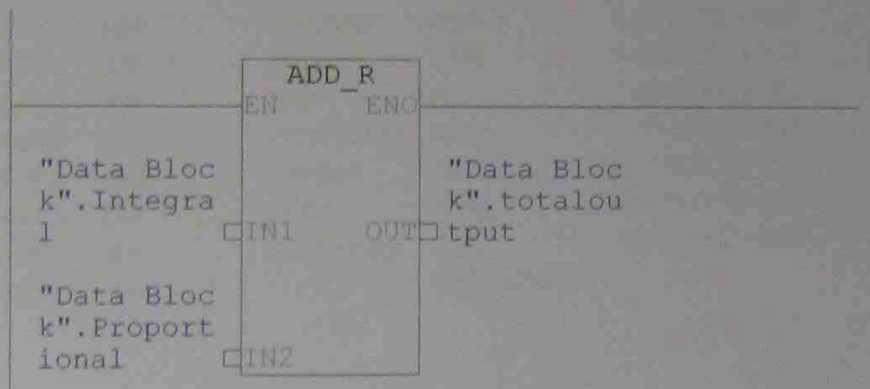
Network: 38

Turn on Brake light



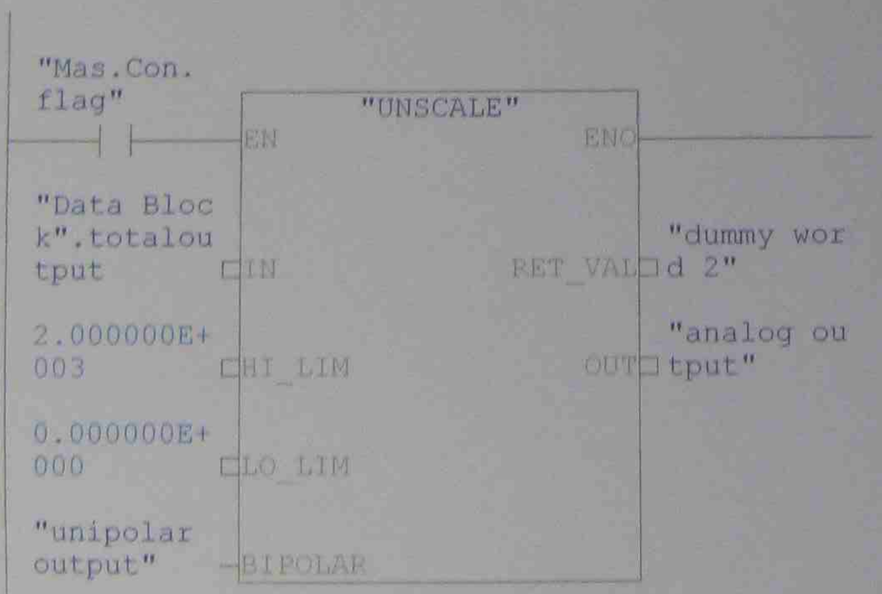
Network: 39

total P + I output



Network: 40

analog output



:Data Block

Name:
Author:
Family:
Version: 1.0
Code version: 2

Time stamp Code: 11/06/09
Interface: 28/05/09
Lengths Block: 00136
Code: 00032
Data: 00026

Block: DB10

Address	Declaration	Name	Type	Start value	Comment
0.0	stat		STRUCT		
+0.0	stat	setpoint	REAL	0.000000E+000	
+4.0	stat	Feedback	REAL	0.000000E+000	
+8.0	stat	Error	REAL	0.000000E+000	
+12.0	stat	gain	REAL	0.000000E+000	
+16.0	stat	Proportional	REAL	0.000000E+000	
+20.0	stat	Integral	REAL	0.000000E+000	
+24.0	stat	DW7	REAL	0.000000E+000	
+28.0	stat	totaloutput	REAL	0.000000E+000	
=32.0	stat		END_STRUCT		

nbol table

tatus	Symbol	Address	Data Type	Comment
	CYCL_EXC	OB 1	OB 1	Cycle Execution
	SCALE	FC 105	FC 105	Scaling Values
	unipolar input	M 1.0	BOOL	
	dummy word	MW 10	WORD	
	Feedback(analog)	PIW 752	WORD	
	switch for setpoint	I 126.0	BOOL	
	setp. one-shot flag1	M 0.0	BOOL	
	setp. one-shot flag2	M 0.1	BOOL	
	thumbwheel	IW 124	WORD	
	Convert 16 BCD - Bin	FC 81	FC 81	Convert BCD to 16 bit binary
	not used	M 244.7	BOOL	
	not used 2	MB 242	BYTE	
	not used 3	MW 238	WORD	
	not used 4	MW 244	WORD	
	not used 5	MW 246	WORD	
	not used 6	C 0	COUNTER	
	setpoint(binary)	MW 40	WORD	
	sw for gain/emergency st	I 126.1	BOOL	
	gain one-shot flag1	M 0.2	BOOL	
	gain one-shot flag2	M 0.3	BOOL	
	dummy flag	M 1.3	BOOL	
	gain(binary)	MW 60	WORD	
	right two thumbwheel dig	MW 58	WORD	
	gain(double int)	MD 64	DWORD	
	UNSCALE	FC 106	FC 106	Unscaling Values
	Data Block	DB 10	DB 10	
	unipolar output	M 1.2	BOOL	
	dummy word 2	MW 20	WORD	
	analog output	PQW 752	WORD	
	switch for integral time	I 126.2	BOOL	
	int.timeone-shot flag1	M 0.4	BOOL	
	int.timeone-shot flag2	M 0.5	BOOL	
	dummy flag 2	M 1.4	BOOL	
	Int.time in binary	MW 74	WORD	
	Int.time double int	MD 78	DWORD	
	Int.time real	MD 82	DWORD	
	cycler flag	M 0.7	BOOL	
	dummy flag 3	M 1.5	BOOL	
	Int. Action cycler	T 1	TIMER	
	flag on when error<0	M 3.0	BOOL	
	flag on when error>0	M 0.6	BOOL	
	not used 8	MW 24	WORD	
	setpoint as double int	MD 44	DWORD	
	not used 9	MW 16	WORD	
	status of int. cycler	M 2.0	BOOL	

Symbol table 2009-06-28 19:46:33

Symbol	Address	Data Type	Comment
start button	I 126.3	BOOL	
N.C stop button	I 126.4	BOOL	
Mas.Con. flag	M 2.2	BOOL	
run light	Q 124.0	BOOL	
select speed control	I 126.5	BOOL	
flag for s.con cycler	M 2.3	BOOL	
acceleration cycler	T 2	TIMER	
P.B to accelerate	I 126.6	BOOL	
P.B to decelerate	I 126.7	BOOL	
test stop is off	M 2.5	BOOL	
flag for speed=0	M 2.6	BOOL	
reset Function Block	FC 1	FC 1	
brake light	I 124.1	BOOL	
emergency stop flag	M 3.1	BOOL	

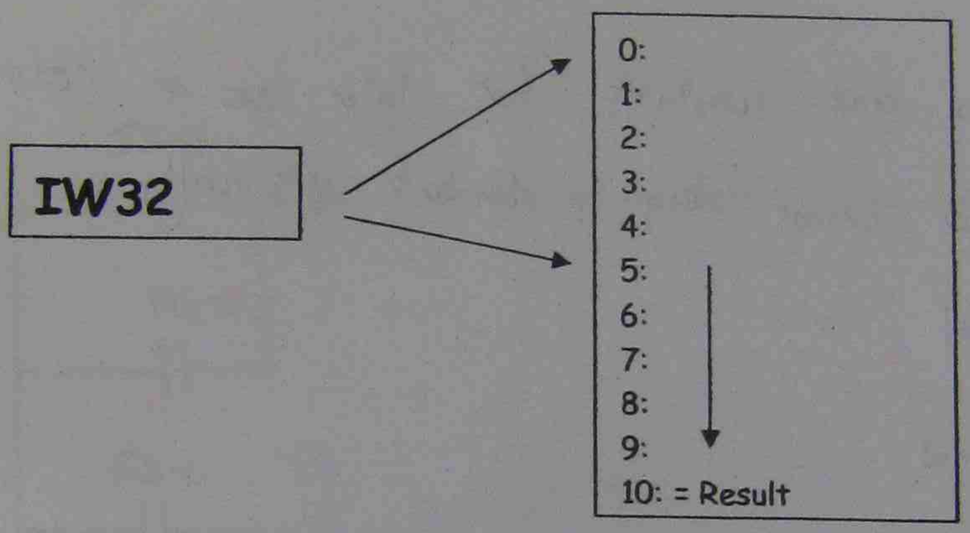
Name : Nicholas Bakirtzidis

Practical 5
Data Blocks

Aim :

to get students to create a data block and monitor it's contents.

- A series of numbers are to be entered into a data block from data word 0 to data word 5 the numbers will then be added together and the result will be displayed at data word 10.



- Use 6 "one shots" to load the values from the thumbwheel switch . take note that the thumbwheel switch's number format is hex.
- Use a one shot to execute the equation
- DW10 = $\frac{(DW0 + DW1 + DW2 + DW3) - (DW4 + DW5)}{}$
- Use a BB (Picture block) to display the contents of your data block .
- Use Data block 40

Questions :

1) What are some of the causes of a PLC going into stop mode ? You should come up with at least 3
2) How can a data block be created automatically in the PLC , give an example , hint : look in the help file of the software for more info.
3) What is the function of DB0 and DB1 ?
4) Explain the purpose of data blocks .
5) Data blocks can only be programmed in STL , true or false , explain your answer.
6) What number format was the result at data word 10 ?
7) What number system does the PLC work in ?
8) What is the function of the two accumulators ?
9) Can we transfer to accumulator 2 ? Explain your answer.
10) Sketch a program that will do the same as this question but it will only add data words 0 to 4 and take away data word 5 and display the result at data word 10.

These Questions are to be answered neatly on a separate sheet of paper.

Answers on back.

1) - incorrect program addresses
-
-

2) L KF+10 by using the L command and specifying how
L DB10 data words are needed.
BE

3)

4) To store values and constants for later use or reference

5) True because you cannot design a data block graphically

6) Hexidecimal - KH

7) ~~Binary~~ Hexidecimal

8) To store data for addition/subtraction etc

9) No data is always sent to accumulator 1 and whatever was in acc 1 will be put into accumulator 2.

