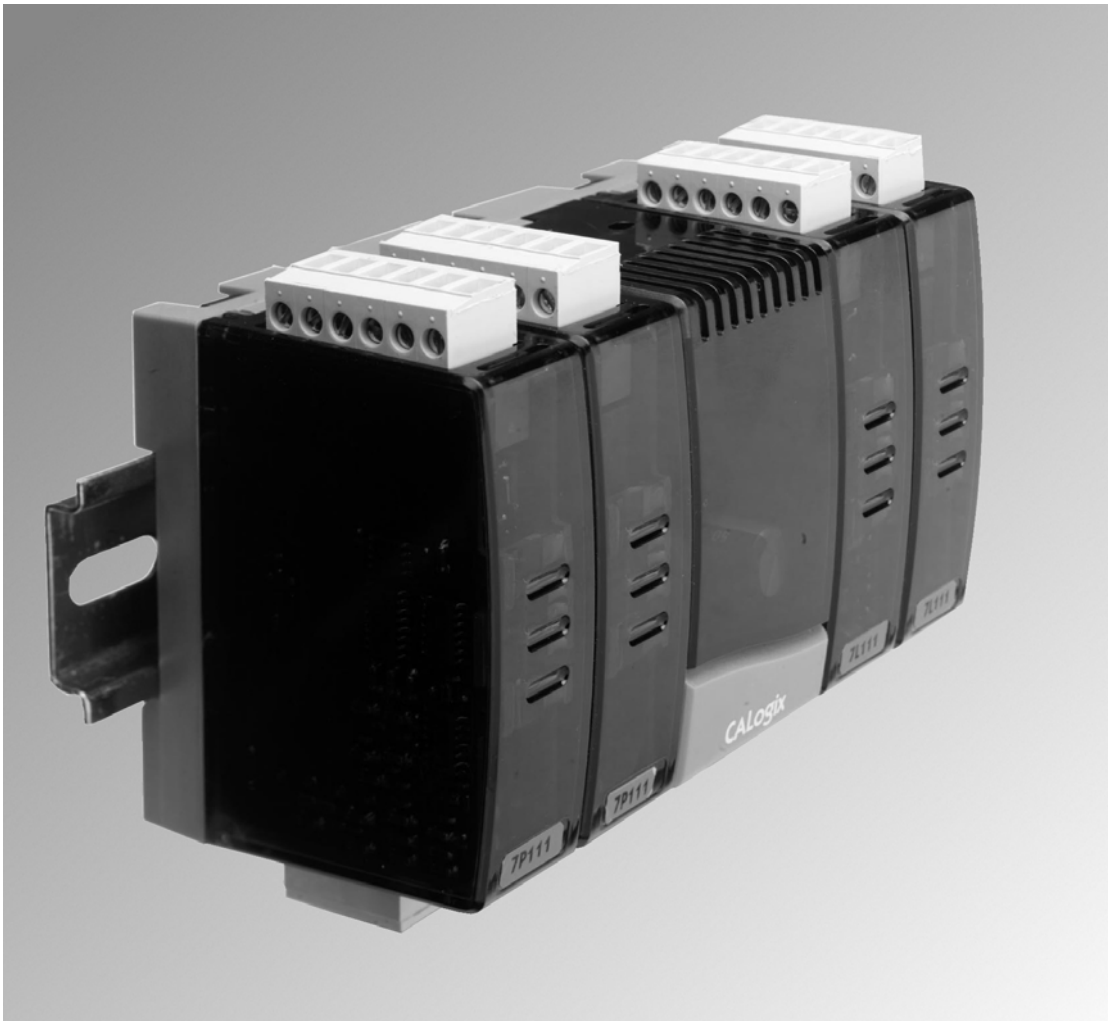


CALogix
MODBUS RTU
programmer's guide

CAL Controls

1. Introduction

This document describes the interaction between a CALogix controller and a PC / PLC attached to the bus acting in a master command mode.

The CALogix controller performs NO verification on each parameter value supplied to it, therefore the PC / PLC application must ensure values outside the working limits of the controller are not supplied. The controller will always assume that the values received have been checked against the published limits and unchecked parameters outside these limits, could have an adverse effect on the controller.

