

# PL410

## PROGRAMMABLE LOGIC CONTROLLER WITH RS485 MODBUS COMMUNICATIONS



## USER MANUAL



P.O.Box 24  
STANFIELD 3613  
SOUTH AFRICA

Tel: +27 (031) 7028033  
Fax: +27 (031) 7028041  
Email: [proconel@proconel.com](mailto:proconel@proconel.com)  
Web: [www.proconel.com](http://www.proconel.com)

# TABLE OF CONTENTS

1.	AN OVERVIEW OF THE PL410 PROGRAMMABLE LOGIC CONTROLLER.....	4
1.1	DESCRIPTION .....	4
2.	PL401 GENERAL INFORMATION .....	5
2.1	PHYSICAL DIMENSIONS .....	5
2.2	GROUNDING/SHIELDING.....	5
3.	PL410 HARDWARE .....	6
3.1	SPECIFICATIONS .....	6
3.2	WIRING TERMINALS .....	6
3.3	FRONT PANEL DESCRIPTION. ....	7
4.	CONFIGURATION .....	8
4.1	HARDWARE CONNECTIONS.....	8
4.1.1	Connecting the Power.....	8
4.1.2	Connecting the Inputs.....	8
4.1.3	Connecting the Outputs.....	9
4.1.4	Connecting the RS232/RS485 Network.....	9
4.1.5	Connecting the programming port to a PC.....	10
4.2	PL410 CPU.....	11
4.2.1	Program Memory.....	11
4.2.2	Data Memory.....	11
4.2.3	Data Memory Map.....	12
4.2.4	Digital Input Map.....	13
4.2.5	Digital Output Map.....	13
4.2.6	Timer Map.....	14
4.2.7	Counter Map.....	14
4.2.8	Control Relay Map.....	14
4.2.9	System Relay Map.....	15
5.	RS485 Modbus Communications.....	16
5.1	Modbus Master.....	16
5.2	Modbus Slave.....	16
6.	PID.....	17
6.1	Introduction .....	17
6.2	Loop Features .....	18
6.3	Loop Calculations.....	18
6.4	Loop Setup Parameters.....	19
6.4.1	Configuring the number of PID loops .....	19
6.4.2	Description of the memory locations.....	20
6.5	Setting up the loop parameters .....	20
6.5.1	Setpoint (SP).....	20
6.5.2	Setpoint Limits .....	21
6.5.3	Process Variable (PV).....	21
6.5.4	Control Output (OP) .....	22
6.5.5	Control Output Limits.....	23
6.5.6	Description of the PID Mode register.....	23
6.5.7	Description of the PID Status register .....	24
6.5.8	PID Mode register bit 0 & bit 1 – Manual/Auto Mode.....	25
6.5.9	PID Mode register bit 2 – Control Output Format Unsigned/Signed .....	26
6.5.10	PID Mode register bit 3 – Control Output Format 12/15 Bits .....	26
6.5.11	PID Mode register bit 4 – PV Linear/Square Root .....	26
6.5.12	PID Mode register bit 5 – Control Output Normal/Inverted.....	26
6.5.13	PID Mode register bit 6 – Error Linear/Squared.....	26
6.5.14	PID Mode register bit 7 – Integral Gain Seconds/Minutes .....	27
6.5.15	PID Mode register bit 8 – Integrator Limit .....	27
6.5.16	PID Mode register bit 9 – Bumpless Transfer .....	27
6.5.17	PID Mode register bit 10 – PI/PID Mode .....	27
6.5.18	PID Mode register bit 11 – Open Loop/Closed Loop Auto Tuning.....	27
6.5.19	PID Mode register bit 12 – PV Value Alarm Enable.....	28

6.5.20	PID Mode register bit 13 – PV Deviation Alarm Enable.....	29
6.5.21	PID Mode register bit 14 – PV Rate of Change Alarm Enable .....	30
6.5.22	PID Error Deadband .....	30
6.5.23	PID Derivative Gain Limit .....	31
6.6	Tuning the PID loop .....	31
6.6.1	Manual Tuning .....	31
6.6.1.1	Determining the Loop Sample Time.....	31
6.6.1.2	Determining the Loop Gains.....	32
6.6.2	Automatic Tuning.....	33
6.6.2.1	Open Loop Method.....	33
6.6.2.2	Closed Loop Method .....	33
6.6.3	Analog Filter.....	34
7.	Modbus Memory Map ( MODULE TYPE = 42) .....	35
8.	Ladder Logic Function Blocks .....	37

# 1. AN OVERVIEW OF THE PL410 PROGRAMMABLE LOGIC CONTROLLER

---

## 1.1 DESCRIPTION

The PL410 PLC has been developed as a versatile controller for machine control as a relay replacement product. The fact that the controller is programmable enables the user to program their own unique logic requirements and not be restricted by a pre-programmed unit or hardwired relays and timers.

The PL410 PLC is programmed in ladder logic. PROCON's PROSOFT windows-based PC software is used to generate the ladder diagram, compile the program, and then download the program to the PL410 via the programming port on the front of the unit.

The I/O consists of 8 digital inputs and 4 relay outputs. The inputs are opto-isolated and a built in field supply is used to power the inputs so no external power supply is required.

All wiring is done with screw terminals on removable connectors.

The programming port requires the use of a special adaptor to connect it to an RS232 communications port of a PC. This port supports the Modbus RTU protocol and all of the internal registers and I/O status can be accessed through this port.

The features in the PL410 include a Real Time Clock and 4 PID loops. The PID loops can be tuned manually or automatically using the open loop or closed loop techniques.

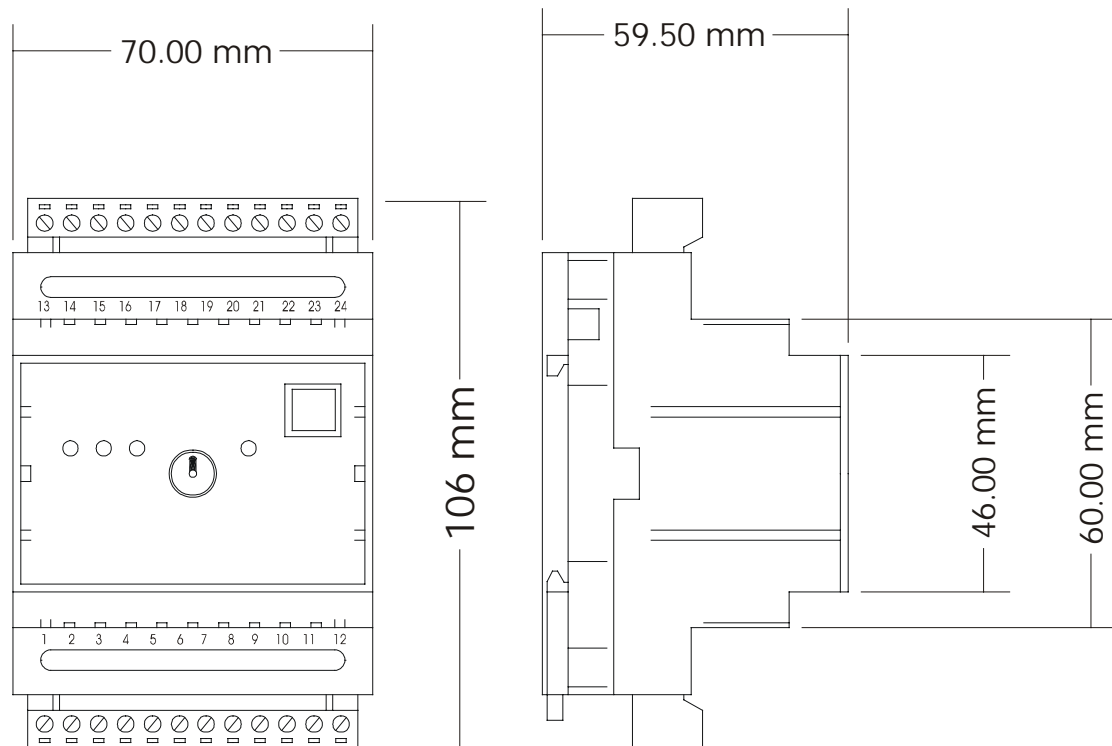
The RS485 communications port can be configured as a Modbus master or Modbus slave.

## 2. PL401 GENERAL INFORMATION

---

### 2.1 PHYSICAL DIMENSIONS

The PL401 enclosure is shown below. The module has been designed with a quick snap-in assembly for mounting onto DIN-rail's as per DIN EN 50 022.



### 2.2 GROUNDING/SHIELDING

In most cases, the PL401 will be installed in an enclosure along with other devices, which generate electromagnetic radiation. Examples of these devices are relays and contactors, transformers, motor controllers etc. This electromagnetic radiation can induce electrical noise into both power and signal lines, as well as direct radiation into the module causing negative effects on the system. Appropriate grounding, shielding and other protective steps should be taken at the installation stage to prevent these effects. These protective steps include control cabinet grounding, module grounding, cable shield grounding, protective elements for electromagnetic switching devices, correct wiring as well as consideration of cable types and their cross sections.