OB1
A I2.1
AN F0.3
= F0.2
A I2.1
= F0.3
A F0.2
JC PB1
A I2.2
AN F0.5
= F0.4
A I2.2
= F0.5
F0.4

OB1
L IW32
T DW0
OB1
L IW32

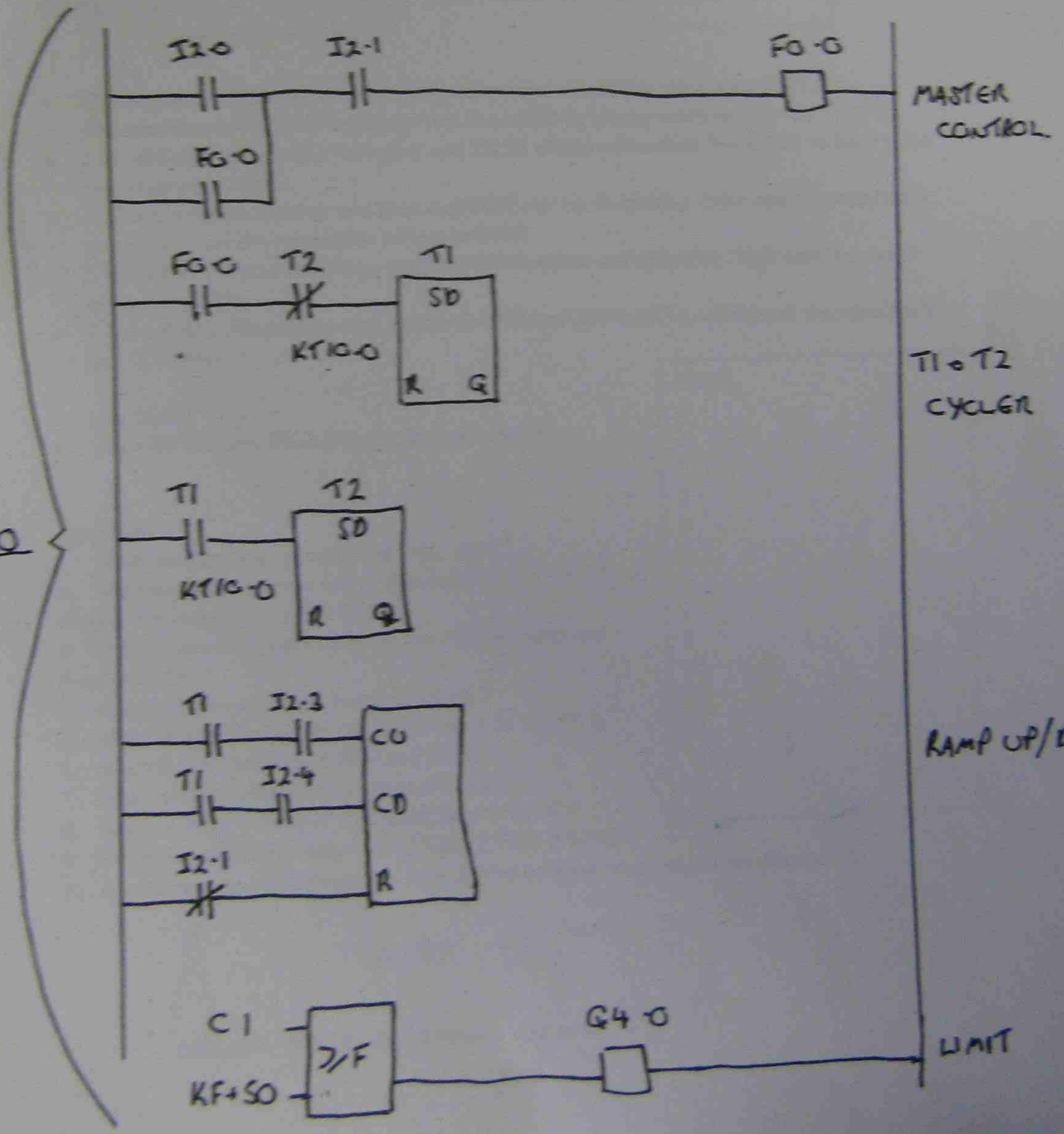
PRAC 6 Nicholas Bakiritzis

Name: File Name:

PART B

Function Block with Formal Operands (Reusable Function Blocks)

Convert the following ladder program into a function block that can be reused many times by converting the program to a function block with variable operands.



TEST THE PROGRAM YOURSELF! HAND IN FOR GRADING.

/40

NAME: Nicholas Baker-Tzedly

FILE NAME: _____

Advanced PLC's assignment 7

- **Data manipulation – Integral Function Blocks**

- I2.0 will activated a one shot circuit , after the one shot is activated IW32 (thumbwheel switch) will be converted from BCD to binary and transferred to DW2
- I 2.1 will activate another one shot and IW32 will be converted from BCD to binary and transferred to DW3
- I2.7 will activate another one shot and DW2 will be divided by DW3 and the result will go to DW4 and the remainder will go to DW5.
- I 2.6 will activate another one shot and DW3 will be multiplied by DW2 and the result will got to DW6
- I 2.4 will activate another one shot and DW6 and DW4 will be added and the result will got to DW10.
- Use DB20
- Use a BB (Picture Block) to display your results.

DB20
0
1
2
3
4
5
6
7
8
10
11
12
13
14

Questions:

1. What number system does the PLC use?
2. Sketch a program that will add decimal +25 to IW32 and display the result at QW32 in BCD.
3. Which data blocks are used by the PLC and what are they used for ?
4. Can you edit an integral function block ?
5. What is the function of the SBCD input to FB240 ?
6. What is the function of FEH on FB243 ?
7. What is the function of Z3=0 on FB 242 ?
8. What is the function of Z31 and Z32 on FB242 ?
9. What format is the answer at Z3 and Z4 for FB243 ?
10. What are formal operands and how do the integral function blocks use them ?

Answers on back

ANSWER THESE QUESTIONS NEATLY !!

1) Hexidecimal

2) L IW32
   L KF25
   +F
   T QW32

3)

10) A formal operand is an operand declared as a name and can be called in a function block. It can be used many times in a plc program since it can have a value put into it many times

4) No because it is already reserved by the plc

5)

Application on Control Technology

* PLC SYSTEM APPLICATIONS. PROJECT #1 TEMPERATURE CONTROL.

David Holbeck
SID: 811383922

applications & TOOLS

Programming Example in Ladder Logic

SIEMENS

Table of Contents

Reference to Automation and Drives Service & Support

This entry is from the internet application portal of Automation and Drives Service & Support. The documentation has the entry ID **23330722**. Click the link below to directly display the download page of this document.

http://support.automation.siemens.com/WW/view/en/23330722

All entries referenced in this document are designated by their entry ID and addressed via the above path.

1 Using FC41 to scale and unscale analog values

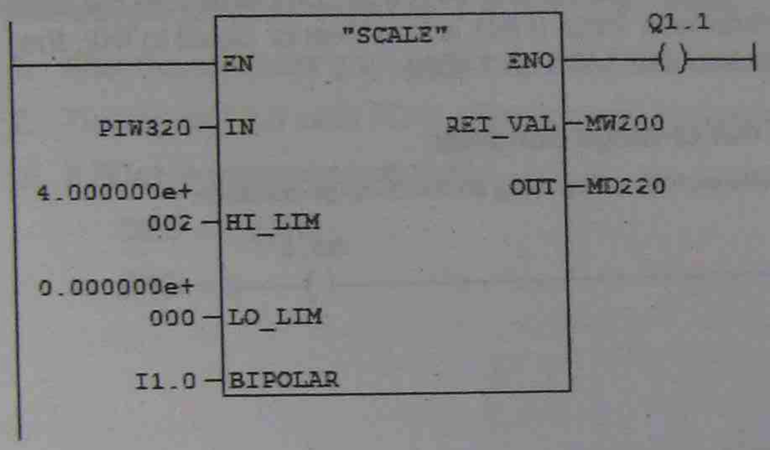
FC41 ("Scaling and Unscaling an Analog Value") provides sample logic for using the Scale function (FC105) and the Unscale function (FC106).

1.1 Using the Scale function (FC105)

Network 1: Scaling the value for an analog input

FC41 calls the Scale function (FC105) to adjust the value of the analog input proportionately between two values (the upper and lower limits).

Figure 1-1: FC41 (Network 1) – Scaling the analog input

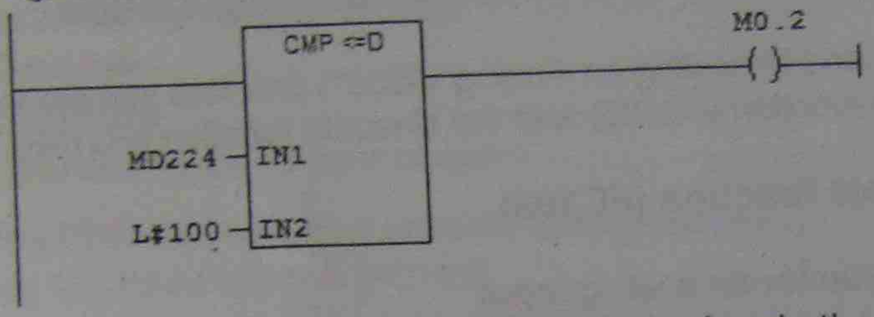


Because there is no contact in the rung, the controller executes FC105 on every scan cycle:

1. FC105 reads the integer value for analog input stored in PIW320 (parameter IN). For this example, the state of I 1.0 (parameter BIPOLAR) determines whether the input value is bipolar or unipolar. Because the value of I 1.0 is 0, FC105 processes the analog input as a unipolar number (0 to 27648)
2. FC105 converts the integer value to a Real number.
3. FC105 scales the Real number to a value between 0.00 (parameter LO_LIM) and 400.00 (parameter HI_LIM):
   - If the function is executed without error, FC105 sets the ENO and Q1.1 to 1 (ON) and sets the RET_VAL (MW200) to W#16#0000 (hexadecimal).
   - If the input integer value is greater than 27648 or less than 0, FC105 sets the output (OUT) to 400.00 (HI_LIM) or 0.00 (LO_LIM), respectively. FC105 also sets the ENO and Q1.1 to 0 (OFF) and returns an error by setting the RET_VAL to W#16#0008 (hexadecimal).
4. FC105 stores the scaled value in MD220 (parameter OUT).

Network 4: Checking for the lower limit of the operating range

Figure 1-4: FC41 (Network 4) – Checking for an out-of-range value (lower limit)

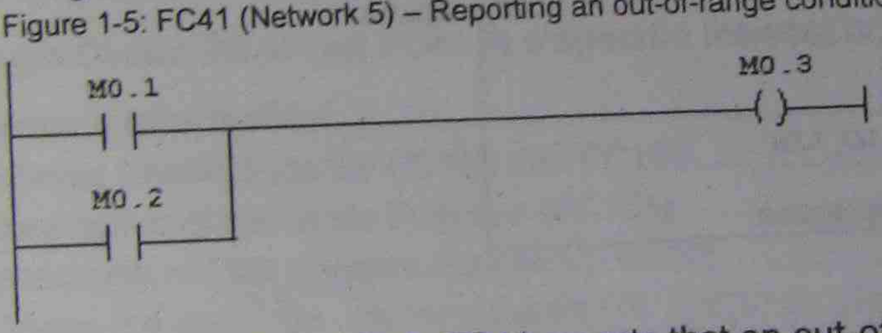


The Compare instruction (CMP <=D) checks whether the scaled value is below the lower limit of the operating range.

On every scan, the Compare instruction (CMP <=D) reads the double-integer value at MD224 (parameter IN1) and compares it to the constant value of 100 (parameter IN2). If IN1 is less than or equal to IN2, the Compare instruction sets M0.1 to 1 (ON).

Network 5: Reporting an out-of-range condition

Figure 1-5: FC41 (Network 5) – Reporting an out-of-range condition



By setting M0.3 to 1 (on), FC41 reports that an out-of range condition has been detected:

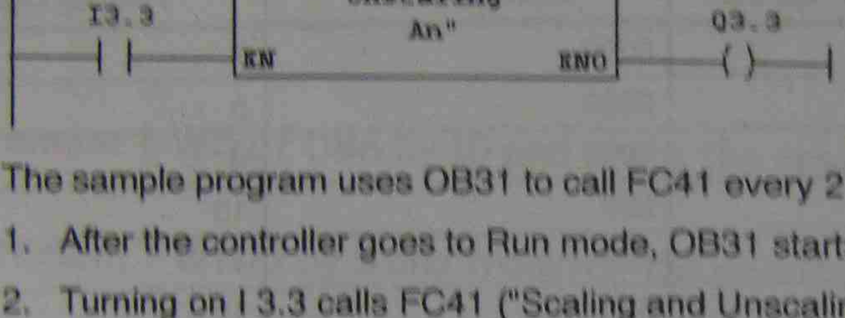
- M0.1 is 1 (ON) if network 3 identified a low out-of-limit value
- M0.2 is 1 (ON) if Network 4 identified a high out-of-limit value

If either M0.1 or M0.2 is 1 (ON), then M0.3 is set to 1 (ON).