1.1 Using the guide

Convention used with this guide are as follows: -

Hexadecimal Numbers	These are preceded by the characters '0x' (e.g. **0x03D6** = value 3D6 Hexadecimal)
Decimal Integers	These are written normally with no decimal point (e.g. **234**). Decimal equivalents are shown in brackets after a Hexadecimal number.
Floating point numbers	Written normally with a decimal point. (e.g. **234.0**)
Read Only Register	Indicated using symbol: - **[R]**
Write Only Register	Indicated using symbol: - **[W]**
Read-Write Register	Indicated using symbol: - **[RW]**
Modbus request & response	Shown using separated digits and the symbol: - **=>** (e.g. 01 03 07 CC 00 01 => 01 03 00 00)
Modbus CRC checksum	shown by **[CRC]** (2 x bytes).

1.2 Communications Settings

The CALogix controller uses RS485 full duplex serial communications to ensure high noise immunity and multi-drop capability. On power up CALogix detects the network speed and will automatically adjust its baud rate accordingly. See section 4.5 for allowable baud rate values.

The serial format expected is: -

Parity	Data Bits	Stop Bits
None	8	1

N.B. This cannot be changed.

1.3 WARNING

As with any computer system writing to any unauthorised Register address could inevitably cause malfunction and may put the instrument in an indeterminate or unrecoverable state. It is the users responsibility to ensure correct use.

2 Summary of Modbus RTU

The Modbus RTU protocol is a single master, multi-slave type command system. It provides an efficient coding method for reading and writing to controllers in an industrial environment.

2.1 Message Construction

The Modbus RTU message format differs from other Modbus command structures in that the data is sent as a binary value, byte by byte. In this guide where values in hex (or decimal) are shown the value must be converted to its binary equivalent before transmission.

N.B. It is a common mistake to try to send the individual ASCII characters of the value as part of the message format.

All Modbus requests and responses are designed in such a way that the recipient can verify that a message is complete. For functions where the request and response are of fixed length, the function code alone is sufficient. For functions messages carrying a variable amount of data in the request or response, the data portion will be preceded by a byte count.

Modbus message format will in general have the following fields : -

Slave Address:	This is a 1 Byte value with range 1 to 247. **N.B.** The CALogix units do not handle the value zero that is traditionally used for broadcast messages.
Function Code:	This is a 1 Byte value that indicates the Function required (e.g. read register, write register).
Register Address:	This is a 2 Byte (word) value that identifies the Controller register that requires reading or modifying.
Register Count:	This is a 2 Byte register word-count value with range 1 to 100, only used for Functions that write to a variable number of Registers.
Byte Count:	This is a 1 Byte count value with range 1 to 255, only used for functions that have a variable data payload.
Data Payload:	The data read, or data written to the controller. See Section **3.** CALogic data formats for more details.
CRC Checksum:	This is a 2 Byte (word) CRC Checksum. See Section, **2.3** CRC calculation for details.

2.1.1 Modbus Data Representation

Modbus uses a 'big-endian' representation for address and data items. This means that when a numerical quantity larger than a single byte is transmitted, the MOST significant byte is sent first. So for example

16 – bits	0x1234	would be	0x12,	0x34		
32 – bits	0x12345678	would be	0x12,	0x34,	0x56,	0x78

2.1.2 Modbus Address Representation

All Modus Register Address items detailed within this manual are given as 16 – bit values.
Since Modbus uses a big-endian convention the Register Address specified in the tables etc will point to the most significant byte of the (first) word register.
This means that ALL Register Addresses must have one taken off their value before encoding into the Modbus message. (**N.B.** This must be done before the CRC is calculated.)

2.1.3 Exception Handling

Modbus has a defined set of exception codes to be returned by Slave controllers in the event of problems.

All exceptions are signalled by adding 0x80 to the function code of the request, and following this byte by a single 'exception code' byte indicating the reason for command failure

The list of exception codes returned by the CALogix controller is shown as follows: -

Code	Name	Description
01	ILLEGAL FUNCTION	Returned for an unrecognised Function code.
02	ILLEGAL REGISTER ADDRESS	Returned if the Register address received is not an allowable address for the slave controller.
03	ILLEGAL DATA VALUE	Returned if the value contained in the 'data payload' section of the MODBUS command is not allowable for the slave. **N.B.** This indicates a fault in the structure of the data payload and specifically does NOT mean the data item submitted for storage has a value outside the expectations of the controller.
04	SLAVE DEVICE FAILURE	Indicates that the request was understood but the SLAVE could not comply with the request for an unspecified reason. (Busy)

Example: -

Request: - read 1 register at address 0x1234. response: - exception type 2 – 'illegal data address'.