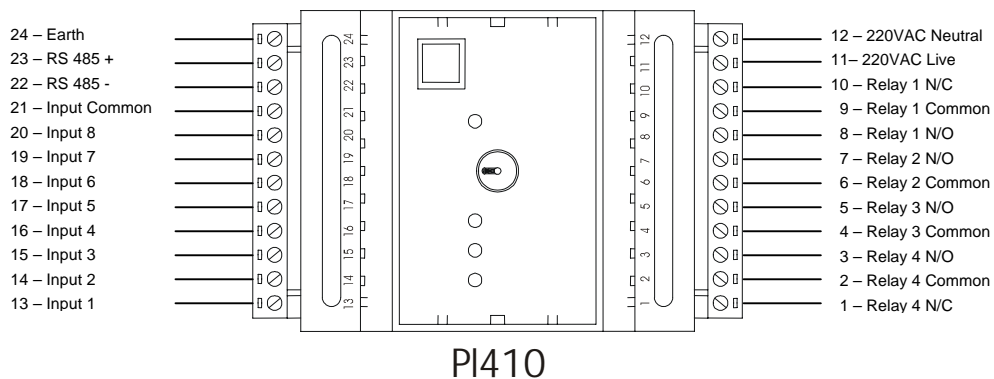
### 3. PL410 HARDWARE

---

#### 3.1 SPECIFICATIONS

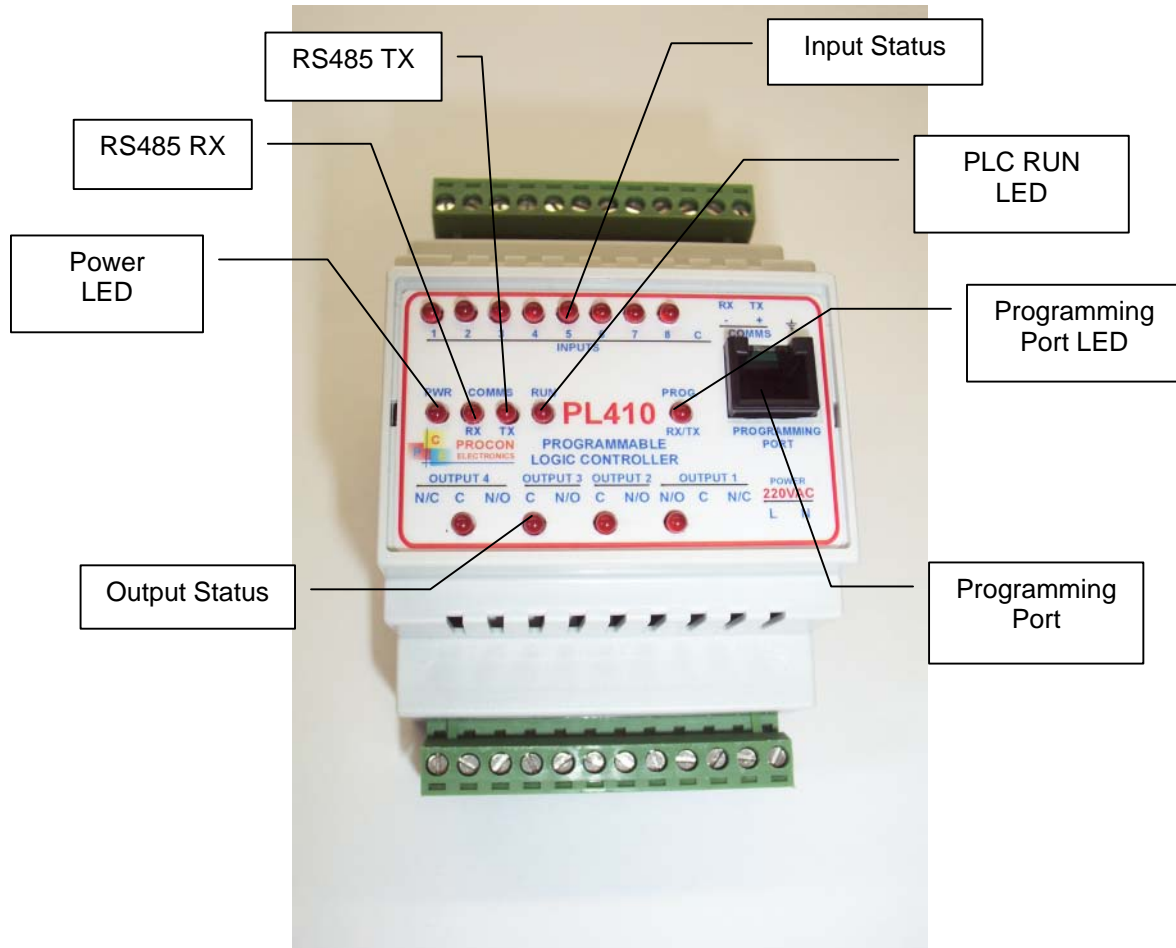
- POWER REQUIREMENT:** 200 - 260VAC 50/60Hz.
- 8 X INPUTS:** These inputs may be activated by a potential free relay contact or open collector NPN transistor output. These inputs are isolated from the logic.
- 4 X OUTPUTS:** These outputs are a normally open relay contact rated at 6A/220VAC (resistive).
- INDICATORS:** LED indicators show power, RS485 Rx & Tx and programming communications.
- RS485 BAUD RATE:** 2400, 4800, 9600, 19200, 38400.
- RS485 PARAMETERS:** Parity- None, Even, Odd. Stop Bits- 1 or 2.
- CONNECTORS:** 2 X 12 Way Removable Connectors with screw terminals.
- DIMENSIONS:** 106mm (HIGH) X 70mm (WIDE) X 59.5mm (DEEP)
- OPERATING TEMPERATURE:** -20°C to +60°C
- STORAGE TEMPERATURE:** -20°C to +65°C
- HUMIDITY:** up to 95% non condensing

#### 3.2 WIRING TERMINALS



### 3.3 FRONT PANEL DESCRIPTION.

The led's on the front panel of the PL410 Module are used to indicate power, RS485 Rx & Tx Data. A programming LED is used to indicate communications with a PC during programming and Debugging. The ON/OFF status of the inputs and outputs are also shown with LED's.

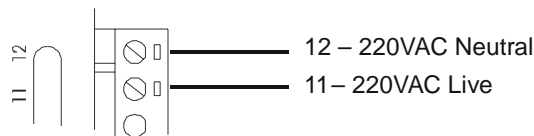


## 4. CONFIGURATION

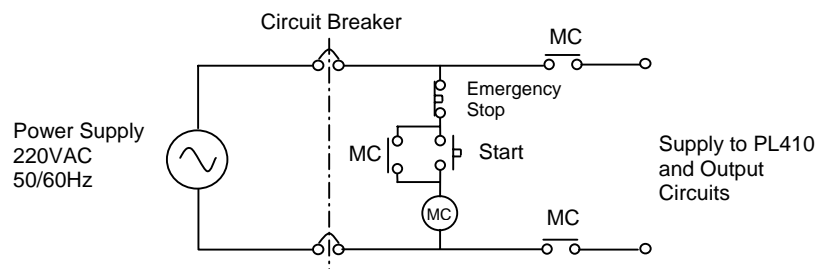
### 4.1 HARDWARE CONNECTIONS.

#### 4.1.1 Connecting the Power.

Power must be applied to terminal 11 (220VAC LIVE) and terminal 12 (220VAC NEUTRAL). When the power is initially applied the power LED will illuminate and all other LED's will be off.

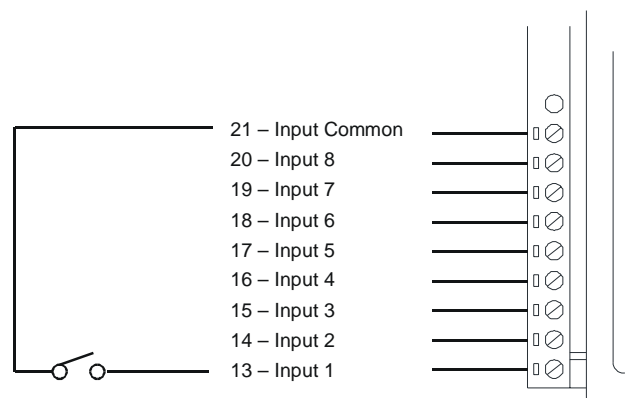


As the PLC is often used to control machinery, which could present a risk of personal injury or damage to equipment, it is good practice to wire an external emergency stop circuit to the power supply on the PLC. The circuit below shows how a mechanical contactor (MC) is used with start/stop buttons to provide this facility.



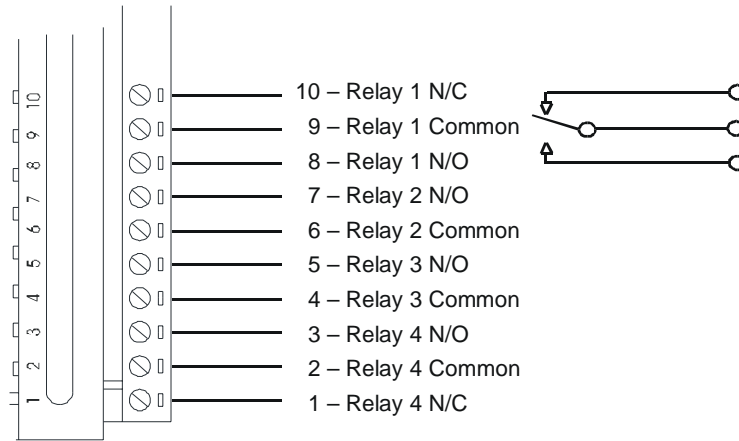
#### 4.1.2 Connecting the Inputs.

The inputs are sourced from an internally isolated power supply and can be switched by a potential free contact or a NPN transistor. The inputs all share the common terminal.

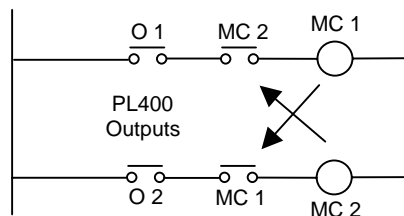


### 4.1.3 Connecting the Outputs.

The outputs are potential free relay contacts. Note that only Relay output 1 and 4 have normally closed contacts as well as normally open contacts.

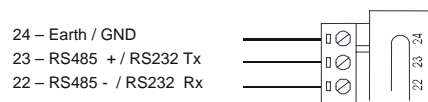


The outputs may be used to control the direction of a motor. For example output 1 could be used to control the forward direction of the motor and output 2 used to control the reverse direction. It could be possible under fault conditions that both outputs switch on at the same time. It is considered good practice to interlock the two outputs both in the ladder program and using external mechanical contactors. The diagram below shows how this is done.



### 4.1.4 Connecting the RS232/RS485 Network.

The diagram below shows how to connect the RS232/485 network to the PL401.





## 4.2 PL410 CPU.

The CPU (central Processing Unit) performs all of the tasks that are required to make the PLC function and run your ladder program. Some of the tasks include:

1. Reading the status of the inputs.
2. Executing the program.
3. Updating the outputs.
4. Doing diagnostics.
5. Servicing the communications ports.
6. Running the timers.

### 4.2.1 Program Memory.

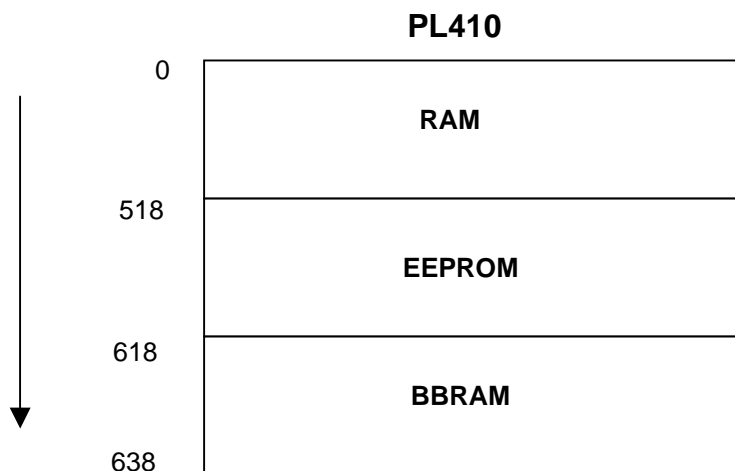
The programming port is used to program the PLC. The program which is sent from the PC using the ProSoft ladder editor, is stored in FLASH memory. This memory does not get lost when the power fails and so will remain permanently in the PLC until it is reprogrammed.

### 4.2.2 Data Memory.

All the variables used in the program are stored in Data memory. Both the Digital and Analog values are stored in this memory along with the timers, counters, and user memory.

The memory is divided up into 3 sections.

1. RAM – Random Access Memory. This memory is the most widely used memory and is where most of the data is stored. All timers, counters, I/O statuses and system information use this memory. If the power fails then all the information in this memory is lost and is re-initialized to zero when the PLC starts again.
2. EEPROM – This memory is used to store parameters such as set-points and configuration data as it retains its memory when the power is turned off. The one point to remember is that this memory can only be written to 10 000 times before it wears out so you must not write to this memory all the time as you can with RAM.
3. BBRAM (Optional extra) – This is battery backed RAM and also retains its memory when the power is switched off. This memory is slow compared to RAM and should not be used where normal RAM can be used. This memory is ideal for storing values such as used in counting applications. The Real time clock is also stored in this memory.



### 4.2.3 Data Memory Map.

All of the variables used in the PLC are stored in data memory. In order for your program to get access to these variables you need to know the memory address. The memory address starts at zero and the size depends on the PLC being used. Each memory location consists of 16 bits. Thus one memory location can be used to store the status of 16 digital I/O points or an analog value from 0 to 65535. Some of the ladder functions use two consecutive memory locations to store larger values. Refer to the ProSoft user manual to find out about the ladder functions.