2 Using a Cyclic OB to call FC41 at a specific interval

By using a cyclic OB (OB30 to OB38), your program can perform a task or operation at a specific interval.

Figure 2-1: OB31 – Calling FC41



The sample program uses OB31 to call FC41 every 2 seconds (2000 ms):

1. After the controller goes to Run mode, OB31 starts every 2 seconds.
2. Turning on I 3.3 calls FC41 ("Scaling and Unscaling an Analog Input").
3. If FC41 is processed correctly:
   - Q3.3 = 1 (ON)
   - ENO = 1

3.2 Modifying the start-intervals for a Cyclic OB (OB30 to OB38)

The S7 controller uses cyclic interrupt OBs to interrupt the main program scan (OB1) at specified intervals. (The time at which the interval starts is the mode transition from STOP to RUN.) Each of the cyclic interrupt OBs has a default execution interval, as shown in the following table. The execution interval value for each cyclic interrupt OB is configurable.

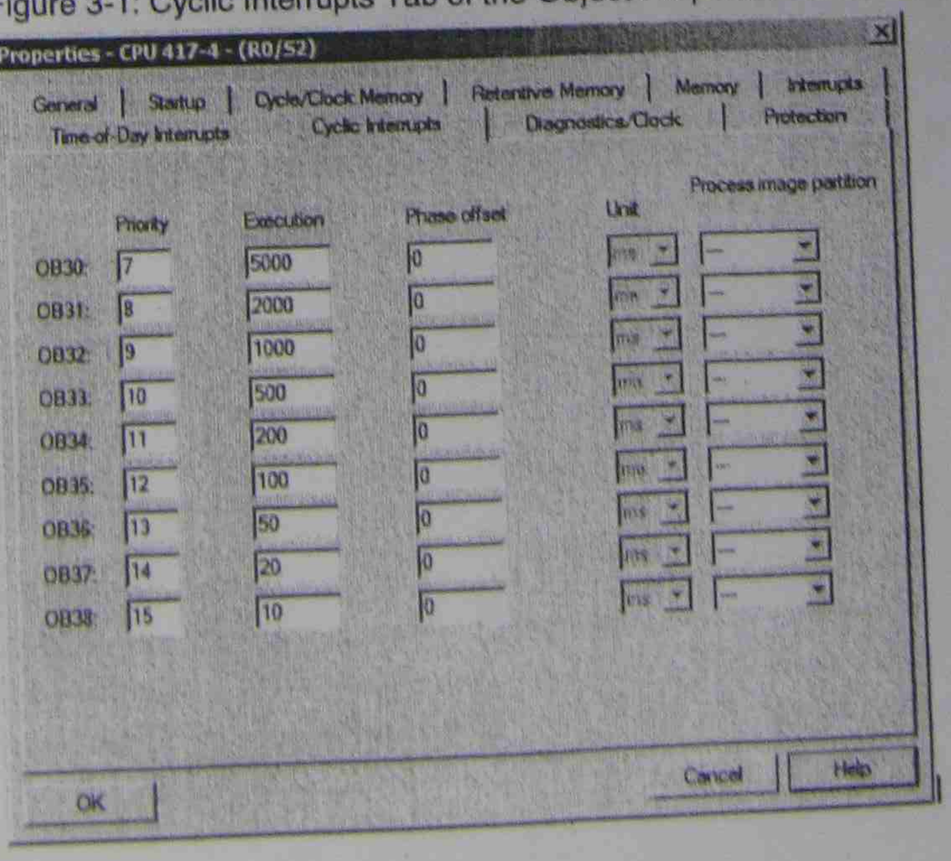
Table 3-1: Cyclic Interrupt OBs

Cyclic Interrupt OB	Interval in ms	Priority Class
OB30	5000	7
OB31	2000	8
OB32	1000	9
OB33	500	10
OB34	200	11
OB35	100	12
OB36	50	13
OB37	20	14
OB38	10	15

When you use STEP 7 to configure the hardware for the project, you can modify the execution rate of cyclic interrupt OBs:

1. Double-click the CPU module to display the CPU Object Properties.
2. Select the Cyclic Interrupts tab.
3. Enter the new execution interval (in milliseconds) and click OK.

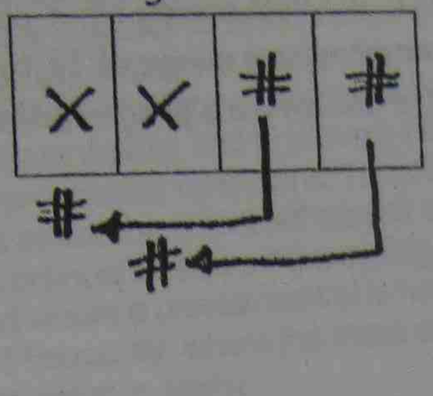
Figure 3-1: Cyclic Interrupts Tab of the Object Properties for the CPU Module



PLC System Applications
Home work Assignment 1

- Write a program that will perform the following tasks
- WRITE A SEPERATE PROGRAM FOR EACH TASK.

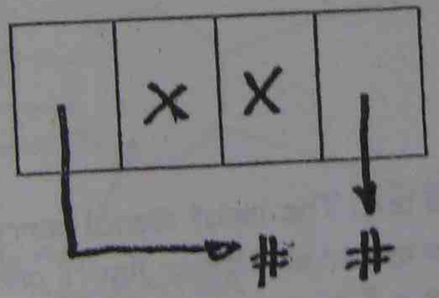
1 Load the thumbwheel at IW 32 and "mask" out the left two digits and shift the right two digits to the left.



2 Increment DW5 of DB6 by 10 and store the result in DW 3

3 Decrement FW100 by 5 and transfer the result to DW2 of DB10

4 Load the thumbwheel at IW 32 and "mask" out the middle two digits and shift the extreme right digit next to the left digit.



5 When the following program is executed which flag will activate output Q4.0 ?

```
L KY 3,2
T FW 100
DO FW 100
A F 0.0
=Q4.0
```

DUE NEXT WEEK

NAME _____

/20

Assignment - Analog Scaling (inputs)

1.
A 1 - 5 Volt signal is to be connected to a 0 - 10 Volt analog input card.
If the 1-5 volt signal corresponded to a temperature range of 12 to 147 degrees Celsius. Calculate the upper and lower limits and write a program that will display the temperature on a 7 segment display.

2. An analog input has a range of 7 to 18 mA. The input signal corresponds to a speed of 0 to 1000 rpm. Calculate the upper and lower limits and write a program that will display the speed on a 7 segment display.

Name:_____ File name:_____

PLC System Applications Temperature control Assignment

On/Off Control

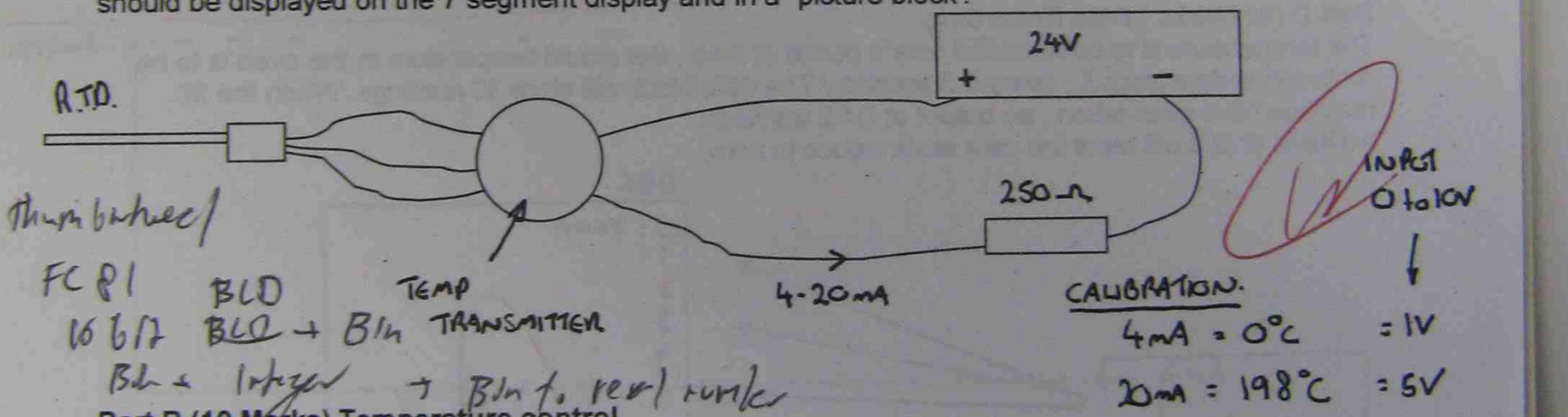
An on-off controller is the simplest form of temperature control device. The output from the device is either on or off, with no middle state. An on-off controller will switch the output only when the temperature crosses the setpoint. For heating control, the output is on when the temperature is below the setpoint, and off above setpoint. Since the temperature crosses the setpoint to change the output state, the process temperature will be cycling continually, going from below setpoint to above, and back below

n cases where this cycling occurs rapidly, and to prevent damage to contactors and valves, an on-off differential, or "hysteresis," is added to the controller operations. This differential requires that the temperature exceed setpoint by a certain amount before the output will turn off or on again. On-off differential prevents the output from "chattering" or making fast, continual switches if the cycling above and below the setpoint occurs very rapidly. On-off control is usually used where a precise control is not necessary, in systems which cannot handle having the energy turned on and off frequently, where the mass of the system is so great that temperatures change extremely slowly, or for a temperature alarm.