01 03 12 34 00 01 [CRC] => ***01 83 02 [CRC]***

where: - ***'CRC'*** represents the 2-byte checksum, Slave address 01.

There are some exceptions where the Slave will not send any response.

Examples of these are listed below: -

1. Broadcast messages (one with Slave address 0).
2. Slave receives incomplete or corrupt message.
3. Slave receives full message but CRC is incorrect.

2.2 Function Codes

The Function codes are used to implement the commands available within the Modbus protocol.

The CALogix ***supported*** Function Codes are listed below: -

Function Code	Name	Description
0x03 (3 decimal)	Read multiple registers	Reads 1 or more words from a CALogix register.
0x04 (4 decimal)	Read input registers	Reads 1 or more words from a CALogix register
0x06 (6 decimal)	Write single register	Writes a single word of data to a CALogix register.
0x10 (16 decimal)	Write multiple registers	Writes 1 or more words to a CALogix register.

Note: All Functions not supported by CALogix will return the exception code: -

01 'ILLEGAL FUNCTION'.

The Function Code details are shown below: -

2.2.1 Read Multiple Registers

Request (outgoing message from Master)

Byte 0: Slave Address
Byte 1: Function Code = 03
Byte 2-3: CALogix Register Address
Byte 4-5: Word count (1-125)
Byte 6-7: CRC Checksum

Response (incoming message from Slave)

Byte 0: Slave Address
Byte 1: Function Code = 03
Byte 2: Byte count of response (B=2 x word count)
Byte 3-(B+1): CALogix Register values
Bytes (x2): CRC Checksum

Exceptions

Byte 1: Function Error Code = 83
Byte 2: exception code 01 or 02

Example

Read 1 register from Slave Address 1, at CALogix Register Address 0x07CB resulting in a value of 0001 hex.

01 03 07 CB 00 01 [CRC] => 01 03 02 00 01 [CRC] (All values in hex.)

2.2.2 Read Input Register

This follows the same format as Function code 03, Read Multiple Register above.
The Function code is 04 hex.

2.2.3 Write Single Register

Request (outgoing message from Master)

Byte 0:	Slave Address
Byte 1:	Function Code = 06
Byte 2-3:	CALogix Register Address
Byte 4-5:	Data Value (always 16 bit. N.B. bit/byte values are leading zero padded)
Byte 6-7:	CRC Checksum

Response (incoming message from Slave)

Byte 0:	Slave Address
Byte 1:	Function Code = 06
Byte 2-3:	CALogix Register Address
Byte 4-5:	CALogix Register values
Byte 6-7:	CRC Checksum

Exceptions

Byte 1:	Function Error Code = 86
Byte 2:	exception code 01 or 02

Example

Write 1 register to Slave Address 1, at CALogix Register Address 0x07CC with a value of 0017 hex.

01 06 07 CC 00 17 [CRC] => ***01 06 07 CC 00 17 [CRC]*** (All values in hex.)

2.2.4 Write Multiple Register

Request (outgoing message from Master)

Byte 0:	Slave Address
Byte 1:	Function Code = 10 hex
Byte 2-3:	CALogix Start Register Address
Byte 4-5:	Number of Registers to write, word count (1-100)
Byte 6:	Number of Bytes in Data Payload (B=2 x word count)
Byte 7-(B+6)	Data Payload (Register Values)
Byte (x2):	CRC Checksum

Response (incoming message from Slave)

Byte 0:	Slave Address
Byte 1:	Function Code = 10 hex
Byte 2-3:	CALogix Register Address
Byte 4-5:	Number of Registers written, word count
Byte 6-7:	CRC Checksum

Exceptions

Byte 1:	Function Error Code = 90 hex
Byte 2:	exception code 01 or 02

Example

Write to 1 complex register (32 bit), Slave Address 1, at CALogix Register Address 0x07CF with a value of 0000 hex.

01 10 07 CF 00 01 02 00 00 [CRC] => ***01 10 07 CF 00 01 [CRC]*** (All values in hex.)