<b>PL410 MEMORY MAP</b>			
Memory Type	Digital Reference	Memory Address	Quantity
Module Type = 41	-	M0	1
Digital Inputs	I1 to I8	M1	8
Digital Outputs	O1 to O4	M2	4
Timer Status	T1 to T64	M3 – M6	64
Counter Status	C1 to C64	M7 – M10	64
Control Relays	R1 to R64	M11 – M14	64
System Relays	S1 to S32	M15 – M16	32
Timer Memory	-	M19 – M82	64
Counter Memory	-	M83 – M146	64
User RAM Memory	-	M147 – M518	371
User EEPROM	-	M519 – M618	100
User BBRAM	-	M619 – M638	20

#### 4.2.4 Digital Input Map.

MSB															PL410 DIGITAL INPUTS															LSB		ADDRESS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	18	17	16	15	14	13	12	11	0								
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I8	I7	I6	I5	I4	I3	I2	I1	M1								

Bit Number	Digital Input Number	Description
0	I1	Digital Input 1
1	I2	Digital Input 2
2	I3	Digital Input 3
3	I4	Digital Input 4
4	I5	Digital Input 5
5	I6	Digital Input 6
6	I7	Digital Input 7
7	I8	Digital Input 8
8	-	-
9	-	-
10	-	-
11	-	-
12	-	-
13	-	-
14	-	-
15	-	-

#### 4.2.5 Digital Output Map.

MSB															PL410 DIGITAL OUTPUTS															LSB		ADDRESS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	O4	O3	O2	O1	0												
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	O4	O3	O2	O1	M2												

Bit Number	Digital Input Number	Description
0	O1	Relay Output 1
1	O2	Relay Output 2
2	O3	Relay Output 3
3	O4	Relay Output 4
4	-	-
5	-	-
6	-	-
7	-	-
8	-	-
9	-	-
10	-	-
11	-	-
12	-	-
13	-	-
14	-	-
15	-	-

#### 4.2.6 Timer Map.

MSB			PL410 TIMER STATUS												LSB		ADDRESS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
T16	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	M3	
T32	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	M4	
T48	T47	T46	T45	T44	T43	T42	T41	T40	T39	T38	T37	T36	T35	T34	T33	M5	
T64	T63	T62	T61	T60	T59	T58	T57	T56	T55	T54	T53	T52	T51	T50	T49	M6	

#### 4.2.7 Counter Map.

MSB			PL410 COUNTER STATUS												LSB		ADDRESS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
C16	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	M7	
C32	C31	C30	C29	C28	C27	C26	C25	C24	C23	C22	C21	C20	C19	C18	C17	M8	
C48	C47	C46	C45	C44	C43	C42	C41	C40	C39	C38	C37	C36	C35	C34	C33	M9	
C64	C63	C62	C61	C60	C59	C58	C57	C56	C55	C54	C53	C52	C51	C50	C49	M10	

#### 4.2.8 Control Relay Map.

MSB			PL410 CONTROL RELAYS												LSB		ADDRESS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R16	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	M11	
R32	R31	R30	R29	R28	R27	R26	R25	R24	R23	R22	R21	R20	R19	R18	R17	M12	
R48	R47	R46	R45	R44	R43	R42	R41	R40	R39	R38	R37	R36	R35	R34	R33	M13	
R64	R63	R62	R61	R60	R59	R58	R57	R56	R55	R54	R53	R52	R51	R50	R49	M14	

#### 4.2.9 System Relay Map.

PL410 SYSTEM RELAYS															ADDRESS		
MSB	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB	0
S16	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	M15	
S32	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	M16	

Bit Number	Digital Input Number	Description
0	S1	ON
1	S2	1st Scan
2	S3	0.1 Second Clock Period
3	S4	1 Second Clock Period
4	S5	1 Minute Clock Period
5	S6	CMP < MEM/K
6	S7	CMP = MEM/K
7	S8	CMP > MEM/K
8	S9	PLC Running
9	S10	PLC Re-Program Request
10	S11	PLC Re-Program Acknowledge
11	-	-
12	-	-
13	-	-
14	-	-
15	-	-

## 5. RS485 Modbus Communications

---

### 5.1 Modbus Master.

The RS485 communications port can be configured to function as a Modbus master device. To enable this mode you must make sure that the Modbus Master tick box is selected in Procon's ProSoft PLC programming software.

In this mode, you can configure the PL410 to read a range of registers from a remote Modbus slave or you can write a range of registers to a remote slave. You can configure up to 20 of these communications blocks.

The setup parameters are as follows:

- **Remote ID.** This is the network ID of the Modbus slave device.
- **Function.** You must enter a value of 3 to read a range of registers and a value of 16 to write to a range of registers. Function 3 reads registers in the slave and stores them in memory in the PL410. Function 16 reads memory in the PL410 and writes them to registers in the slave device.
- **Local Address.** This is the memory location in the PL401 where the data will be read from or written to. For example, if you want to access memory M1 then you must put a 1 into the local address field. ( Do not put the Modbus address 30002 ).
- **Range.** This is the number of consecutive memory locations that will be transmitted.
- **Remote Address.** This is the register location in the slave device where data will be written to or read from. If you want to access a modbus register for example 40010 in the remote slave device, then you must put a value of 9 into this field.

### 5.2 Modbus Slave.

The RS485 communications port can be configured to function as a Modbus slave device.

When configured as a modbus slave, the PL410 will respond to network requests from a modbus master on the network. This could be another PL410.

The modbus functions supported are as follows:

PL410 Modbus Slave Commands				
Modbus Function	Description	Memory start	Memory end	Max. Range
1or2	Reads a range of bits from any part of RAM	M0	M518	1600
3or4	Reads a range of registers from RAM, EEPROM and BBRAM.	M0	M638	100
5	Reads a single Bit from any part of RAM	M0	M518	1
6	Reads a single register from RAM, EEPROM and BBRAM.	M0	M638	1
15	Writes a range of bits to RAM.	M2	M518	1600
16	Writes a range of registers from RAM, EEPROM and BBRAM.	M2	M638	100

## 6. PID

---

### 6.1 Introduction

The Proportional-Integral-Derivative controller is a standard component in industrial control applications. It measures the output of a process (process variable – PV) and controls an input, with a goal of maintaining the output at a target value, which is called the setpoint (SP). A common application is to control a process temperature, in which case the PID controller acts as a sophisticated thermostat. It can be used to control pressure, flow rate, force, speed or a number of other variables.

The basic idea is that the controller reads a sensor. Then it subtracts the measurement from a desired setpoint to determine an error. The error is then treated in three different ways simultaneously:

1. To handle the present, the error is multiplied by a proportional constant **P**. This term is responsible for process stability. A low value could cause the PV to drift away from the setpoint, whilst a high value could cause the PV to oscillate.
2. To handle the past, the error is integrated (summed) over a period of time, and then multiplied by a constant **I**. This term is responsible for driving the error to zero. If it is set too high it can also lead to instability.
3. To handle the future, the first derivative of the error (its rate of change) is calculated with respect to time, and multiplied by a constant **D**. This term is responsible for the system response. If it is too high the PV will oscillate and if it is too low the PV will respond sluggishly.

These terms are added together to give the PID control output variable which controls the process.

The PL410 PLC can operate up to 4 PID loops at the same time and takes care of the calculations as well as handling the alarms.

The sensor would normally be a thermocouple, RTD or other analog input. As the PL410 has only got digital inputs and outputs, it is necessary to get the analog value for the PV from a remote device such as one of the MOD-MUX input modules manufactured by Procon Electronics. This device would be connected to the PL410 on the RS485 network and the PL410 would read the PV from the remote I/O module.

If an analog output is required (continuous control) then an analog output module will have to be placed onto the RS485 network and the PL410 would write values to this device. If a digital output is required (time proportional control) then the relays on the PL410 can be used.

## 6.2 Loop Features

The PID loops have a number of features that provide for flexibility of use. These features are configured by setting up the corresponding memory locations.

- 0, 1, 2, 3 or 4 PID loops can be configured. Each loop uses a certain amount of user memory only if it is configured.
- The PID loops use the Position PID equation.
- The loop sample time is user configurable.
- The square root of the PV can be enabled for flow applications.
- The error term can be squared.
- A deadband can be applied to the error term so that no change is made to the control output when the error is within the deadband.
- The control output can be inverted for reverse acting systems.
- Limits can be applied to the setpoint and to the control output lower and upper values.
- An anti-windup function stops the integrator when the control output reaches its lower or upper limit.
- PV value, deviation and rate of change alarms are provided.
- Manual and auto-tuning options. Auto-tuning by open loop or closed loop.

## 6.3 Loop Calculations

The PID loop uses the Ziegler-Nichols method for calculating the controller output. The algorithm is as follows:

$$\text{Controller Output (t)} = K_P \left[ \overbrace{e(t)}^{\text{P}} + \overbrace{\frac{1}{K_I} \int_0^t e(t) dt}^{\text{I}} + \overbrace{K_D \frac{de(t)}{dt}}^{\text{D}} \right] + OP_{\text{OFFSET}}$$

The error  $e(t) = SP(t) - PV(t)$

The following points are important to remember:

1. The Proportional Gain  $K_P$  affects all of the terms.
2. The Integral Gain  $K_I$  is an inverse gain. This means that a larger value will have a smaller effect.

It is possible to configure the algorithm to perform PI only or PID.

## 6.4 Loop Setup Parameters

### 6.4.1 Configuring the number of PID loops

The PL410 can run up to 4 PID loops at once. The number of loops is programmed into memory location **M521**. This memory corresponds to the area which is held in EEPROM so the value is not lost at power down. A value of 0 means that no loops are configured, 1 means that only a single loop is to be used, etc up to 4 loops. Do not enter a number if you are not going to use the loops as this will use up unnecessary memory and consume program execution time.

The PID software uses part of user memory (RAM) and (EEPROM). If you have not configured any of the loops then this memory can be used for other purposes. The memory used by each loop is shown in the table below:

PID Loop Number	RAM	EEPROM
1	M480 - M486	M530 - M549
2	M487 - M493	M550 - M569
3	M494 - M500	M570 - M589
4	M501 - M507	M590 - M609

## 6.4.2 Description of the memory locations

The table below lists the memory used by the loops. These values are repeated according to the number of loops you have configured.

Memory Address				Description	Value	Format
Loop1	Loop2	Loop3	Loop4			
480	487	494	501	Setpoint (SP)		Binary
481	488	495	502	Process Variable (PV)		Binary
482	489	496	503	Accumulated Error		Binary
483	490	407	504	Output Offset		Binary
484	491	498	505	Last Error		Binary
485	492	499	506	Control Output (OP)		Binary
486	493	500	507	Status		Bits
530	550	570	590	Mode		Bits
531	551	571	591	Sample Time	0.05 to 99.99	Binary
532	552	572	592	P Proportional Gain	0.01 to 99.99	Binary
533	553	573	593	I Integral Gain	999.9 to 0.1	Binary
534	554	574	594	D Derivative Gain	0.01 to 99.99	Binary
535	555	575	595	PV LO-LO Alarm Value		Binary
536	556	576	596	PV LO Alarm Value		Binary
537	557	577	597	PV HI Alarm Value		Binary
538	558	578	598	PV HI-HI Alarm Value		Binary
539	559	579	599	PV Deviation LO Alarm		Binary
540	560	580	600	PV Deviation HI Alarm		Binary
541	561	581	601	PV Rate of Change Alarm		Binary
542	562	582	602	PV Alarm Hysteresis		Binary
543	563	583	603	Deadband		Binary
544	564	584	604	Derivative Gain Limit		Binary
545	565	585	605	Setpoint LO Limit Value		Binary
546	566	586	606	Setpoint HI Limit Value		Binary
547	567	587	607	Output LO Limit Value		Binary
548	568	588	608	Output HI Limit Value		Binary
549	569	589	609	Reserved		

## 6.5 Setting up the loop parameters

In the following section all memory addresses will refer to PID loop1. You must change the addresses according to the table above if you are referring to a different loop. It is advised that you configure the PID parameters in the order below although you can change any of the parameters at any later time.