Use the analog input to your PLC to perform the following tasks

Part A (10 Marks) Temperature measurement

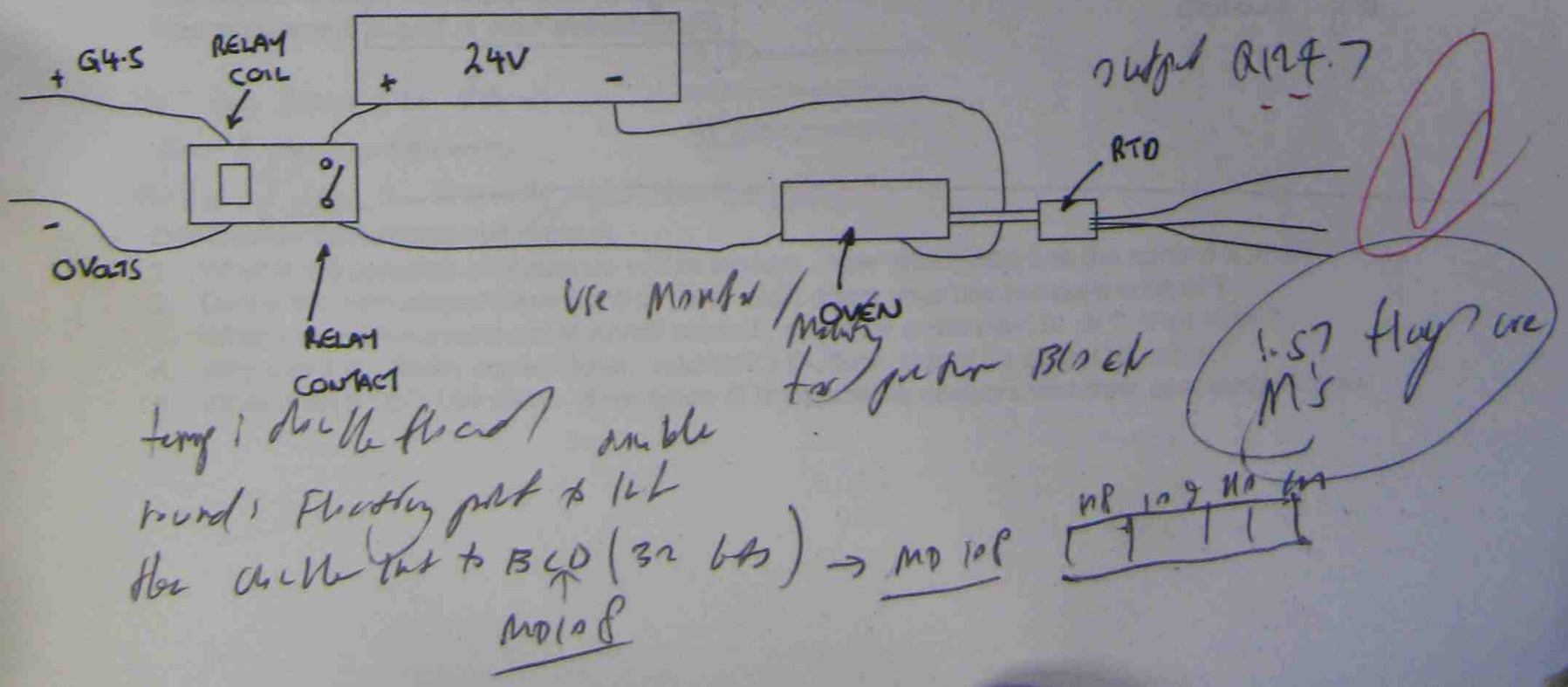
Connect a 4-20mA current loop as shown and measure the temperature in the room. The temperature should be displayed on the 7 segment display and in a "picture block'.



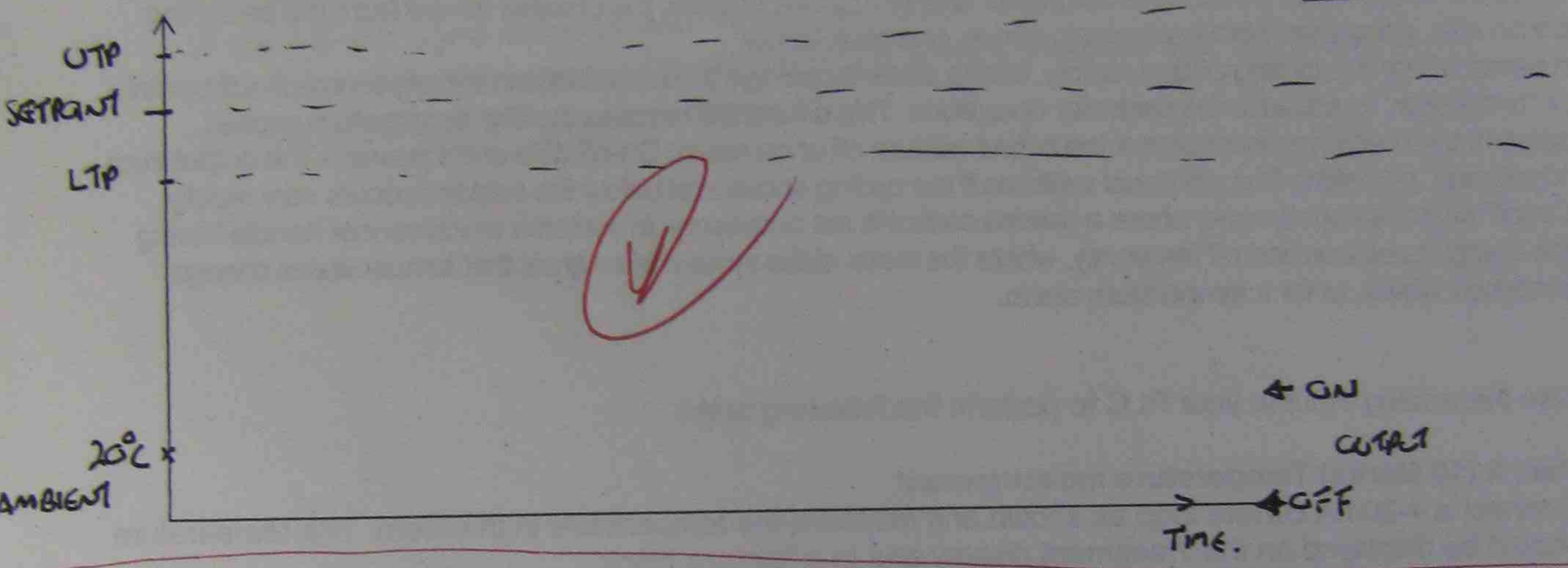
Part B (10 Marks) Temperature control

The temperature in an oven is to be controlled by your PLC , a digital output will control the power to the heater element at Q4.5.

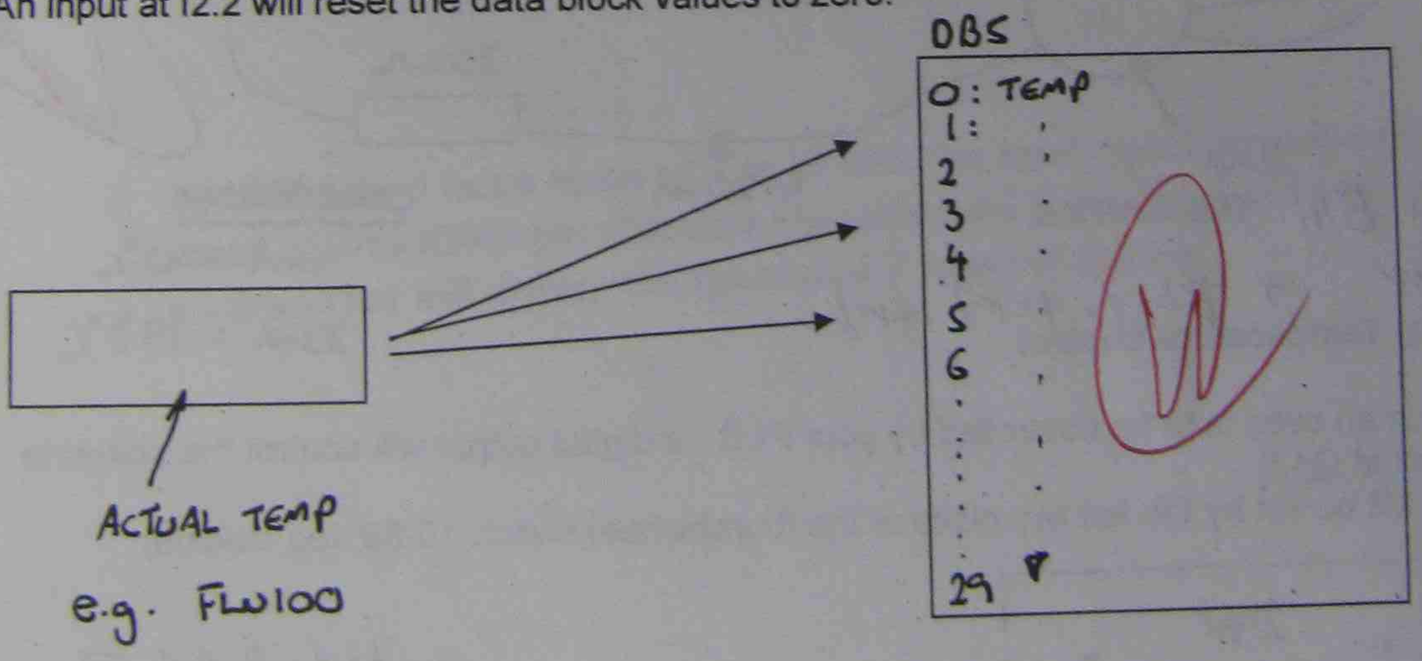
The temperature will be set by the left two digits of the thumbwheel switch. (0-99 deg celcius)



Part C (10 Marks) Hystersis
Add some Hystersis to you your control system , the Hystersis temperature is set by the right two digits of your thumbwheel switch.



Part D (30 Marks) Data Recording
The temperature is to be recorded over a period of time , the actual temperature in the oven is to be recorded in data block 5 , every 0.5 seconds/ The data block will store 30 readings. When the 30 readings have been taken , an output at Q4.2 will flash.
An input at I2.2 will reset the data block values to zero.



Proportional Control

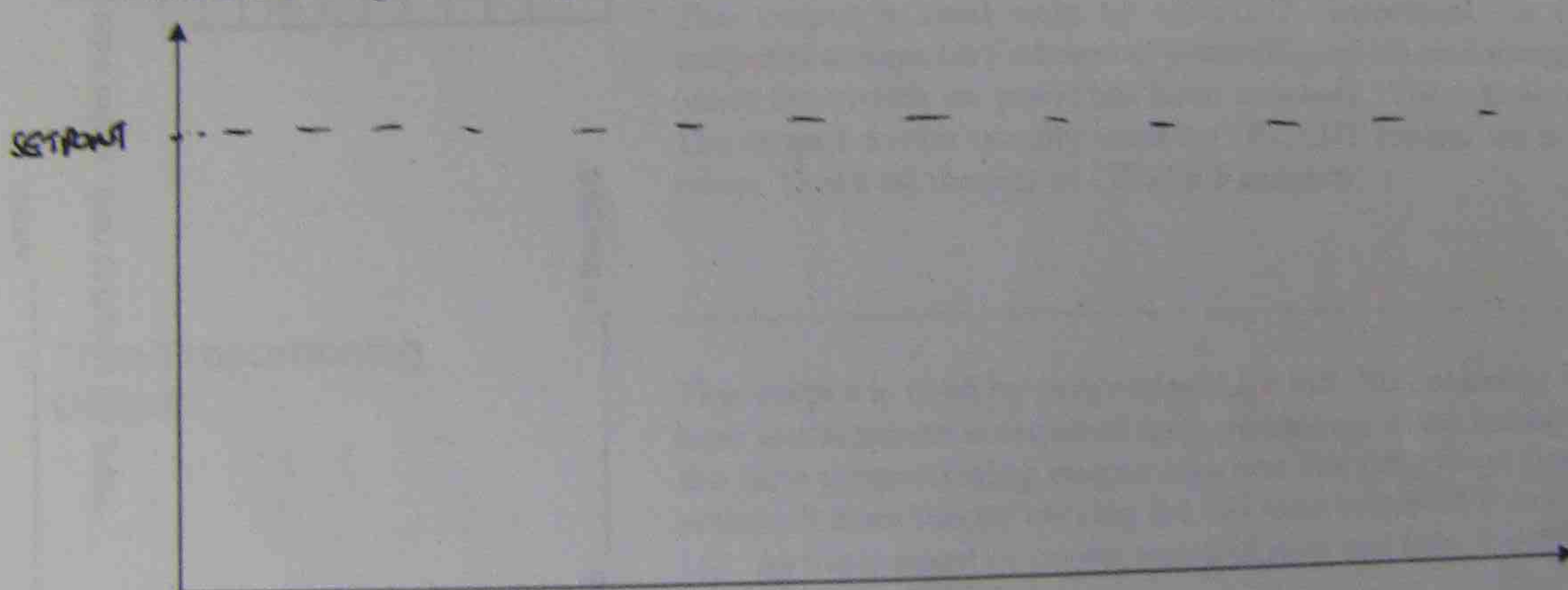
Proportional controls are designed to eliminate the cycling associated with on-off control. A proportional controller decreases the average power supplied to the heater as the temperature approaches setpoint. This has the effect of slowing down the heater so that it will not overshoot the setpoint, but will approach the setpoint and maintain a stable temperature. This proportioning action can be accomplished by turning the output on and off for short time intervals. This "time proportioning" varies the ratio of "on" time to "off" time to control the temperature. The proportioning action occurs within a "proportional band" around the setpoint temperature. Outside this band, the controller functions as an on-off unit, with the output either fully on (below the band) or fully off (above the band). However, within the band, the output is turned on and off in the ratio of the measurement difference from the setpoint. If the temperature is further from the setpoint, the on- and off-times vary in proportion to the temperature difference. If the temperature is below setpoint, the output will be on longer; if the temperature is higher, the output will be off longer.