2.3 CRC calculation

All Modbus RTU transfers are protected by utilising a cyclic redundancy checksum word, appended to the end of the message. The checksum sums all bytes of the binary formatted message, including the Slave address.

There are two ways to implement the CRC, one uses an iterative algorithm (shown below), and the other uses pre-computed lookup tables, which make for faster calculation.

For further details of the table-driven CRC algorithm see the Modicon ModBus Protocol Guide (PI-MBUS-300 Rev G, Nov 1994)

A step-by-step iterative method for calculating the CRC checksum is given below: -

1. Load a 16 bit register with 0xFFFF (all 1's). Call this the **CRC register**.
2. Exclusive OR the first 8 bits byte of the message with the low-order byte of the 16 bit CRC register, putting the result back into the **CRC register**.
3. Look at the Least Significant Bit of the **CRC register** and remember it. Call this the **LastBit.**
4. Shift the **CRC register** one bit right, putting 0 in the top bit.
5. If the **LastBit** was 1, Exclusive OR the **CRC register** with value 0xA001 hex (1010 0000 0000 0001).
6. Repeat steps 3, 4, 5 until 8 shifts have been performed.
7. Repeat from step 2 for the next byte of the message until all bytes have been processed.
8. The final contents of the **CRC register** is the CRC value to use.
9. When the CRC is placed in the message, the Least Significant Byte is sent first, then the Most Significant Byte.

2.3.1 CRC example in C

Assume the Modbus message to be transmitted is held in array: - *mess[ ]* (unsigned char).

```
unsigned short  crc;
unsigned short  thisbyte;
unsigned short  shift;
unsigned char   highbyte, lowbyte;
unsigned char   lastbit, i;

crc = 0xFFFF;
for( i=0; i<len(mess); i++ )
{
        thisbyte = mess[i];
        crc = crc ^ thisbyte;
        for( shift=1; shift<=8; shift++ )
        {
                lastbit = crc & 1;
                crc = (crc >> 1) & 0x7FFF;
                if( lastbit==1 )
                {
                        crc = crc ^ 0xA001;
                }
        }
}
highbyte = (crc>>8) & 0xFF;
lowbyte  = crc & 0xFF;
```

3 CALogix data formats

The CALogix controller utilises several data formats as listed below: -

Standard Data Types

Data Type	Size	Description	Example
Byte	*8 bits*	*values lie between 0 and 255*	*Output Power,* ***0 – 100%***
Word	*16 bits*	*values lie between 0 and 65536*	*Program Loop count* ***1 – 999***
Double Word	*32 bits*	*values lie between 0 and 4.2 thousand million*	*Program Segment time,* ***1 second – several hours***

Encoded Data Types

Data Type	Size	Description	Example
Encoded Boolean	*8 bits*	*values are 1 or 0*	*Output State,* ***Off or On***
Encoded Byte	*8 bits*	*values have special meanings N.B not all values might be used or be contiguous.*	*Program Mode,* ***Run, Halt, Stop***
Encoded Word	*16 bits*	*values generally lie between 0 and 65536 but require scaling to be interpreted correctly.*	*SP1(Heat) Integral Time,* ***1 to 1000 in 0.1 second steps***
Encoded Double Word	*32 bits*	*these values are used where a Encoded Word is not large enough to accommodate the range of data.*	*AutoTune Heat Quarter time 1,* ***1mS to several hours***
Float	*32 bits*	*these values are encoded using the IEEE single precision floating point format.*	*Process Variable.* ***123.456***

3.1 Standard Data Types

Standard data types are transmitted / received following the standard Modbus RTU formats: -

3.1.1 Byte Data

Bytes are always transmitted / received as a 16 bit value. When writing a byte value the Modbus function code 6 (Write Single Register) is used. To retrieve byte size information the function code 3 (Read Multiple Register) is used with a size count value of 1.
Both these Modbus function codes utilise a data payload of size 16 bits therefore the Most Significant Byte of the payload is always set to zero.

Example 1: - *Request Output Power setting, returning a value of 100%.*

01 03 08 A3 00 01 [CRC] => 01 03 02 ***00 64*** *[CRC]* (All values in hex.)