### 6.5.1 Setpoint (SP)

The setpoint is the desired operating point of the process and is a variable which is programmed by the operator. The setpoint is written to memory location **M480**. This could come from an operator interface such as a SCADA system, an operator panel or from a potentiometer connected to an analog input module. The user needs to be able to adjust this value to set the operating point of the process.

The format of the setpoint value must be the same as the process variable PV. For example if the PV is 0 – 4095 (12 bit unsigned) then the setpoint must be set within this range. If the PV is a temperature say between 10.0 and 150.0 then the setpoint must be within this range.

During testing it is possible to enter the setpoint from the **Debug->PLC->View Memory** menu item in ProSoft.

### 6.5.2 Setpoint Limits

The setpoint limits ensure that the setpoint value being used in the loop calculations is always within the limits. The lower limit is located in memory **M545** and the upper limit is in **M546**. The format of the limit value must be the same as the process variable PV. The upper limit must always be greater than the lower limit and these two values must be within the working range of the PV.

### 6.5.3 Process Variable (PV)

The process variable is the value that is read back from the process and is used in the error term of the loop calculation. The PV is located in memory **M481**. The PL410 does not have a facility for analog inputs, so this value must be written to by an external device, or the Modbus Master Poll Table can be used to read an analog value from a Modbus slave module.

Setup the analog input module now. If for example you have a MOD-MUX thermocouple module as your input device (MM8TC) with an ID of 1, then the Modbus Master Poll Table can be used to read an analog value from ID 1 and to put the temperature value into memory location M481 in the PL410.

Function 3 - Read Registers  
Function 16 - Write Registers

Poll Block Number	Remote ID	Function	Local Address	Range	Remote Address
1	1	3	481	1	1
2	0	0	0	0	0
3	2	16	2	1	1
4	0	0	0	0	0
5	0	0	0	0	0

Submit Close

## 6.5.4 Control Output (OP)

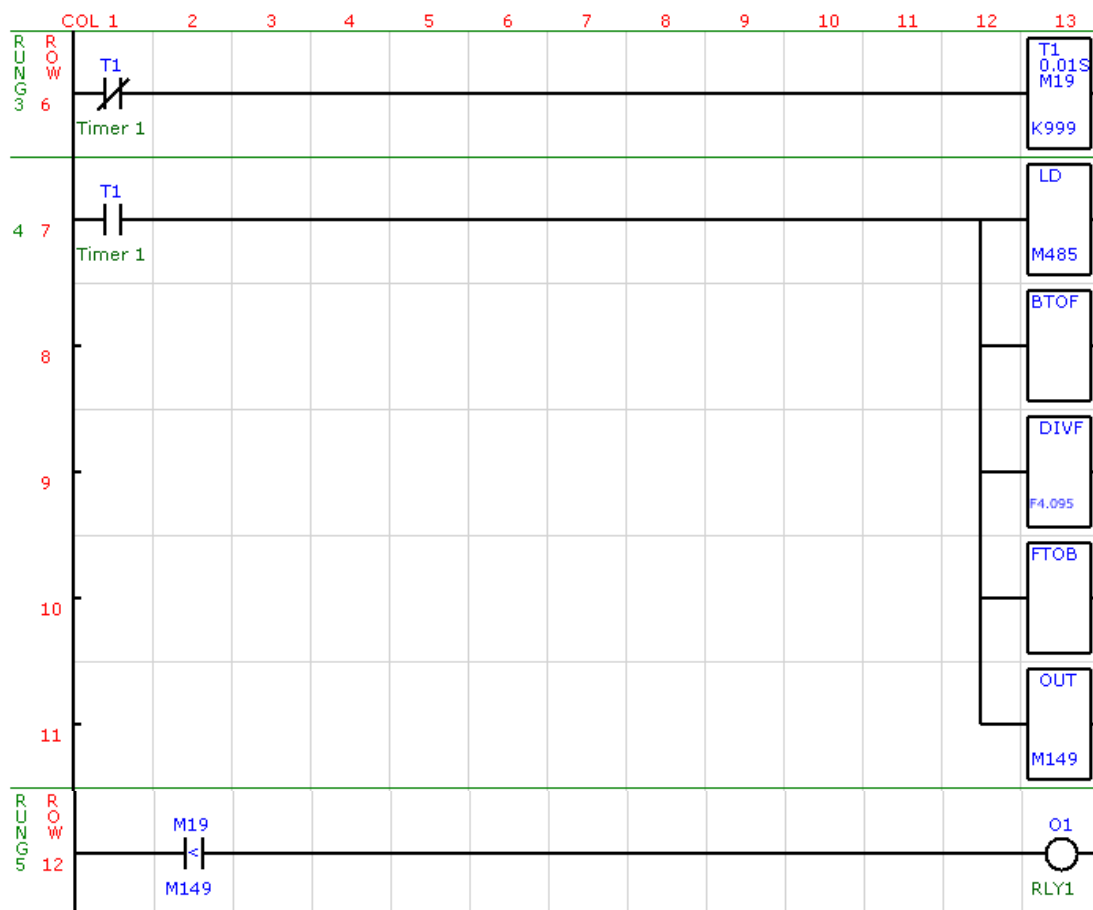
The control output is the result of the PID loop calculation and this value is used to control the process. The value is placed in memory location **M485** by the software and must be transferred to an analog output module for continuous control or converted to a pulse width modulated value for time proportional control (ON-OFF).

The ladder program below shows how a 12 bit unsigned control output value (0 - 4095) is converted to a 0 – 10 second time proportional control value to switch relay output 1 on and off.

Timer 1 is used to generate the 10 second period with 0.01 second resolution. (0 – 999 ticks). Every 10 seconds the timer resets itself back to zero.

When timer T1 reaches 999 a new value is calculated from the control output value and saved in memory location M149.

Every PLC scan cycle the value in timer 1 is compared to the value in M149. If the timer value is less than M149 then the relay output is switched on, otherwise it is switched off.

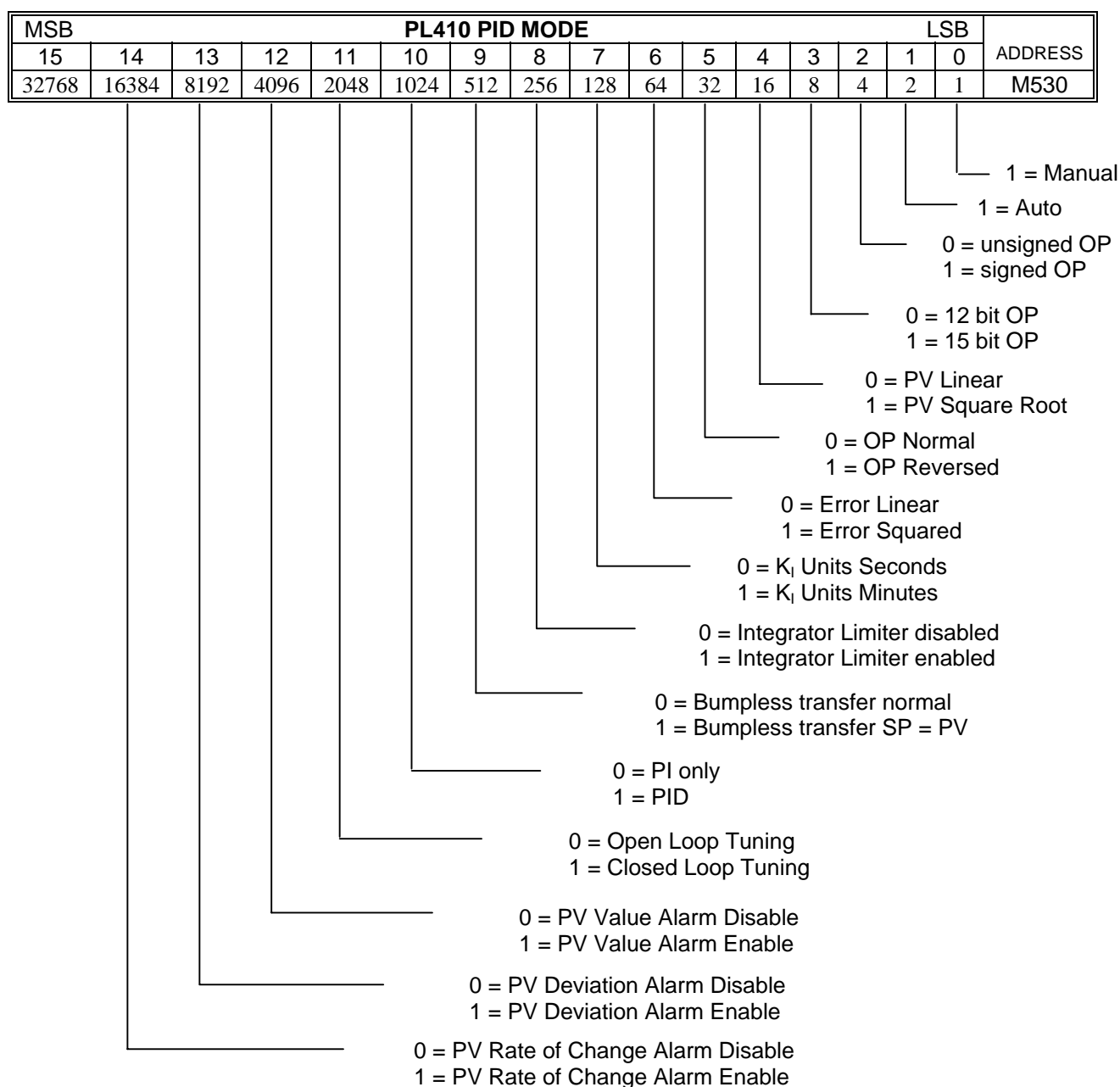


## 6.5.5 Control Output Limits

The control output limits ensure that the output value resulting from the loop calculations is always within the limits. The lower limit is located in memory **M547** and the upper limit is in **M548**. The format of the limit value must be the same as the output value. The upper limit must always be greater than the lower limit and these two values must be within the working range of the process.