Part E (30 Marks) Proportional Control.

Remove the Hystersis and 0n/Off control from your program and design a proportional control system that will average the output using pulse width modulation. Therefore controlling the temperature more accurately.

This is a design question and marks will be awarded for accuracy and simplicity of you system.

Use the two right hand digits to set the gain of you proportional control system.



Notes

Use one shots for loading your data from the thumbwheel switches.

Document your program with line comments explaining the function of each part of your program.

Do your own work.

You **MUST** hand in this sheet with your assignment! Don't lose it !

This assignment is part of your assessment.

Use two timers -> on time of timer 1 proportional to error, off time of timer 2 inversely proportional to error

Conclusion Questions (10 marks)

1. What is the purpose of Hystersis in this system , how dies it improve the control if at all ?
2. Define the term proportional control , how did it affect your temperature control ?
3. What causes the overshoot in on/off control , can it be eliminated at all ? If so how ?
4. Why use a 4 - 20mA current loop , wouldn't a 0-20mA signal be easier to scale ?
5. What is an RTD ? List some other types of temperature sensors and their operating ranges.

PLC System Applications

Name :_____

Temperature Proportional Control reading Part E Practical 1

Record the temperature in your oven every 30 seconds in the table below , then complete the graph

Hand this in with your file

Reading #	Temp	Reading #	Temp	Reading #	Temp
1		11		21	
2		12		22	
3		13		23	
4		14		24	
5		15		25	
6		16		26	
7		17		27	
8		18		28	
9		19		29	
10		20		30	

Setpoint = _____ deg

Temp ↑

Control Outputs

As we know, the control block uses ON/OFF or PID functions to "decide" how much power the heater should get. At this point, we need to explore how the control output converts control block commands into action.

The control output is typically connected to an output device. This output device may be mounted inside the controller if small enough. Or it may be an external device. The job of the **control output** is to *signal the output device.* The job of the **output device** is to switch the power (electric current) to the heater. So, depending on what the control output signal "says," the output device switches the heater ON and OFF accordingly.

Since the control output and output device work so closely together, they are often simply referred to as the "output." We can separate outputs into three categories: **ON/OFF output, time proportioning output** and **process output**. We examine these next.

ON/OFF Output

This output is used **only** by ON/OFF controllers. In ON/OFF control, the output is always ***OFF above set point*** (Figure 15) and always ***ON below set point*** (once the switch on point has been crossed). There is no proportioning action. The output device usually used by ON/OFF controllers is an electromechanical relay. That's all there is to ON/OFF outputs!

Time Proportioning Output

This output is used by proportional (or full PID controls). First, the PID decides how much power is required (as a percentage of the heater's total power). Then the time proportioning output converts this percentage power requirement into action. It does this by varying the ON time versus OFF time of the output (Figure 16). As the amount of power required rises and falls, the ON time rises or falls in relation to the OFF time.

Note in Figure 16, that the output is always switched (or cycled) ON and OFF inside the proportional band. However, the Rate function may override this and switch (or cycle) the output ON and OFF above (or below) the proportional band.

Figure 15
ON/OFF Output

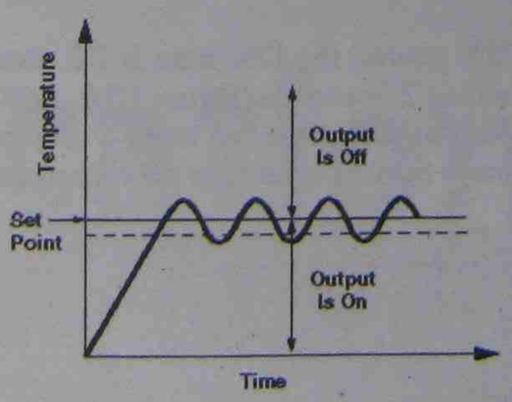
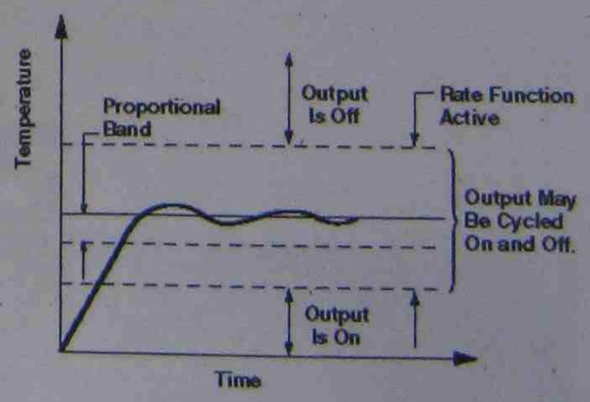


Figure 16
Time Proportioning Output



If 25% power is required by the controller, for example, the ON time will be ⅓ rd the OFF time. This allows the heater to produce 25% or ¼ th of its power. The ON time plus OFF time is equal to the "cycle time." Let's explore cycle time and its impact on a time proportioning output.

Time Proportioning Output (con't)

Duty cycle is a term which describes the amount of power the heater is supplying to the work load. It is calculated by dividing the ON time by the total cycle time. For example, an ON time of 3 seconds during a 10 second cycle time means a duty cycle of 30%,

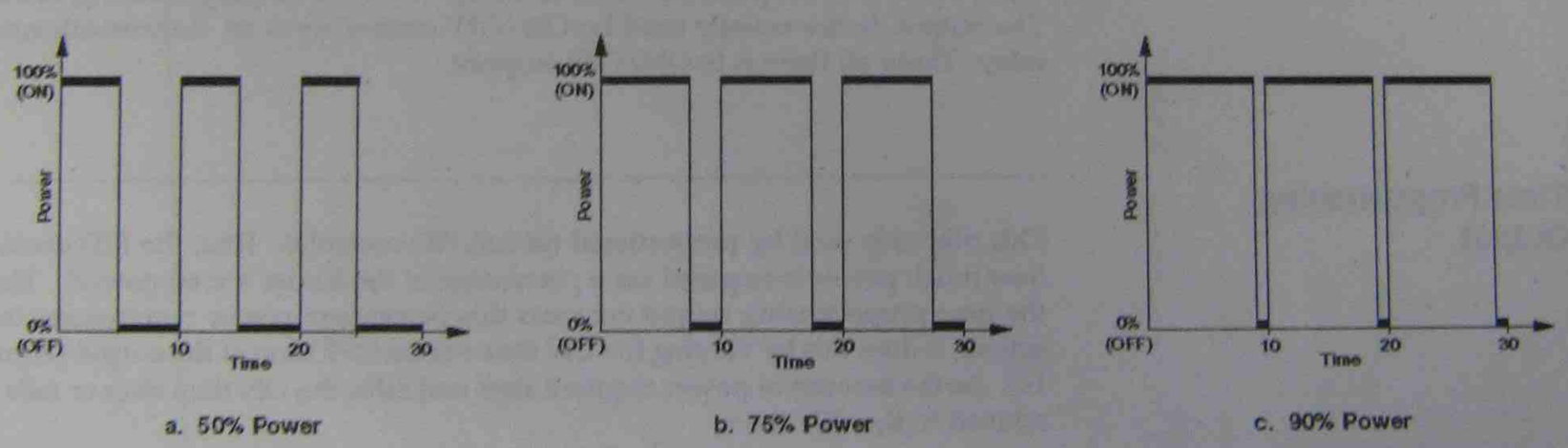
Cycle Time

The cycle time basically sets the time period within which a control output must switch the heater ON, switch it OFF and switch it back ON again. The cycle time is a fixed value (like 5 or 10 seconds) in some controls, but is adjustable in most of them. The amount of ON time versus the total cycle time is equal to the power level (or duty cycle) which the PID requires the output to provide. Let's work through a few examples to see how this is applied.

Example: A PID controller decides that a 50% power level (or duty cycle) is required. The time proportioning output is set for a cycle time of 10 seconds. How long is the output switched ON and how long is it switched OFF during each 10 second cycle?

At a 50% power requirement, the control output signals the output device to switch ON for 5 seconds (50% of 10 seconds). Thus, the ON time is 5 seconds (Figure 17a). The output then switches OFF for the remaining 5 seconds. So the OFF time is also 5 seconds.

Figure 17
ON and OFF Times Based on a 10 Second Cycle Time



a. 50% Power b. 75% Power c. 90% Power

Example: At various times, the PID control decides that 75% and 90% power levels (duty cycles) are required. A 10 second cycle time is used. How long will the output be ON during the 10 second cycle time in each case?

At 75% power, the ON time is 7.5 seconds (75% of 10). The OFF time is the remaining 2.5 seconds (Figure 17b). At 90% power, the ON time is 9 seconds (90% of 10). The OFF time is 1 second (Figure 17c). Do you notice how the ON time increases as the percentage power required increases? Now it's your turn!

Exercise One

The cycle time of a time proportioning output is 4 seconds. What are the ON and OFF times for 50% and 75% power requirements? Draw graphs in the left margin to illustrate this. Explain the similarities to Figures 17a and 17b.

OB1:CYCL_EXC

Cycle Execution

Name:		Time stamp	Code:	27/06/09
Author:			Interface:	20/01/04
Family:		Lengths	Block:	00304
Version:	0.0		Code:	00172
Code version:	2		Data:	00028

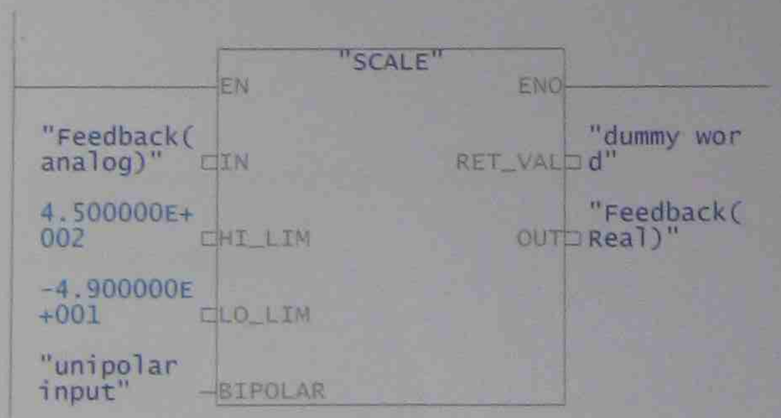
Block: OB1 "Main Program Sweep (Cycle)"

Analog input, display temperature in 7-segment display, perform On/Off control

Address	Declaration	Name	Type	Start value	Comment
0.0	temp	OB1_EV_CLASS	BYTE		Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
1.0	temp	OB1_SCAN_1	BYTE		1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
2.0	temp	OB1_PRIORITY	BYTE		1 (Priority of 1 is lowest)
3.0	temp	OB1_OB_NUMBR	BYTE		1 (Organization block 1, OB1)
4.0	temp	OB1_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB1_RESERVED_2	BYTE		Reserved for system
6.0	temp	OB1_PREV_CYCLE	INT		Cycle time of previous OB1 scan (milliseconds)
8.0	temp	OB1_MIN_CYCLE	INT		Minimum cycle time of OB1 (milliseconds)
10.0	temp	OB1_MAX_CYCLE	INT		Maximum cycle time of OB1 (milliseconds)
12.0	temp	OB1_DATE_TIME	DATE_AND_TIME		Date and time OB1 started

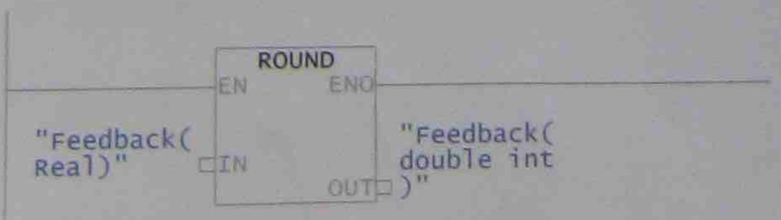
Network: 1

Scaling function block, Scale analog value between -4.9 and 540



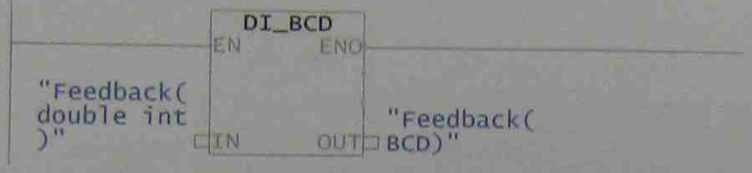
Network: 2

converts temperature as a real no. to a double int.