Example 2: - *Set Manual Output Power to 50%*

01 06 08 67 ***00 32*** *[CRC] => 01 06 08 67 00 32 [CRC]* (All values in hex.)

In the examples shown the data payload is highlighted.

3.1.2 Word Data

Words are always transmitted / received as per the same rules for Byte transfers.
All data payload values will be treated as *full* 16 bit values.
See above examples.

3.1.3 Double Word Data

Double Words (DWord) are always transmitted / received as a pair of 16 bit values. When writing DWord values the Modbus function code 10 (Write Multiple Register) should be used. This will ensure that 2 bytes are written in one transaction and thus eliminate the problem of split transfers having out-of-date values.
To retrieve DWord size information the function code 3 (Read Multiple Register) is used with a size count value of 2.

Example 1: - *Request AutoTune Heat approach time A, returning a value of 10000 mS.*

01 03 08 C7 00 02 [CRC] => 01 03 04 ***00 00 27 10*** *[CRC]* (All values in hex.)

In the examples shown the data payload is highlighted.

3.2 Encoded Formats

The encoded data formats provide for Modbus communications that are specific to the needs of the CALogix controller.
The Byte, Word and Double Word data formats are handled exactly as per the standard types listed above, however internally the interoperation of the data will be dependent on the actual CALogix register requested and these are detailed under the sections that provide the Registers individual actions.

Two types of data require further explanation. These are detailed below: -

3.2.1 Boolean Data

These data types are treaded as if they were Byte values with only 2 possible values, 0 or 1.
During a write operation (06) the Master should only set or clear the Least Significant Bit of the data payload. All other bits will be ignored (i.e. treated as *don't-care-bits*).
During read operations the CALogix controller will only ever return a value of zero or one.

Example 1: - *Request Output State, returning a value of 1 (=ON)*

01 03 08 5B 00 01 [CRC] => 01 03 02 ***00 01*** *[CRC]* (All values in hex.)

Example 2: - *Set Output Emergency Action to 1 (Output ON)*

01 06 08 67 ***00 01*** *[CRC] => 01 06 08 67 00 01 [CRC]* (All values in hex.)

In the examples shown the data payload is highlighted.

3.2.2 Float Data

Single Precision Floating point data is used extensively by the CALogix controller to report values and set important items such as Setpoints. These data types are packaged up as a 32 bit data

object using the standard IEEE single precision floating point format. This format is the standard used by most PC based operating systems and PC based compilers.
When transferring Float Data types a Double Word Data transfer is used to carry the encoded data to / from Master / Slave.

N.B. Special care must be take to ensure the bytes of the encoded Float data are sent Most Significant Byte first (big-endian). Master processors / PLC's based upon a little-endian data format may have to reverse the transmitted bytes to ensure that they are reconstructed by CALogix correctly.

Example 1: - *Request PV, returning a value of 50.0*

01 03 0A 1B 00 02 [CRC] => 01 03 04 ***42 48 00 00*** *[CRC]* (All values in hex.)

Example 2: - *Set the Heat Setpoint to 100.0*

01 10 07 CF 00 02 04 ***42 C8 00 00*** *[CRC] => 01 10 07 CF 00 02 [CRC]*

(All values in hex.)

In the examples shown the float data payload is highlighted.

For more information on IEEE Floating Point format see:-
ANSI/IEEE Standard 754-1985, Standard for Binary Floating Point Arithmetic.

3.2.3 Example C code Float conversion

Converts a floating-point data value for Modbus transfer.

In this example the 'cast' operator is used to convert a float to the 4 byte equivalent double word. Other methods could use a 'union' structure or treat the floating-point data as a byte array.

```
float          fValue = 10.0;
unsigned int   dwData;              // 4 byte DWORD holding register
unsigned int   *pdwData;            // pointer to DWORD

pdwData = (unsigned *) (&fValue);   // a pointer to the float type is take, but has
                                    // been cast to a DWORD pointer type

dwData = *pdwData;                  // the DWORD pointer is de-referenced to
                                    // yield the DWORD data.
```

The dwData (***41 20 00 00***) is incorporated in the modbus message.

3.3 Parameter Update Procedures

The CALogix system has 3 distinct methods of updating parameters. These are listed below: -

1. System configuration parameters are only updated at power-up.
2. Critical system function parameters require an *Update* command to accept parameter updates
3. Non-critical parameters are updated immediately.