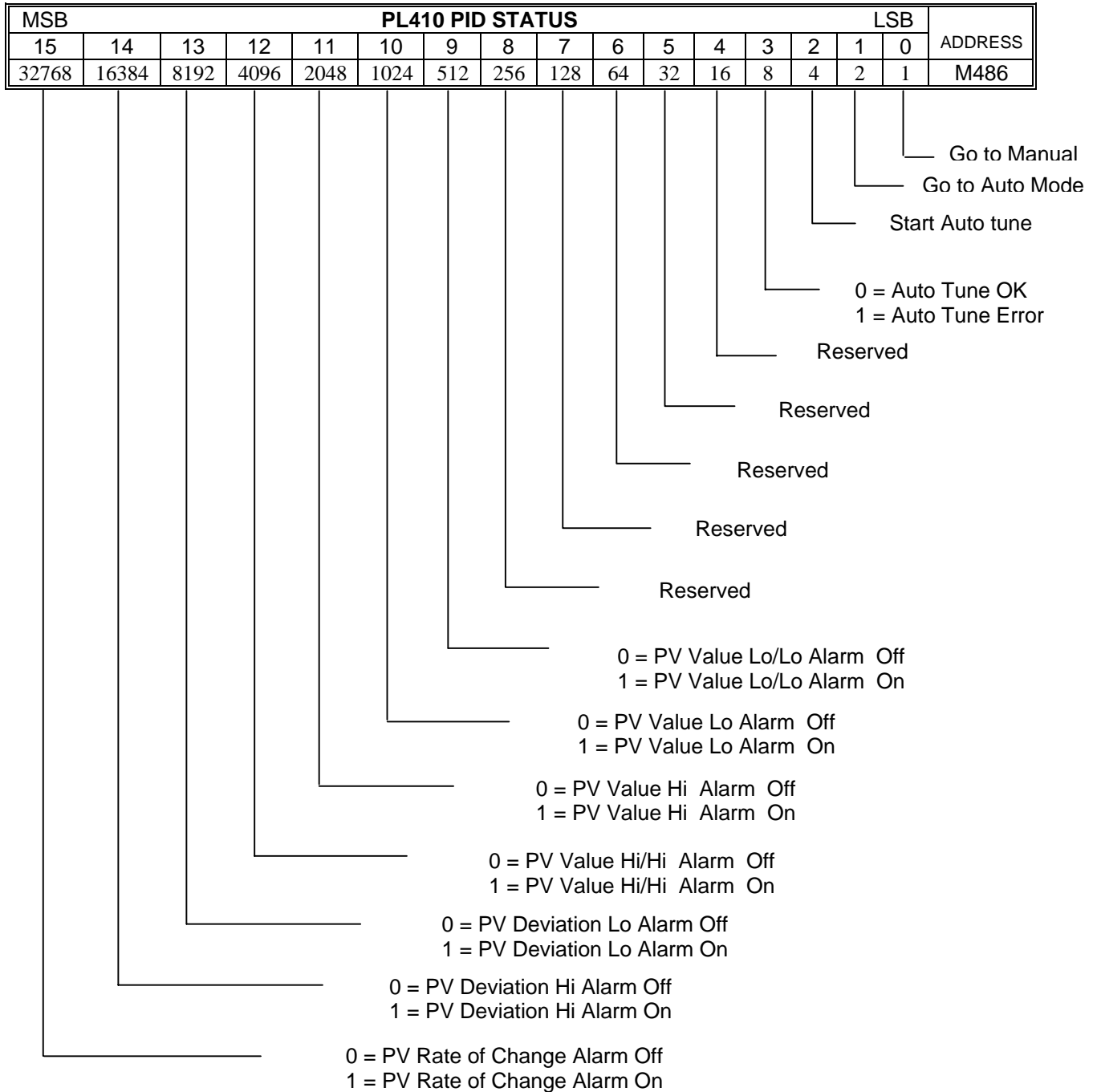
## 6.5.6 Description of the PID Mode register

The mode register is in memory location M530. This register consists of 16 bits. Each bit is used to setup the various mode options. The register is as follows:



## 6.5.7 Description of the PID Status register

The status register is in memory location M486. This register consists of 16 bits. Each bit is used to read the various status options. The register is as follows:



## 6.5.8 PID Mode register bit 0 & bit 1 – Manual/Auto Mode

The first two bits in the mode register are used to save the operating mode of the PID loop. You can read these bits to determine what mode the loop is in.

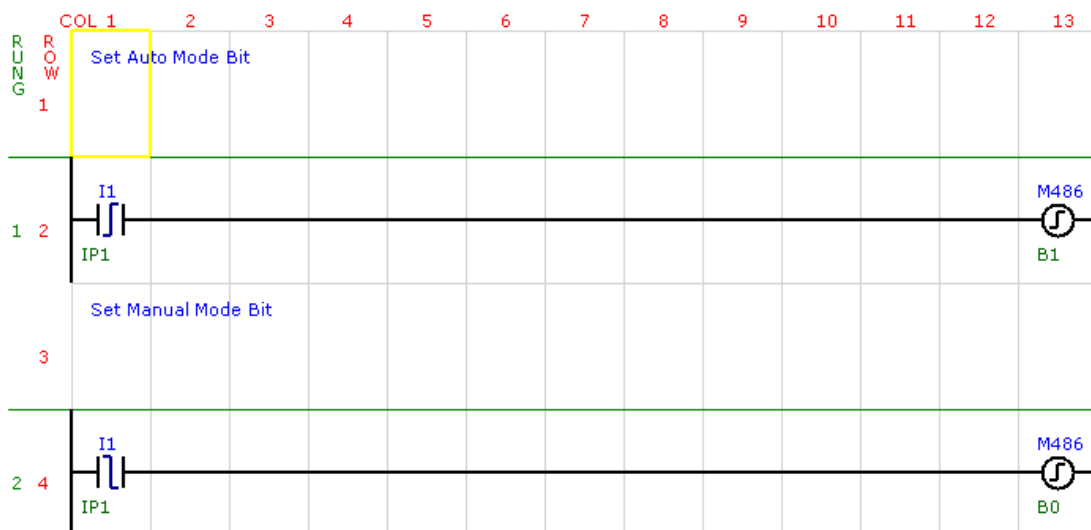
Manual Mode M530 bit0 is on.

Auto Mode M530 bit1 is on.

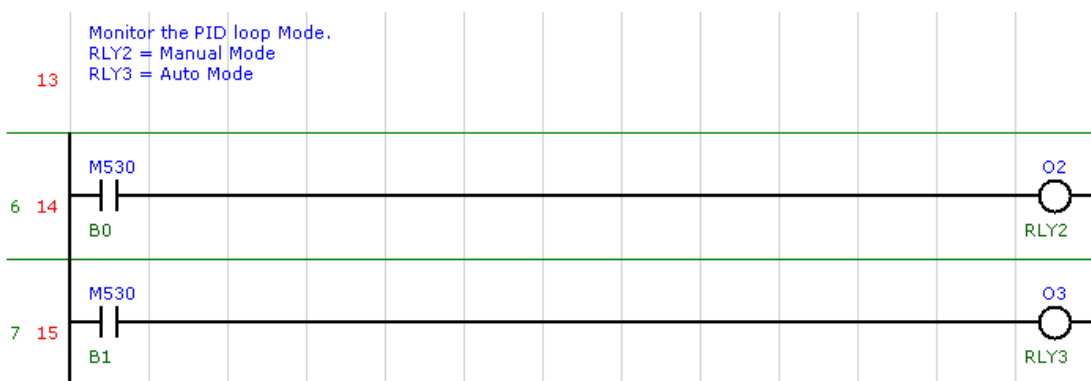
You can change to manual mode by writing a 1 to bit0 in the status register M486. The PID loop software will check this register and if it is possible to change to manual mode, then bit0 in the Mode register M530 will be on and bit1 in the Mode register will be off. When the mode change is complete then bit0 will be cleared in the status register back to zero.

You can change to auto mode by writing a 1 to bit1 in the status register M486. The PID loop software will check this register and if it is possible to change to auto mode, then bit1 in the Mode register M530 will be on and bit0 in the Mode register will be off. When the mode change is complete then bit1 will be cleared in the status register back to zero.

The following sample program shows you how you can change between the auto and manual modes in ladder software.



The following ladder logic lines shown how you can read back mode register to check which mode the PID loop is in.



### **6.5.9 PID Mode register bit 2 – Control Output Format Unsigned/Signed**

You can configure the control output value to be a unsigned or signed value.

For unsigned the output value will range from 0 to 4095 (12 bit) or 0 to 32767 (15 bit).  
For signed the output value will range from -4096 to 4095 (12 bit) or -32768 to 32767 (15 bit).

You need to set this bit to match the format of the output device. In most cases you will use the unsigned mode.

### **6.5.10 PID Mode register bit 3 – Control Output Format 12/15 Bits**

You can configure the control output value to be a 12 bit value or a 15 bit value. You need to set this bit to match the format of the output device. In most cases you will use the 12 bit mode.

### **6.5.11 PID Mode register bit 4 – PV Linear/Square Root**

In some applications it is necessary to perform the square root function on the PV before it is compared with the setpoint. Typically this would be when the PV is coming from a flow sensor which gives an output which is approximately the square of the flow. By performing the square root function the PV is effectively linearised.

In most applications this function is not used so bit 4 is cleared to zero.

### **6.5.12 PID Mode register bit 5 – Control Output Normal/Inverted**

In most applications such as heating, the control output will increase to raise the temperature of the process. This is called a normal output and an unsigned 12 bit output would go from 0 to 4095.

In some instances such as a cooling process an increase in the control output is required to cool the process. In this case the control output value must be inverted and an unsigned 12 bit output would go from 4095 to 0.

It is very important that you determine whether you have a normal or inverted process as it is impossible for the PID loop to control an incorrectly configured control output. You can test this by increasing the output value manually and checking if the PV increases or decreases. If the PV decreases you have an inverted process.

### **6.5.13 PID Mode register bit 6 – Error Linear/Squared**

When selected the error squared function affects the control output by reducing its response to smaller error values and maintaining its response to bigger error values. This function can help if the PV signal is noisy as it will reduce the effect of the error caused by the noise. It is suggested that you try the loop initially without this function enabled.

#### **6.5.14 PID Mode register bit 7 – Integral Gain Seconds/Minutes**

The integral gain  $K_I$  is a value in time. This bit in the mode register is used to determine if the gain value used is in seconds or minutes. For very slow processes it is necessary to have a high value  $K_I$ , so you will select minutes. Remember that since the integral gain is an inverse value, the higher the gain the less the effect it has on the PID loop calculations.

#### **6.5.15 PID Mode register bit 8 – Integrator Limit**

One of the problems associated with PID controllers is that if there is a problem with the process and the control output reaches one of the limits, the integral term will continue to increase or decrease to try to control the loop. When the process is returned to normal, the integral term will then take a long time to return to its correct operating value, resulting in the process undergoing violent fluctuations before the loop settles again. If the Integrator Limit is enabled, the integrator will be disabled as soon as the control output value reaches one of the limits. When the process is corrected, the integral term will quickly respond and the loop will settle more quickly. This function is commonly known as Anti-Windup.

#### **6.5.16 PID Mode register bit 9 – Bumpless Transfer**

When the PID loop is changed from manual mode to auto mode, the loop will start to try to control the process by changing the control output value according to the value in the setpoint register. The integral term starts at zero and it could take a long time for this term to reach the correct operating value which results in a large overshoot on startup. To avoid this problem the output offset term in the PID algorithm is pre-loaded with the output value.

If the PV is not equal to the SP when the mode is changed, then the PID loop will still have to ramp up or down to get to the correct operating point. If the bumpless transfer bit 9 is set, then the SP is made equal to the PV and the loop should go into auto mode without any change in the process. The operator must then change the SP slowly until the process has reached the correct operating point.

#### **6.5.17 PID Mode register bit 10 – PI/PID Mode**

There are cases where the process only requires PI operation. When bit 10 is cleared the PID loop algorithm will be setup to only do the Proportional and Integral terms and not include the Differential term.

If you are tuning the loop manually, then a value of zero written into the  $K_D$  register will result in the loop calculations only doing the P and I terms.

If you write a value of zero to the  $K_I$  register, then this term will also be ignored and the PID loop calculations will only do the P term.

#### **6.5.18 PID Mode register bit 11 – Open Loop/Closed Loop Auto Tuning**

There are two modes that can be used to auto tune the loop. Refer to the auto tuning chapter for the description on these options.

### 6.5.19 PID Mode register bit 12 – PV Value Alarm Enable

If this bit is set, then the PID loop software will check the PV against the PV alarm limits. If the PV is outside of the alarm limits then the PV alarm bits 9, 10, 11 or 12 will be set in the status register M486.