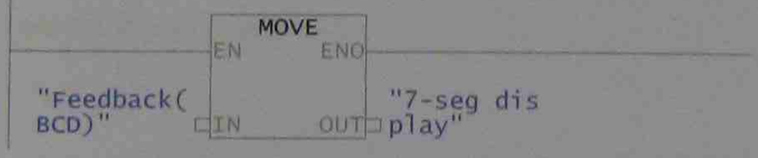
Network: 3

converts doubke integer Temperature to BCD



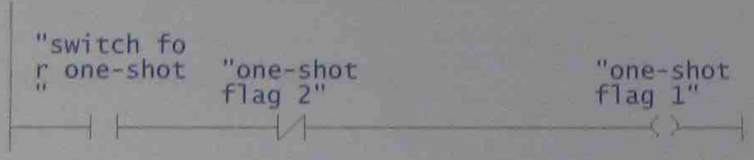
Network: 4

double BCD to 7-segment display



Network: 5

one shot for setpoint

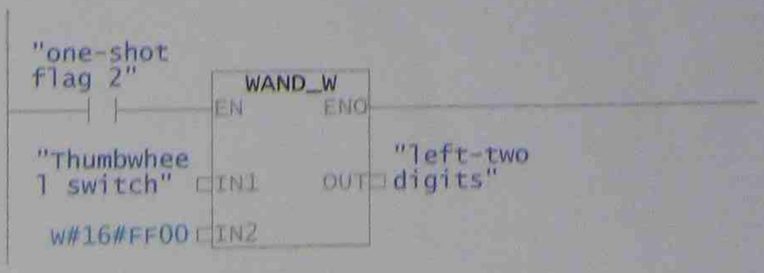


Network: 6

one shot for setpoint

Network: 7

get thumbwheel setpoint data

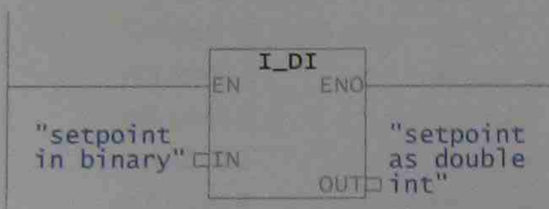


Network: 8

setpoint BCD - binary

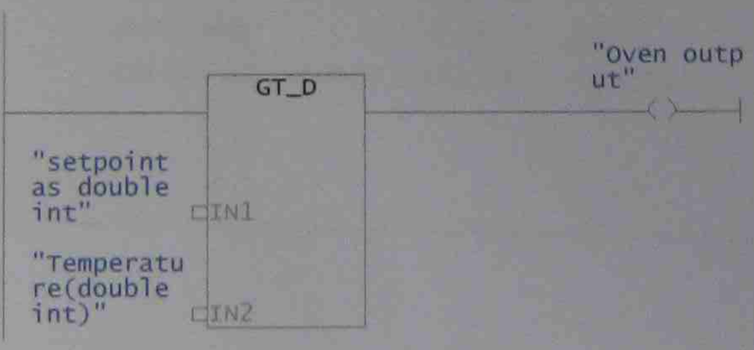
Network: 9

convert setpoint to double int



Network: 10

compare setpoint with actual temp.



Symbol table

Status	Symbol	Address	Data Type	Comment
	CYCL_EXC	OB 1	OB 1	Cycle Execution
	SCALE	FC 105	FC 105	Scaling Values
	unipolar input	M 1.0	BOOL	
	dummy word	MW 40	WORD	
	Feedback(Real)	MD 100	DWORD	
	Feedback(analog)	PIW 752	WORD	
	Feedback(double int)	MD 116	DWORD	
	Feedback(BCD)	MD 122	DWORD	
	7-seg display	QB 125	BYTE	
	Thumbwheel switch	IW 124	WORD	
	left-two digits	MW 150	WORD	
	switch for one-shot	I 126.0	BOOL	
	one-shot flag 1	M 0.0	BOOL	
	one-shot flag 2	M 0.1	BOOL	
	Convert 16 BCD - Bin	FC 81	FC 81	Convert BCD to 16 bit binary
	not used 1	M 244.7	BOOL	
	not used 2	MB 242	BYTE	
	not used 3	MW 238	WORD	
	not used 4	MW 244	WORD	
	not used 5	MW 246	WORD	
	not used 6	C 0	COUNTER	
	dummy flag	M 0.3	BOOL	
	setpoint in binary	MW 155	WORD	
	setpoint in BCD	MW 149	WORD	
	Oven output	Q 124.7	BOOL	
	setpoint as double int	MD 160	DWORD	

OB1:CYCL_EXC

Cycle Execution

Name:		Time stamp	Code:	27/06/09
Author:			Interface:	20/01/04
Family:		Lengths	Block:	00526
Version:	0.0		Code:	00356
Code version:	2		Data:	00028

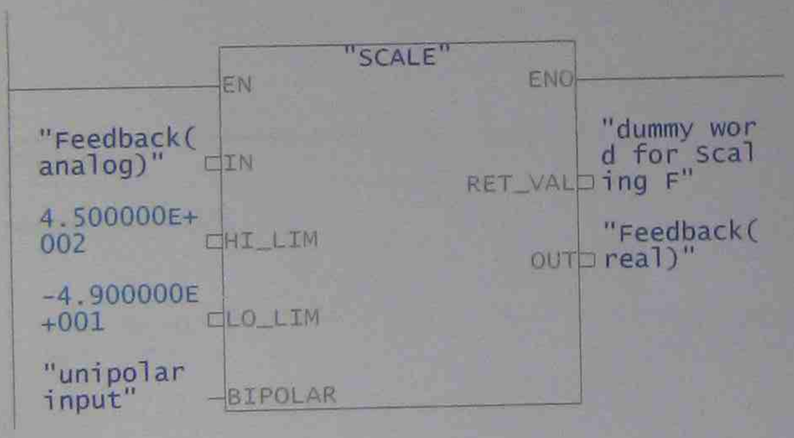
Block: OB1 "Main Program Sweep (Cycle)"

Analog input, display temperature in 7-segment display, perform Hysteress control, perform data logging

Address	Declaration	Name	Type	Start value	Comment
0.0	temp	OB1_EV_CLASS	BYTE		Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
1.0	temp	OB1_SCAN_1	BYTE		1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
2.0	temp	OB1_PRIORITY	BYTE		1 (Priority of 1 is lowest)
3.0	temp	OB1_OB_NUMBR	BYTE		1 (Organization block 1, OB1)
4.0	temp	OB1_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB1_RESERVED_2	BYTE		Reserved for system
6.0	temp	OB1_PREV_CYCLE	INT		Cycle time of previous OB1 scan (milliseconds)
8.0	temp	OB1_MIN_CYCLE	INT		Minimum cycle time of OB1 (milliseconds)
10.0	temp	OB1_MAX_CYCLE	INT		Maximum cycle time of OB1 (milliseconds)
12.0	temp	OB1_DATE_TIME	DATE_AND_TIME		Date and time OB1 started