The CALogix system will only update its module information at power-up This means if modules are removed, added or changed whilst the unit is powered the new parameters will NOT be updated.
In any case this is not an advisable course of action by the user due to electrical safety considerations.

Effected CALogix Registers are shown below: -

System Flags (Base Unit)
Function Type (module x 4)
Output Configuration (module x 4)
Serial Number (module x 4)

Always isolate the power supply before any changes to the electrical installation are made.

3.3.1 Update Command

Some critical CALogix system parameters will require an Update command before the new parameter is used by the system. This ensures a measure of safety and allows all critical parameters to be modified before action is taken.
During a critical update cycle, the CALogix unit will stop the internal processing task associated with the parameter being updated and restart that task with the new data. Unrelated tasks will not be affected.

The Update command takes the form of a write only register with a special data value 0x0055. See below: -

Register Name	Description	Data Size/Format	Default	Address
Update Command	Update critical CALogix parameter.	Word [W]	0x0055 (85)	**0x07CE** (1998)

The critical CALogix registers that require the Update command are shown below: -

TBD

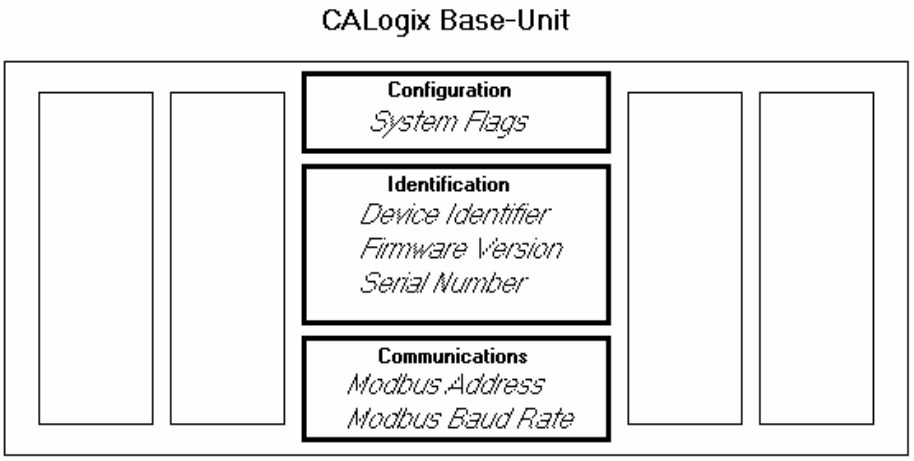
3.3.2 Validity

Check the validity of supplied parameters to the CALogix unit is a very important part of any program application. Under some circumstances the CALogix unit will reject parameters if they cannot be utilised. Under other conditions, some parameters may be truncated by the CALogix unit, and thus the system could enter an undesired mode of operation.
The programming application should always read back any important parameter that it has written to ensure that it has been accepted. Care should be taken with critical parameters that require the Update command since their new value will not be reported until a Update command has been issued.

4 CALogix Base-unit Details

The CALogix Base-unit provides the Configuration & Identification information and allows control of the Communication parameters of the CALogix system.

A simplified block diagram is shown below:-



4.1 Base-Unit Address Table

The Modbus Registers associated with the Base-Unit are shown below: -

Register Name	Description	Data Size/Format	Default	Address
Device Identifier	Device Identifier Tag. Value = 4000 (decimal) for CALogix	Word [R]	0x0FA0 (4000)	**0x07CB** (1995)
Firmware Version	CALogix System Firmware Version. Encoded, see text.	DWord [R]	None	**0x07B2** (1970)
Serial Number	Base-Unit Serial Number. (Factory use only)	DWord [R]	None	**0x07BE** (1982)
System Flags	Provide information on the modules present and the units. Encoded	Byte [RW]	0	**0x07B8** (1976)
Modbus Address	Modbus Slave Address	Byte [RW]	1	**0x07CC** (1996)
Modbus Baud Rate	Modbus communications baud rate. Encoded, see text.	Byte [RW]	0xFF (255)	**0x07CD** (1997)

4.2 Firmware Version

The Firmware Version register returns a 32 bit data value with the following encoding: -

Bits 31- 24	Bits 23 –16	Bits 15 – 8	Bits 7 – 0
OEM Identity (0=CAL Controls Ltd.)	Release Build (0 for development)	Feature Code (major code changes)	Software fix (minor code change)

4.3 System Flags

The System Flags, report the presence of the modules fitted to the base unit. Additionally this register can be used to set the PV units (e.g. Degrees Celsius / Degrees Fahrenheit). The unit flags do not effect the operation of Logic Modules.