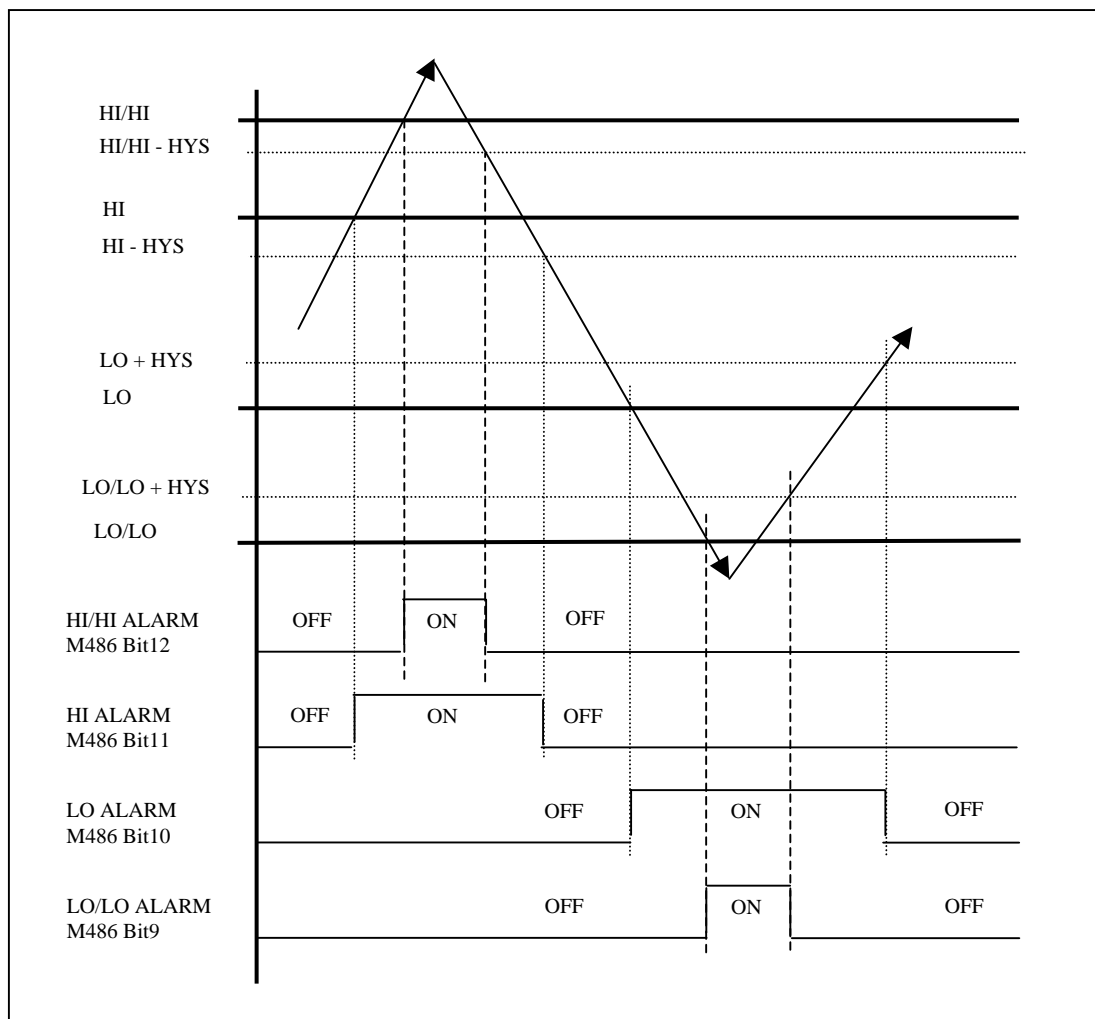
The PV Value alarm limits are programmed into the following registers:

- PV LO/LO Alarm Limit - M535
- PV LO Alarm Limit - M536
- PV Hi Alarm Limit - M537
- PV Hi/Hi Alarm Limit - M538

It is important to ensure that the PV LO Limit is greater than the PV LO/LO Limit value, that the PV Hi Limit is greater than the PV LO Limit value, and that the PV Hi/Hi Limit is greater than the PV Hi Limit value.

You must also program the PV Alarm Hysteresis value into the memory location M542.

The alarm works as follows:



## 6.5.20 PID Mode register bit 13 – PV Deviation Alarm Enable

If this bit is set, then the PID loop software will check the PV deviation from the SP against the PV deviation alarm limits. If the PV is outside of the alarm limits then the PV alarm **bits 13 or 14** will be set in the status register **M486**.

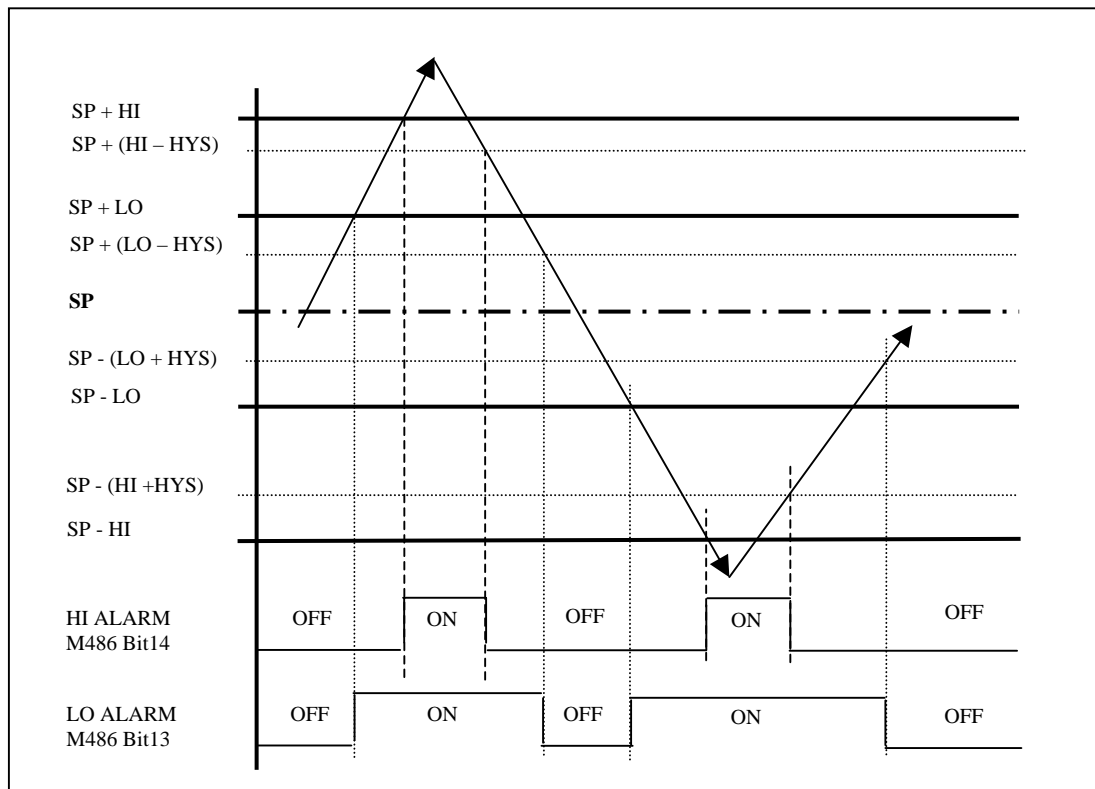
The PV deviation alarm limits are programmed into the following registers:

PV LO deviation Alarm Limit - **M539**  
 PV HI deviation Alarm Limit - **M540**

It is important to ensure that the PV LO Limit is less than the PV HI Limit value.

You must also program the PV Alarm Hysteresis value into the memory location **M542**.

The alarm works as follows:



### 6.5.21 PID Mode register bit 14 – PV Rate of Change Alarm Enable

If this bit is set, then the PID loop software will check the rate of change of the PV against the PV rate of change alarm value. If the rate of change of the PV is outside of the alarm limits then the PV alarm **bit 15** will be set in the status register **M486**.

The PV rate of change alarm limit is programmed into the following registers:

PV rate of change Alarm Limit - **M541**

The value you program into this register is the maximum allowed change in PV in one sample time.

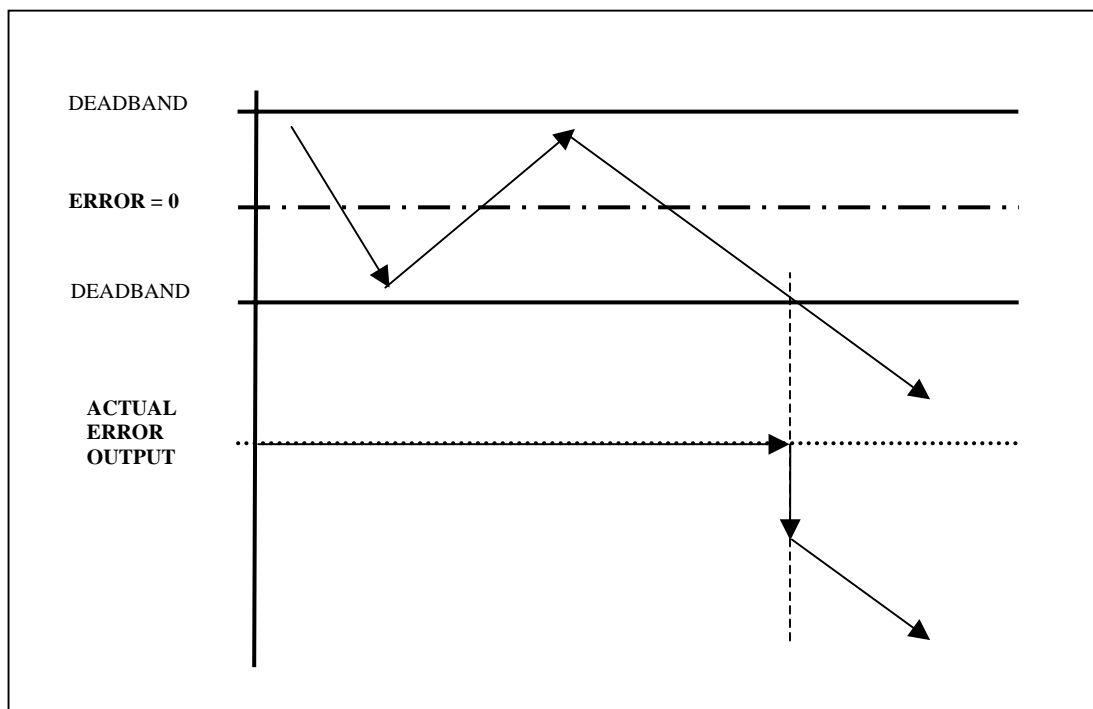
Note that this alarm does not use the PV Alarm Hysteresis value.

### 6.5.22 PID Error Deadband

The error deadband function is used to prevent changes in the control output when the error (SP-PV) is less than the deadband value which is programmed into register **M543**. The deadband applies to both positive and negative errors and is symmetrical.

If the error value is greater than the deadband value then this error value is put into the PID loop calculations.

If the error value is less than the deadband value then this error value is made equal to zero and this is put into the PID loop calculations.



### 6.5.23 PID Derivative Gain Limit

The derivative term of the PID algorithm can cause violent swings in the control output. The gain limit function will restrict the derivative term to be less than the limit setup in register **M544**. If this value is zero then the function is disabled.

## 6.6 Tuning the PID loop

Loop tuning is a very important step in setting up the PID loop. The main aim is to adjust the sample time and loop gains to get the optimum performance of the system.