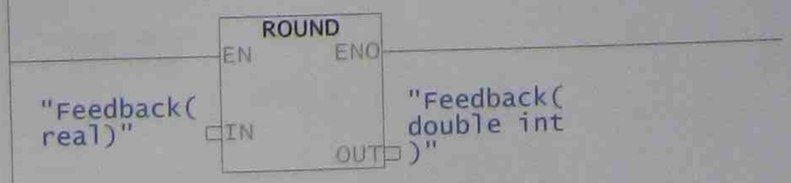
Network: 1

Scaling function block, scale analog value between -4.9 and 450



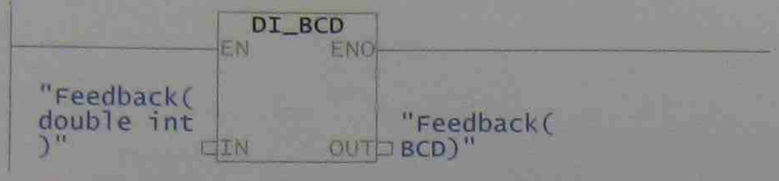
Network: 2

converts double floating point to double int



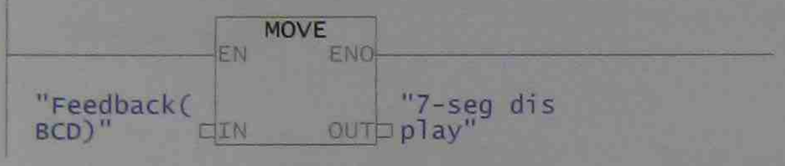
Network: 3

converts doubke integer to BCD



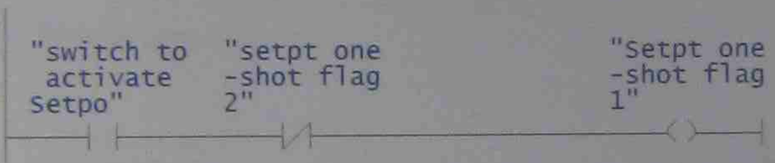
Network: 4

double BCD to 7-segment display



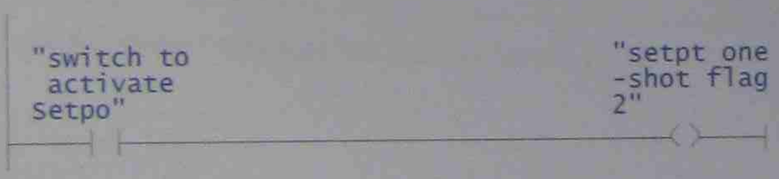
Network: 5

setpoint one shot



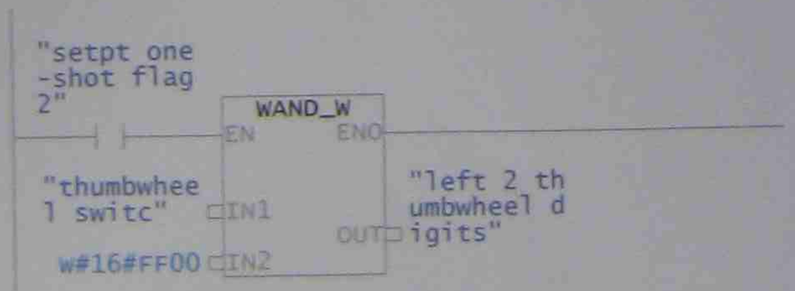
Network: 6

setpoint one shot



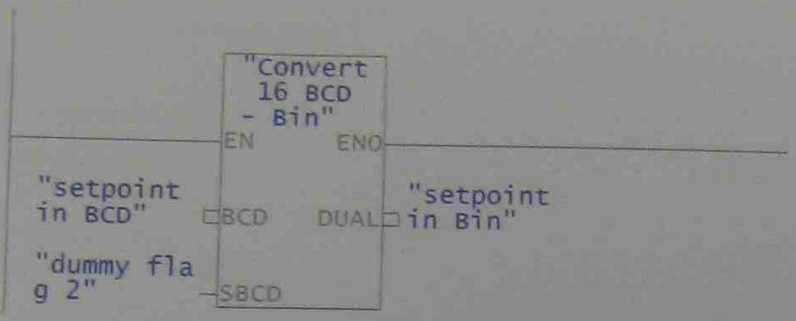
Network: 7

get setpoint from thumbwheel data



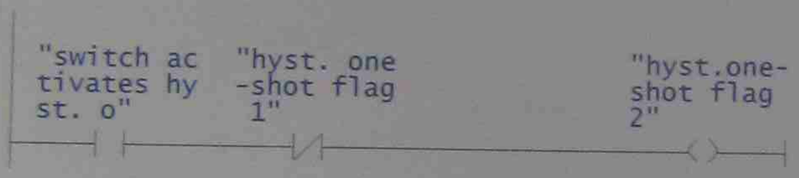
Network: 8

BCD - binary



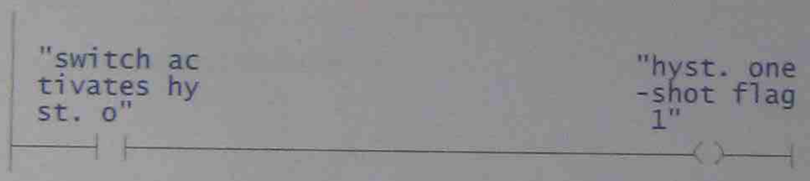
Network: 9

hysteresis one shot



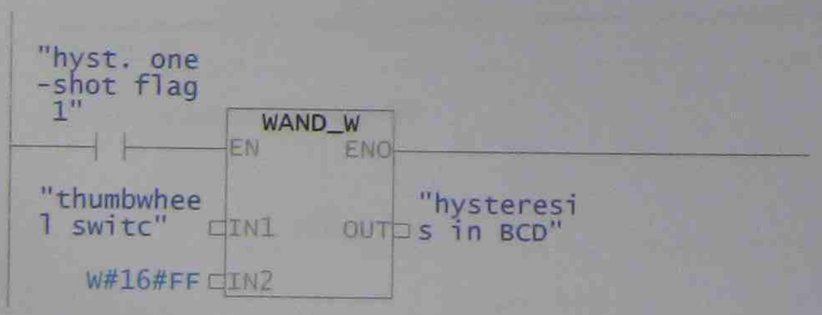
Network: 10

hysteresis one shot



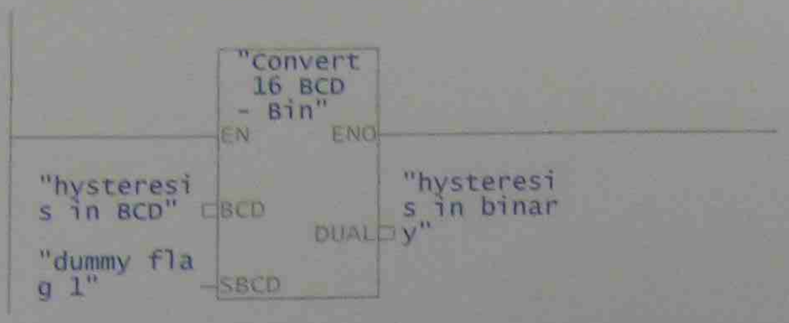
Network: 11

get thumbwheel data right two digits for hysteresis



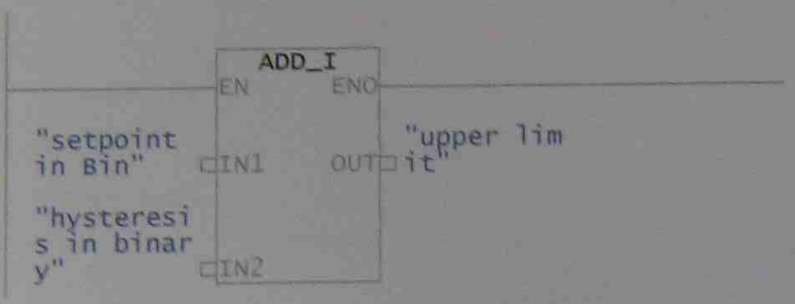
Network: 12

BCD to binary



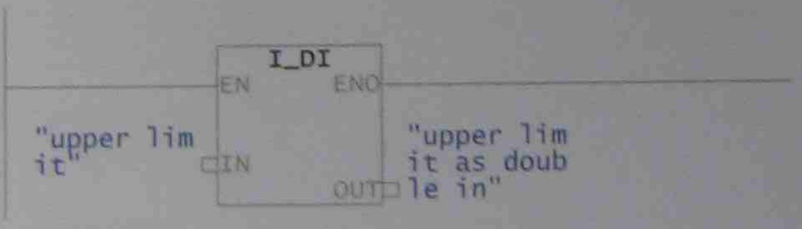
Network: 13

add hysteresis to setpoint



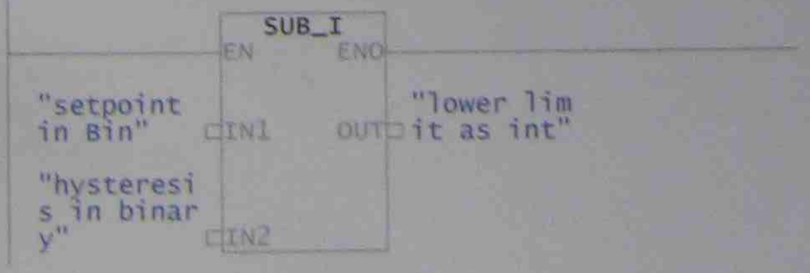
Network: 14

convert uper limit to double int



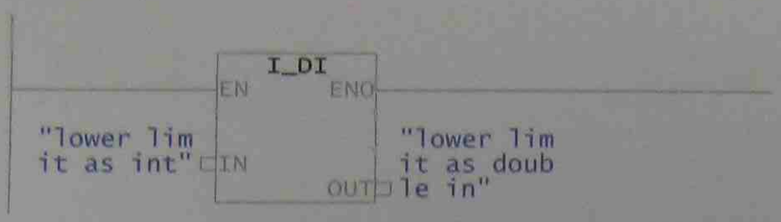
Network: 15

subtract hysteresis from setpoint



Network: 16

convert lower limit to double int

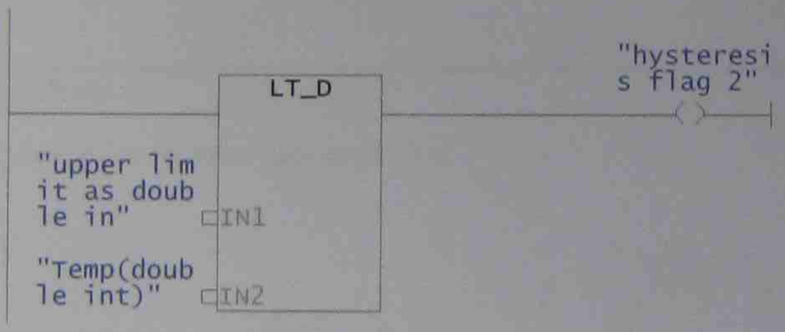


Network: 17

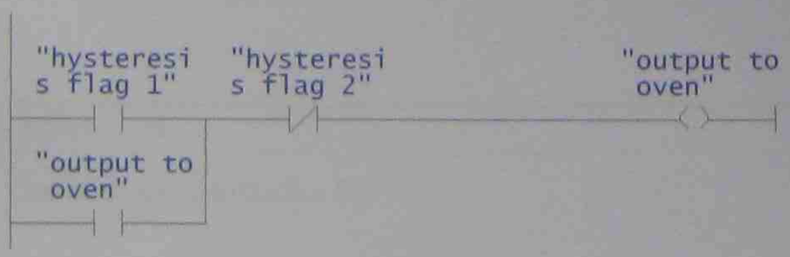
compare lower limit with measured temperature

Network: 18

compare upper limit with temperature

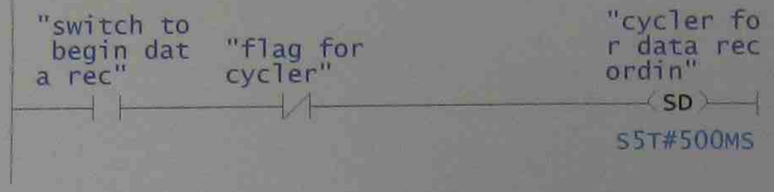


Network: 19



Network: 20

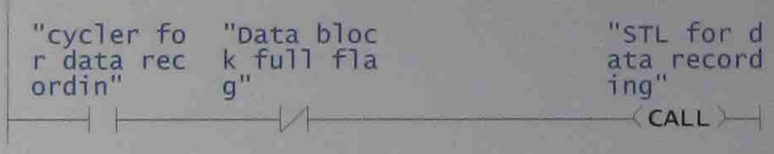
2 second cycler for data recording



Network: 21

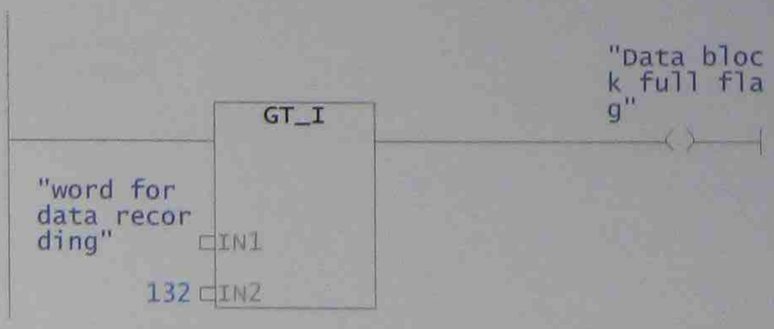
Network: 22

jump to indexed addressing block every 2 seconds



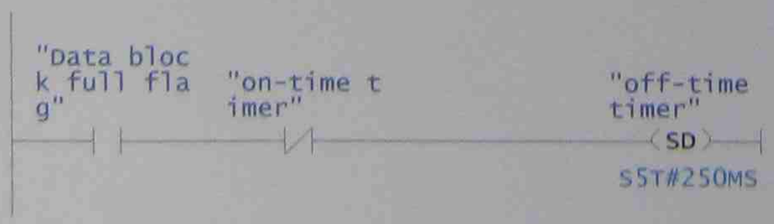
Network: 23

comparator for when data block is full



Network: 24

flasher to say data block is full



Network: 25

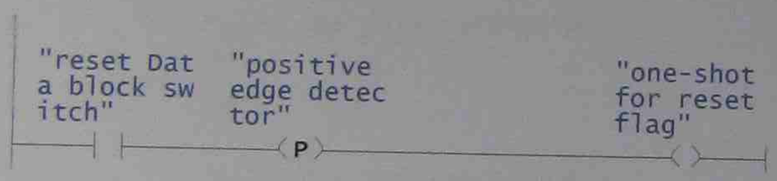
flasher to sat data block is full

Network: 26

flasher to say data block is full

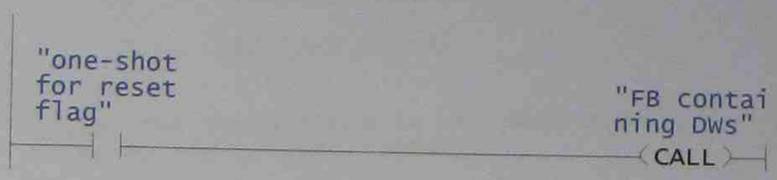
Network: 27

One shot for reset



Network: 28

jump to reset block



FC1:STL for data recording

```
Name:                  Time stamp Code:      27/05/09
Author:                           Interface: 08/04/09
Family:                Lengths    Block:     00140
Version:       1.0                Code:      00046
Code version: 2                   Data:      00008
```