The System Flags register has the following encoding: -

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Module 4 0 = absent 1 = present	Module 3 0 = absent 1 = present	Module 2 0 = absent 1 = present	Module 1 0 = absent 1 = present	Module 4 0 = Deg C 1 = Deg F	Module 3 0 = Deg C 1 = Deg F	Module 2 0 = Deg C 1 = Deg F	Module 1 0 = Deg C 1 = Deg F

4.4 Modbus Address

The Modbus Address register allows modification of the Slave address of the CALogix unit.
The range is from 1 to 247 (the last 8 Modbus Addresses are reserved). The CALogix will not allow a Modbus Address of *zero* to be programmed.
If the CALogix unit receives a valid Modbus Address it will utilise the new value immediately and the normal response message from the Slave will incorporate the new value. If the Modbus Address value produces an error for some reason (e.g. value=0) the Slave will respond using its original Modbus Address.

The Master application should be aware of these conditions and handle Modbus Address changes with care.

4.5 Modbus Baud Rate

The CALogix unit sets itself to Automatic Baud-Rate detection at power-up therefore it is not normally necessary to program baud rates once communications have been established.
After the Automatic Baud-Rate detection has occurred the Modbus Baud Rate register will report the communication speed that has been selected *(although the Master already knows what it is since it must already be communicating at that speed in order to retrieve the information!)*.
The Modbus Baud-Rate register also allows user modification of the communication speed of the unit, if required.
If the CALogix unit receives a valid Modbus Baud Rate value it will utilise the new value immediately and the normal response message from the Slave will be at the new communications speed. If the Modbus Baud Rate value produces an error for some reason the Slave will respond at its original baud rate.

The Master application should be aware of these conditions and handle Modbus Baud Rate changes with care.

The Modbus Baud Rate register has the following encoding: -

Value	Baud Rate
0x01 (1)	115200
0x03 (3)	57600
0x05 (5)	38400
0x0B (11)	19200
0x17 (23)	9600
0x2F (47)	4800
0x5F (95)	2400
0xBF (191)	1200
0xFF (255)	Auto Baud

5 CALogix Module Details

There are four Module positions available on a CALogix system and each Modbus register function has 4 possible addresses available, one for each Module.

Configuration data used to set the operational parameters of the Modules is contained within the CALogix Base-Unit, therefore care must be exercised when reading a Module register where the Module is not fitted. In many cases the Module configuration data will be returned, and the Master must first determine if a Module is present, and thus the data is valid, before displaying the items read.
Similarly some items written to Module registers for modules that are not fitted could be accepted and stored, however in cases where the Module registers control critical functions the data will not be updated to stop control issues if a module is subsequently fitted at a later date.
In all cases the Master application should always test the acceptance of any parameter written by using 'write-verify' policy.

Each type of CALogix module fitted to a Base-Unit have a common set of Modbus registers that are used to provide information for that specific module.

The following table provides details of these registers: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
Function Type	Module Function (PID, LOGIC) Identifier Tag.	Byte [R]	**0x0A60** (2656)	**0x0A61** (2657)	**0x0A62** (2658)	**0x0A63** (2659)
Output Configuration	Output board configuration. Encoded, see text.	Byte [R]	**0x0A64** (2660)	**0x0A65** (2661)	**0x0A66** (2662)	**0x0A67** (2663)
Serial Number	Module Serial Number.	DWord [R]	**0x0A58** (2648)	**0x0A5A** (2650)	**0x0A5C** (2652)	**0x0A5E** (2654)

5.1 Function Type

Return values: -

Value	Module Function
0	No module fitted
1	PID Module
2	LOGIC Module
3 – 255	Invalid Module