There are several methods for tuning a PID loop. The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and the response speed of the system. If the system can be taken offline, the best method for auto tuning involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters. This technique is called the Open Loop Tuning method. The other auto tuning method used is called the Closed Loop Tuning method where the control output is switched between its lower and upper limits and the cycle time is measured to determine the control parameters.

This section discusses the different ways of tuning the PID loop and shows you how to setup the four remaining registers:

Sample Time - **M531**  
P Gain ( $K_P$ ) - **M532**  
I Gain ( $K_I$ ) - **M533**  
D Gain ( $K_D$ ) - **M534**

### 6.6.1 Manual Tuning

#### 6.6.1.1 Determining the Loop Sample Time

The first step in the manual tuning process is to determine the rate at which the PLC runs the PID loop calculations.

The following points are important in determining the loop sample time:

1. The sample time must not be faster than the sample rate of the analog inputs.
2. The sample rate should be about ten times faster than the response time of the process. The response time is measured by inducing a step response into the process.

From the points above, you can see that you must make sure that the sample rate of the analog inputs is fast enough for the process you are wanting to control.

Follow the following steps to measure the response time:

1. To measure the response time you should make sure that the PID loop is in manual mode. This will stop the PID loop calculations from being performed and this allows you to control the loop directly by writing to the control output register.
2. Write a value to the control output register **M485** which will put the PV value in register **M481** into the middle of a safe working range. Write down this PV value once the process has settled (PV1).

3. Now you can put a step change into the control output register of about 10% of the output range. If the range is 0-4095, and the control output is currently at 2000, then you will write a new value into the control register of 2409.
4. Wait for the PV to settle and note the new PV value (PV2).
5. You can now calculate the 10% and 90% PV values as follows:  
 $10\% \text{ PV} = \text{PV1} + (\text{PV2} - \text{PV1}) \times 0.1$  and  $90\% \text{ PV} = \text{PV1} + (\text{PV2} - \text{PV1}) \times 0.9$
6. Put the initial control output value back into the control output register (2000) and wait for the process to settle again.
7. Now induce the 10% step change again and start monitoring the PV value.
8. When the PV reaches the 10%PV point note the starting time, and when the PV reaches the 90%PV point note the end time.
9. The difference in the two time readings is the rise time of the process. You can now calculate the sample time by dividing this time by 10.
10. You can now program the sample time into the register in memory **M531**. You must remember that the resolution of the value in M531 is 0.01 seconds so if you have a sample time of 9 seconds, the value you write into the register is 900.

### 6.6.1.2 Determining the Loop Gains

The next step in the manual tuning process is to determine the gains for each of the P, I and D terms.

Follow the following steps to setup these gains:

1. Make sure that the PID loop is in manual mode.
2. Write a small  $K_P$  gain value to the register **M532**. A starting value of 10 (0.10) should be sufficient.
3. Write a zero to the  $K_I$  and  $K_D$  registers **M533** and **M534**. This will disable these terms.
4. Write a control output to put the PV in the center of a safe working area. You can use the same value as you used above.
5. When the PV has stabilized, write the PV value that you read in M481 to the setpoint register **M480**.
6. You can now set **bit1** in the status register **M486** to put the PID loop into auto mode.
7. This bit will auto clear when the mode has been changed from manual to auto. You can read back **bit1** in the mode register **M530** to verify that the mode has changed.
8. The PID loop should now start controlling the output to ensure the PV equals the SP. Increase the SP by a small amount and you should see the PV increasing up to this value. If it does not move then increase the value of  $K_P$  until the PV changes.
9. You can now add a small amount of integral gain  $K_I$ . Start with a value of 1000.(100.0) Remember that high values of  $K_I$  result in a small effect on the calculations.
10. You can now put a step change into the SP register of about 10% and monitor the PV value. If the PV overshoots the SP then you need to reduce the gains. If the PV moves up very slowly towards the SP then you can add more gain.
11. Continue with this cycle of step change and gain adjustment until you are satisfied with the response of the PV.
12. Now you can add a small amount of differential gain  $K_D$  by putting a small value into the register **M534**.
13. Redo the step change test and you should see that the PV rises faster after the step change due to the gain  $K_D$ . If you get too much overshoot then you need to reduce this gain value.

## 6.6.2 Automatic Tuning

### 6.6.2.1 Open Loop Method

The open loop automatic tuning method puts a 10% step change on the control output and monitors the PV. At the end of the tune period the sample time and the gains are automatically written to the respective registers.

Follow the following steps to perform the open loop auto tune sequence:

1. Make sure that the PID loop is in manual mode.
2. Write a value to the control output register **M485** which will put the PV value in register **M481** into the middle of a safe working range and wait for the PV to settle.
3. Make sure that the **bit11** in the mode register **M530** is zero. This selects open loop tuning.
4. You can now set **bit2** in the status register **M486** to put the PID loop into auto tune mode.
5. The control output will change by 10% and you will see that the PV starts to increase.
6. Monitor the auto tune **bit2** in the status register **M486**. You will know that the auto tune is complete as soon as the bit is set back to zero by the PID software. The sample time and gains will now be updated.
7. You can now set **bit1** in the status register **M486** to put the PID loop into auto mode.

### 6.6.2.2 Closed Loop Method

The closed loop automatic tuning method switches the control output between the minimum and maximum values for 3 cycles. The PID software monitors the PV and switches the control output as soon as the PV passes the SP. At the end of the tune period the sample time and the gains are automatically written to the respective registers.

Follow the following steps to perform the closed loop auto tune sequence:

1. Make sure that the PID loop is in manual mode.
2. Write a value to the setpoint SP register **M480** which will correspond to the normal operating setpoint.
3. Make sure that the **bit11** in the mode register **M530** is set. This selects closed loop tuning.
4. You can now set **bit2** in the status register **M486** to put the PID loop into auto tune mode.
5. The control output will change between the minimum and maximum values as the PV crosses the SP.
6. Monitor the auto tune **bit2** in the status register **M486**. You will know that the auto tune is complete as soon as the bit is set back to zero by the PID software. The sample time and gains will now be updated.
7. You can now set **bit1** in the status register **M486** to put the PID loop into auto mode.

### 6.6.3 Analog Filter

In real life applications electrical interference can be induced into the analog input signals resulting in a PV which is very noisy. This noise can cause a problem with loop stability and as a result it is advised you use an analog filter to filter out the noise.

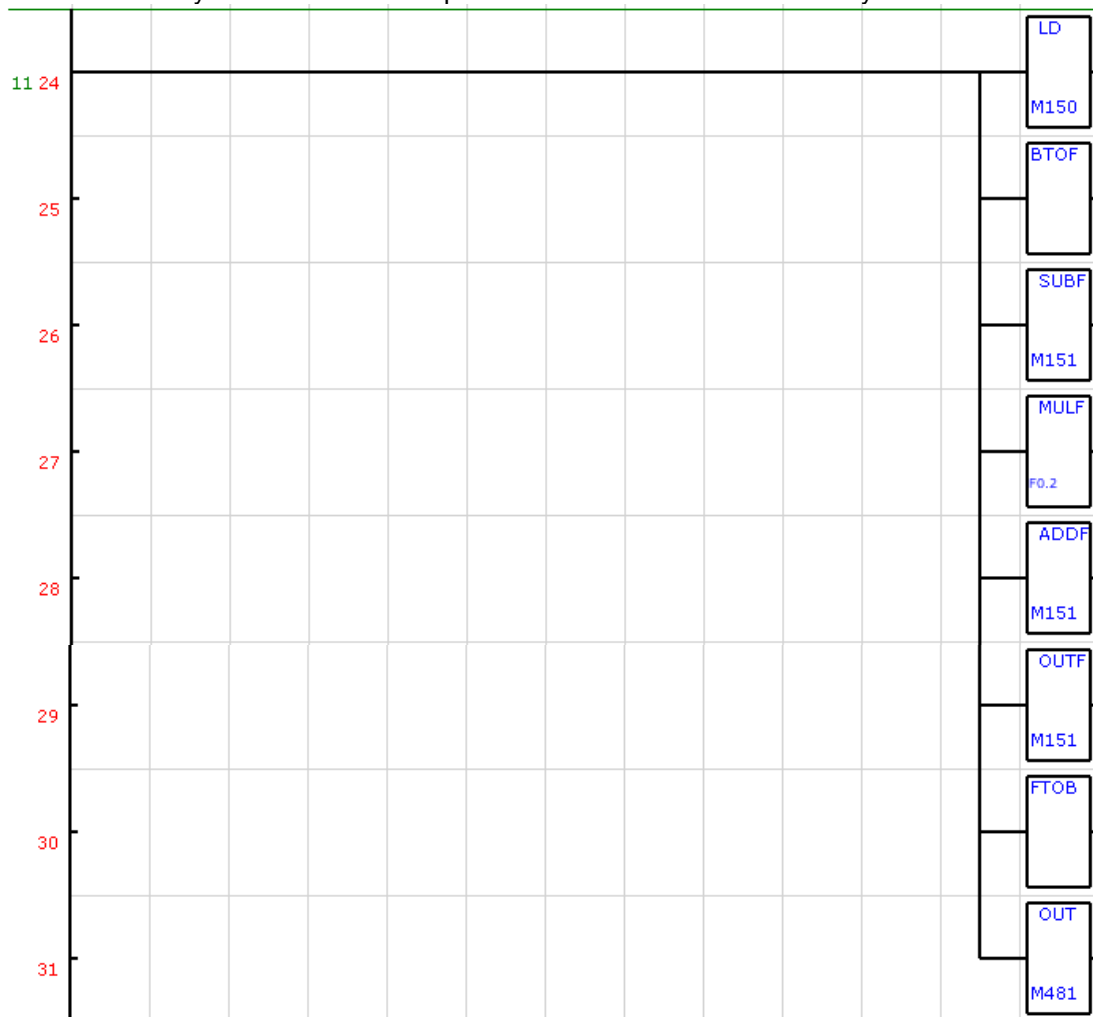
More important is that the PV value should be as noise free as possible during the auto tuning process as electrical noise could cause the PID software to make incorrect decisions on the reaction of the process being tuned, thus producing less than optimum gain factors.

The following low pass filter can be implemented in ladder logic to filter out a large part of this noise. The equation describing the filter is as follows. The smaller the gain factor (K) the lower the filter cut-off point and the slower the response.

$$F_t = K \times (PV_t - F_{t-1}) + F_{t-1}$$

- $F_t$  - Filter output (PV out)
- $PV_t$  - Filter input (PV in)
- $F_{t-1}$  - Last Filter output
- $K$  - Gain Factor (0.1 to 1.0)

The ladder diagram below shows how the filter can be implemented. The analog value is written to memory M150. The filter output is then written to the PV memory M481.



## 7. Modbus Memory Map ( MODULE TYPE = 42)

The data in the PL410 is stored in registers. These registers are accessed over the network using the MODBUS RTU communication protocol.

There are 4 types of variables which can be accessed from the module. Each module has one or more of these data variables.