Block: FC1

Address	Declaration	Name	Type	Start value	Comment
	in				
	out				
	in_out				
	temp				

Network: 1

Data Recording

```
      OPN   "data block for data reco"



      L     "word for data recording"
      L     112
      >=I
      JC    here


      L     "word for data recording"
      SLD   3
      LAR1
//load counter number into address reg        //load counter number into
                                               address reg


      L     "Temperature(real)"                //load  temp

      T     DBD [AR1,P#0.0]


//load the temperature to the data word spec   //load the temperature to
ified by the address register                  the data word specified by
                                               the address register


      L     "word for data recording"
      L     4
      +I
      T     "word for data recording"

// Increments the counter word by 4 so it wi   // Increments the counter
l transfer the next available data word        word by 4 so it wil transf
                                               er the next available data
                                               word


      BEU

here: L     0
      T     "word for data recording"
```

FC2:FB containing DWs

```
Name:                  Time stamp Code:      27/05/09
Author:                           Interface: 29/04/09
Family:                Lengths    Block:     00246
Version:       1.0                Code:      00154
Code version: 2                   Data:      00008
```

Block: FC2

Address	Declaration	Name	Type	Start value	Comment
	in				
	out				
	in_out				
	temp				

Network: 1

```
      OPN   "data block for data reco"

      L     0.000000E+000
      T     DBD    0
      T     DBD    4
      T     DBD    8
      T     DBD   12
      T     DBD   16
      T     DBD   20
      T     DBD   24
      T     DBD   28
      T     DBD   32
      T     DBD   36
      T     DBD   40
      T     DBD   44
      T     DBD   48
      T     DBD   52
      T     DBD   56
      T     DBD   60
      T     DBD   64
      T     DBD   68
      T     DBD   72
      T     DBD   76
      T     DBD   80
      T     DBW   84
      T     DBD   88
      T     DBD   92
      T     DBD   96
      T     DBD  100
      T     DBD  104
      T     DBD  108
      T     DBD  112
      T     DBD  116
      T     DBD  120
      T     DBD  124
      T     DBD  128
      T     DBD  132
      T     DBD  136
      T     "word for data recording"
      T     "not used"
```

DB5:data block for data reco

Name:
Author:
Family:
Version: 1.0
Code version: 2

Time stamp Code: 08/04/09
Interface: 08/04/09
Lengths Block: 00298
Code: 00140
Data: 00080

Block: DB5

Address	Declaration	Name	Type	Start value	Comment
0.0	stat		STRUCT		
+0.0	stat	data	REAL	0.000000E+000	
+4.0	stat	data1	REAL	0.000000E+000	
+8.0	stat	data2	REAL	0.000000E+000	
+12.0	stat	data3	REAL	0.000000E+000	
+16.0	stat	data4	REAL	0.000000E+000	
+20.0	stat	data5	REAL	0.000000E+000	
+24.0	stat	data6	REAL	0.000000E+000	
+28.0	stat	data7	REAL	0.000000E+000	
+32.0	stat	data8	REAL	0.000000E+000	
+36.0	stat	data9	REAL	0.000000E+000	
+40.0	stat	data10	REAL	0.000000E+000	
+44.0	stat	data11	REAL	0.000000E+000	
+48.0	stat	data12	REAL	0.000000E+000	
+52.0	stat	data13	REAL	0.000000E+000	
+56.0	stat	data14	REAL	0.000000E+000	
+60.0	stat	data15	REAL	0.000000E+000	
+64.0	stat	data16	REAL	0.000000E+000	
+68.0	stat	data17	REAL	0.000000E+000	
+72.0	stat	data18	REAL	0.000000E+000	
+76.0	stat	data19	REAL	0.000000E+000	
+80.0	stat	data20	REAL	0.000000E+000	
+84.0	stat	data21	REAL	0.000000E+000	
+88.0	stat	data22	REAL	0.000000E+000	
+92.0	stat	data23	REAL	0.000000E+000	
+96.0	stat	data24	REAL	0.000000E+000	
+100.0	stat	data25	REAL	0.000000E+000	
+104.0	stat	data26	REAL	0.000000E+000	
+108.0	stat	data27	REAL	0.000000E+000	
+112.0	stat	data28	REAL	0.000000E+000	
+116.0	stat	data29	REAL	0.000000E+000	
+120.0	stat	data30	REAL	0.000000E+000	
+124.0	stat	data31	REAL	0.000000E+000	
+128.0	stat	data32	REAL	0.000000E+000	
+132.0	stat	data33	REAL	0.000000E+000	
+136.0	stat	data34	REAL	0.000000E+000	
=140.0	stat		END_STRUCT		

Symbol table

Status	Symbol	Address	Data Type	Comment
	CYCL_EXC	OB 1	OB 1	Cycle Execution
	SCALE	FC 105	FC 105	Scaling Values
	unipolar input	M 1.0	BOOL	
	dummy word for Scaling F	MW 40	WORD	
	Feedback(real)	MD 100	DWORD	
	Feedback(analog)	PIW 752	WORD	
	Feedback(double int)	MD 116	DWORD	
	Feedback(BCD)	MD 122	DWORD	
	thumbwheel switc	IW 124	WORD	
	left 2 thumbwheel digits	MW 150	WORD	
	switch to activate Setpo	I 126.0	BOOL	
	Setpt one-shot flag1	M 0.0	BOOL	
	setpt one-shot flag2	M 0.1	BOOL	
	Convert 16 BCD - Bin	FC 81	FC 81	Convert BCD to 16 bit binary
	not used1	M 244.7	BOOL	
	not used2	MB 242	BYTE	
	not used 3	MW 238	WORD	
	not used 4	MW 244	WORD	
	not used 5	MW 246	WORD	
	not used 6	C 0	COUNTER	
	hyst. one-shot flag 1	M 0.3	BOOL	
	setpoint in Bin	MW 155	WORD	
	setpoint in BCD	MW 149	WORD	
	hyst.one-shot flag 2	M 0.2	BOOL	
	upper limit	MW 166	WORD	
	switch activates hyst. o	I 126.1	BOOL	
	dummy flag 1	M 1.3	BOOL	
	dummy flag 2	M 1.1	BOOL	
	hysteresis in BCD	MW 160	WORD	
	hysteresis in binary	MW 164	WORD	
	lower limit as int	MW 170	WORD	
	upper limit as double in	MD 174	DWORD	
	output to oven	Q 124.7	BOOL	
	lower limit as double in	MD 178	DWORD	
	hysteresis flag 1	M 2.0	BOOL	
	hysteresis flag 2	M 2.1	BOOL	
	STL for data recording	FC 1	FC 1	
	data block for data reco	DB 5	DB 5	
	not used	MD 200	DWORD	
	FB containing DWs	FC 2	FC 2	
	switch to begin data rec	I 126.7	BOOL	
	flag for cycler	M 0.4	BOOL	
	cycler for data recordin	T 1	TIMER	
	7-seg display	QB 125	BYTE	
	reset Data block switch	I 126.6	BOOL	
	positive edge detector	M 2.6	BOOL	
	one-shot for reset flag	M 2.7	BOOL	

Status	Symbol	Address	Data Type	Comment
	on-time timer	T 4	TIMER	
	off-time timer	T 5	TIMER	
	data block full flasher	Q 124.6	BOOL	
	Data block full flag	M 0.5	BOOL	
	CONT_C	FB 41	FB 41	Continuous Control
	PULSEGEN	FB 43	FB 43	Pulse Generation
	word for data recording	MW 140	WORD	

PLC System Applications Project 1 Temperature Control

By David Holbeche SID: 311383922

Conclusion Questions:

1. Hysteresis prevents the output from "chattering" or making fast, continual switches if the cycling above and below the setpoint occurs very rapidly. It prevents damage to contactors and valves caused by noise at the output. This noise can cause "chattering" at the output. The addition of hysteresis to our system does improve the control of the system. It lowered the cycling frequency at the output, so that the output crossed the setpoint less frequently, giving a more accurate control of the temperature, This improvement is however minimal, as the process temperature is still cycling above and below the setpoint, and does not settle at or near the setpoint, as is the case with proportional only , or PID control.

2. Proportional control provides a continuous correction of the output and provides stability to the process being controlled. It will eliminate the cycling associated with On/Off control and provides a much more accurate control of the process temperature. In proportional control, the output is proportional to the error(The setpoint – the feedback), so the output is continuously updated or corrected based on the error of the system. As the output reaches the setpoint the power applied to the output is decreased (as the error decreases), which has the effect of slowing down the oven output so that it will not overshoot the setpoint, but will approach the setpoint and settle with some offset. This offset is necessary for proportional control to work, and is an inherent problem with proportional only control. For our purposes, a form of pulse-width-modulation is used, to vary the ON and OFF times of the output. When the error is large, more power is applied to the oven, so the ON time of the output is much larger than the OFF time. When the output approaches the setpoint, the error is small and the ON time is less than the OFF time. I did not complete the Proportional Control part of this project, however in the major project proportional control provided a much more accurate form of control than On/Off control. There was minimal cycling around the setpoint, and the output settled very close to the setpoint with a small error (about 3.95 degrees Celsius) .

3. Overshoot is caused by the process output increasing too quickly as it approaches the setpoint. This is a problem in On/Off control, where the power supplied to the output is not proportional to the error: it is either fully "On" or fully "Off". In On/Off control the output is cycling, so overshoot occurs every cycle. This can be eliminated by using proportional, or PID control. With proportional control the power applied to the process is decreased, as the output temperature approaches the setpoint, which will prevent overshoot. In a PID system, derivative action can prevent overshoots by providing a correction proportional to the rate of change of the error.

4. The main advantage of a 4- 20 mA current loop is that the accuracy of the signal is not affected by voltage drop in the interconnecting wiring, and that the loop can supply operating power to the device. The 4 mA lower limit allows a receiving instrument to detect some failures in the loop and also allows transmitter devices to be powered from the same current loop (called two-wire transmitters). The 4 – 20 mA current loop is a very robust sensor signalling standard. In a 4 – 20 mA current loop, all of the signalling current flows through all the components. All the components in the loop drop voltage due to the signalling current flowing through them. The signaling current is not affected by these voltage drops as long as the power supply voltage is greater than the sum of the voltage drops around the loop at the maximum signaling current of 20mA. Other advantages are the simplicity of its two wire construction, and its insensitivity to noise. A 4 – 20 mA current loop is more useful than a