<u>Type</u>	<u>Start Address</u>	<u>Variable</u>
1	00001	Digital Outputs
2	10001	Digital Inputs
3	30001	Input registers (Analog)
4	40001	Output registers (Analog)

Modbus Address	Mem Addr	Register Name	Low Limit	High Limit	Access	Comments
10017	1.1	Digital Input 1	0	1	R	Status of Digital Inputs 1.
"	"	"	"	"	"	"
10032	1.16	Digital Input 16	0	1	R	Status of Digital Inputs 16.
00033	2.1	Digital Output 1	0	1	R/W	Status of Digital Outputs 1.
"	"	"	"	"	"	"
00048	2.16	Digital Output 16	0	1	R/W	Status of Digital Outputs 16.
00049	3.1	Timer 1	0	1	R/W	Status of Timer 1.
"	"	"	"	"	"	"
00112	6.16	Timer 64	0	1	R/W	Status of Timer 64.
00113	7.1	Counter 1	0	1	R/W	Status of Counter 1.
"	"	"	"	"	"	"
00176	10.16	Counter 64	0	1	R/W	Status of Counter 64.
00177	11.1	Control Relay 1	0	1	R/W	Status of Control relay 1.
"	"	"	"	"	"	"
00240	14.16	Control Relay 64	0	1	R/W	Status of Control relay 64.
00241	15.1	System Relay 1	0	1	R/W	Status of System relay 1.
"	"	"	"	"	"	"
00272	16.16	System Relay 32	0	1	R	Status of System relay 32.
30001	0	S/W Version / Module Type	N/A	N/A	R	High Byte = Software Version Low Byte = 42
30002	1	Digital Inputs	N/A	N/A	R	Digital Inputs in 16 bits.
40003	2	Digital Outputs	N/A	N/A	R/W	Digital Outputs in 16 bits.
40004	3	Timer Status	N/A	N/A	R/W	Timer Status 16 – 1
40005	4	Timer Status	N/A	N/A	R/W	Timer Status 32 – 17
40006	5	Timer Status	N/A	N/A	R/W	Timer Status 48 – 33
40007	6	Timer Status	N/A	N/A	R/W	Timer Status 64 – 49
40008	7	Counter Status	N/A	N/A	R/W	Counter Status 16 – 1
40009	8	Counter Status	N/A	N/A	R/W	Counter Status 32 – 17

40010	9	Counter Status	N/A	N/A	R/W	Counter Status 48 – 33
40011	10	Counter Status	N/A	N/A	R/W	Counter Status 64 – 49
40012	11	Control Relay	N/A	N/A	R/W	Control Relay 16 – 1
40013	12	Control Relay	N/A	N/A	R/W	Control Relay 32 – 17
40014	13	Control Relay	N/A	N/A	R/W	Control Relay 48 – 33
40015	14	Control Relay	N/A	N/A	R/W	Control Relay 64 - 49
40016	15	System Relay	N/A	N/A	R/W	System Relay 16 – 1
40017	16	System Relay	N/A	N/A	R/W	System Relay 32 – 17
-	17	-	N/A	N/A	-	Do not use – System only
-	18	-	N/A	N/A	-	Do not use – System only
40020	19	Timer 1 Value	0	65535	R/W	Timer range 0 to 65535.
“	“	“	“	“	“	“
40083	82	Timer 64 Value	0	65535	R/W	Timer range 0 to 65535.
40084	83	Counter 1 Value	0	65535	R/W	Counter range 0 to 65535.
“	“	“	“	“	“	“
40147	146	Counter 64 Value	0	65535	R/W	Counter range 0 to 65535.
40148	147	User Memory	0	65535	R/W	0 to 65535.
“	“	“	“	“	“	“
40519	518	User Memory	0	65535	R/W	0 to 65535.
40520	519	Baud rate Progport	9600	19200	R/W	Default = 19200
40521	520	ID Prog port	0	255	R/W	Default = 1
40522	521	Number PID Loops	0	4	R/W	0 = None
40523	522	User EEPROM	0	65535	R/W	User EEPROM
“	“	“	“	“	“	“
40619	618	User EEPROM	0	65535	R/W	User EEPROM
40620	619	Seconds	0	59	R/W	RTC Seconds
40621	620	Minutes	0	59	R/W	RTC Minutes
40622	621	Hours	0	23	R/W	RTC Hours
40623	622	Day	1	7	R/W	RTC Day
40624	623	Date	1	31	R/W	RTC Date
40625	624	Month	1	12	R/W	RTC Month
40626	625	Year	0	100	R/W	RTC Year
40627	626	User BBRAM	0	65535	R/W	User BBRAM
“	“	“	“	“	“	“
40639	638	User BBRAM	0	65535	R/W	User BBRAM

## 8. Ladder Logic Function Blocks

The function blocks supported by the PL410 are listed below:

<b>PL410 Function Blocks</b>	
<b>Function</b>	<b>Function Block Description</b>
Timer 0.1Sec	Single input timer with 0.1 Second time base. The timer will run as long as the input is on. The timer will be reset to zero when the input is off.
Timer 0.01Sec	Single input timer with 0.01 Second time base. The timer will run as long as the input is on. The timer will be reset to zero when the input is off.
TimerA 0.1Sec	Accumulating timer with 0.1 Second time base. The timer will run as long as the input is on and stops when the input is removed. The timer will continue when the input is on again. The timer will be reset to zero when the reset input is on.
TimerA 0.01Sec	Accumulating timer with 0.01 Second time base. The timer will run as long as the input is on and stops when the input is removed. The timer will continue when the input is on again. The timer will be reset to zero when the reset input is on.
Counter	Up counter with reset input. The counter will count up when the count input goes from off to on. The counter will be reset to zero when the reset input is on. The counter output will go on when the count value is greater or equal to the preset value. The counter memory is addressed as the counter number + an offset
Counter Up/Dn	Up/Down counter with reset input. The counter will count up when the Up count input goes from off to on. The counter will count down when the Down count input goes from off to on. The counter will be reset to zero when the reset input is on. The counter output will go on when the count value is greater or equal to the preset value. The counter memory is addressed as the counter number + an offset of 16, so for example the value for counter 1 is in memory 17
NOP	This is a no operation function.
END	Placing this output function in the ladder program will indicate the end of the program. Any ladder after this function will not be run.
LD	Load the accumulator from memory(M) or with a constant(K).
LDD	The Load Double loads the accumulator with a 32 bit value from memory(M) or with a constant(K). The memory used is the two consecutive 16 bit memory locations, M & M+1.
LDF	The Load Float loads the accumulator with a float value from memory(M) or with a constant(F). The memory used is the two consecutive 16 bit memory locations, M & M+1.
OUT	Outputs the accumulator to memory(M).
OUTD	Outputs the 32 bit accumulator to two consecutive memory locations, M & M+1.
OUTF	Outputs the float accumulator to two consecutive memory locations, M & M+1.
AND	AND the accumulator with memory(M) or with a constant(K).
ANDD	AND the 32 bit accumulator with memory(M) or with a constant(K). The memory used is the two consecutive 16 bit memory locations, M & M+1.
OR	OR the accumulator with memory(M) or with a constant(K).
ORD	OR the 32 bit accumulator with memory(M) or with a constant(K). The memory used is the two consecutive 16 bit memory locations, M & M+1.
XOR	Exclusive OR the accumulator with memory(M) or with a constant(K).

<b>PL410 Function Blocks</b>	
<b>Function</b>	<b>Function Block Description</b>
XORD	Exclusive OR the 32 bit accumulator with memory(M) or with a constant(K). The memory used is the two consecutive 16 bit memory locations, M & M+1.
CMP	Compare the accumulator lower 16 bits with memory(M) or with a constant(K). If the value in the accumulator is less than the value in memory/constant then system bit S6 is turned on. If the value in the accumulator is equal to the value in memory/constant then system bit S7 is turned on. If the value in the accumulator is greater than the value in memory/constant then system bit S8 is turned on.
CMPD	Compare the 32 bit accumulator with memory(M) or with a constant(K). If the value in the accumulator is less than the value in memory/constant then system bit S6 is turned on. If the value in the accumulator is equal to the value in memory/constant then system bit S7 is turned on. If the value in the accumulator is greater than the value in memory/constant then system bit S8 is turned on.
CMPF	Compare the 32 bit accumulator with memory(M) or with a constant(F). If the value in the accumulator is less than the value in memory/constant then system bit S6 is turned on. If the value in the accumulator is equal to the value in memory/constant then system bit S7 is turned on. If the value in the accumulator is greater than the value in memory/constant then system bit S8 is turned on.
ADD	Add the memory(M) or constant(K) to the accumulator. The result is stored in the accumulator.
ADDD	Add the memory(M) or constant(K) to the 32 bit accumulator. The result is stored in the accumulator. The memory used is the two consecutive 16 bit memory locations, M & M+1.
ADDF	Add the memory(M) or constant(F) to the float accumulator. The result is stored in the float accumulator. The memory used is the two consecutive 16 bit memory locations, M & M+1.
SUB	Sub the memory(M) or constant(K) from the accumulator. The result is stored in the accumulator
SUBD	Sub the memory(M) or constant(K) from the 32 bit accumulator. The result is stored in the accumulator. The memory used is the two consecutive 16 bit memory locations, M & M+1.
SUBF	Sub the memory(M) or constant(F) from the float accumulator. The result is stored in the float accumulator. The memory used is the two consecutive 16 bit memory locations, M & M+1.
MUL	Multiply the accumulator with the memory(M) or constant(K). The result is stored in the accumulator
MULD	Multiply the 32 bit accumulator with the memory(M) or constant(K). The result is stored in the accumulator. The memory used is the two consecutive 16 bit memory locations, M & M+1.
MULF	Multiply the float accumulator with the memory(M) or constant(F). The result is stored in the float accumulator. The memory used is the two consecutive 16 bit memory locations, M & M+1.
DIV	Divide the accumulator by the memory(M) or constant(K). The result is stored in the accumulator.
DIVD	Divide the 32 bit accumulator by the memory(M) or constant(K). The result is stored in the accumulator. The memory used is the two consecutive 16 bit memory locations, M & M+1.
DIVF	Divide the float accumulator by the memory(M) or constant(F). The result is stored in the float accumulator. The memory used is the two consecutive 16 bit memory locations, M & M+1.
INC	Increment the memory(M). The result is stored in the memory(M)

<b>PL410 Function Blocks</b>	
<b>Function</b>	<b>Function Block Description</b>
INCD	Increment two consecutive memory(M) locations. The result is stored in the memory M & M+1.
DEC	Decrement the memory(M). The result is stored in the memory (M).
DECD	Decrement two consecutive memory(M) locations. The result is stored in the memory M & M+1.
INV	Invert the bits in the accumulator
MOV	Moves a variable in a memory location to a new location. The accumulator must already contain the address of the memory location to be moved.
SHL	The bits in the accumulator are shifted left by the memory(M) or constant(K). The lower bits are filled with zeros.
SHR	The bits in the accumulator are shifted right by the memory(M) or constant(K). The upper bits are filled with zeros.
CALL	This function is used to call a subroutine. The constant(k) is the label of the subroutine.
SUBR	This function is the start of a subroutine. The constant(k) is the label of the subroutine which is called by the call function.
RET	This function must be placed at the last line of a subroutine. The function can also be used in the subroutine for a conditional return.
RAND	A random number from 0 to 100 is placed in the accumulator
ACOSF	Arc Cosine of float accumulator
ASINF	Arc Sine of float accumulator
ATANF	Arc Tangent of float accumulator
COSF	Cosine of float accumulator
SINF	Sine of float accumulator
TANF	Tangent of float accumulator
SQRTF	Square Root of float accumulator
BTOF	The value in the 32 bit accumulator is converted to a float value and stored in the float accumulator.
FTOB	The value in the float accumulator is converted to a binary number and stored in the 32 bit accumulator.
RADF	The Radian of the float accumulator.
DEGF	The degrees of the float accumulator.
LOGF	The log of the float accumulator.
EXPF	The exponential of the float accumulator
PWRF	The power of the float accumulator.
COMM	Communications function. Enter a parameter number to select the data to be saved. 0 = Port Number (default = 1) 1 = Protocol (default = 0) 2 = Slave network ID 3 = PLC Memory Address 4 = Range 5 = Slave Address 6 = Timeout 7 = Function