5.2 Output Configuration

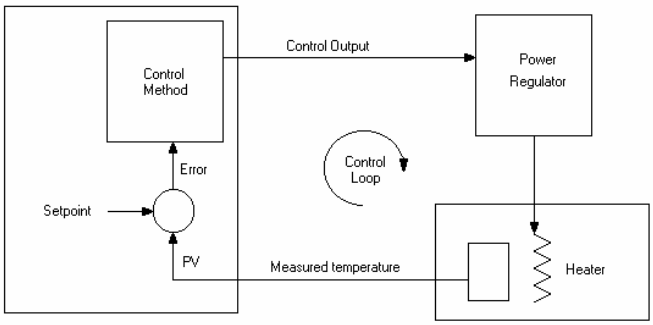
This register details the output configuration of the module. Encoded as follows: -

Value	Output 1 type	Output 2 type	Output 3 type
0x15 (21)	SSD	SSD	SSD
0x16 (22)	SSD	SSD	Relay
0x1A (26)	SSD	RLY	RLY
0x25 (37)	Relay	SSD	SSD
0x26 (38)	Relay	SSD	Relay
0x2A (42)	Relay	Relay	Relay
0x45 (69)	Analogue	SSD	SSD
0x46 (70)	Analogue	SSD	Relay
0x4A (74)	Analogue	Relay	Relay

(SSD = Solid State Drive)

6 CALogix PID module

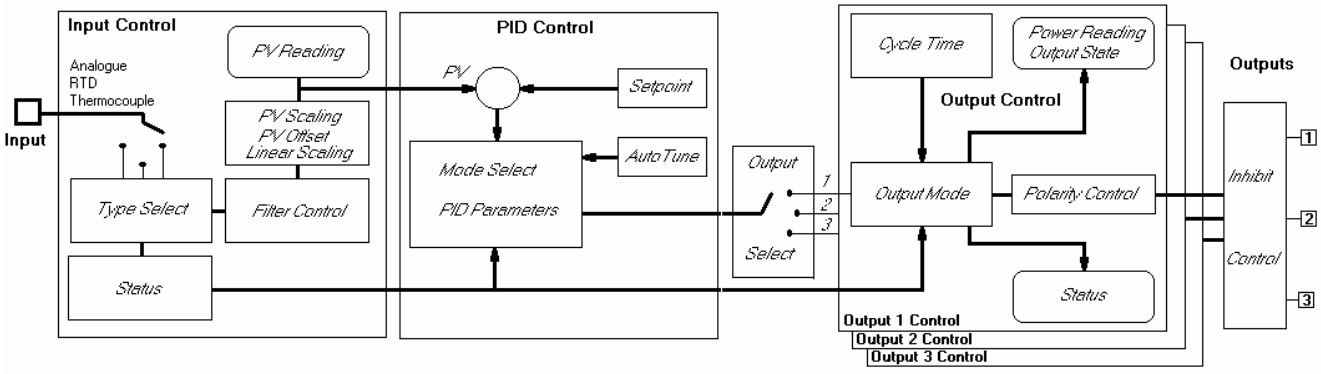
The CALogix PID modules provide all the interfaces required for PID Process control.
An example of a temperature control loop is shown below: -



The actual temperature measured at the process (PV) is connected to the input of the controller. This is compared with a setpoint (or required) temperature (SP). If there is an error between the set and measured temperature the controller calculates an output value to call for heating (or cooling). The calculation depends on the process being controlled but will normally utilise a PID algorithm to bring the error as close to zero as possible. The output(s) from the controller are connected to devices on the plant which cause the heating (or cooling) demand to be adjusted which in turn is detected by the temperature sensor. This is referred to as the close loop control.

6.1 Controller Block Diagram

A block diagram of the PID module controller is shown below: -



The PID module is divided into 3 functional block. These blocks are detailed below

6.2 Input Control

The following table gives details of the Input Control registers: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
Input Sensor	Sensor Type. (e.g. J –Type, RTD 2 wire). Encoded. see text.	Byte [RW]	**0x09D0** (2512)	**0x09D1** (2513)	**0x09D2** (2514)	**0x09D3** (2515)
Input Zero	Sensor Zero adjustment.	Float [RW]	**0x09D4** (2516)	**0x09D6** (2518)	**0x09D8** (2520)	**0x07B2** (2522)

Input Average	Input Filter Averaging value.	Byte [RW]	**0x0A40** (2624)	**0x0A41** (2625)	**0x0A42** (2626)	**0x0A43** (2627)
Input Band	Input Filter Band value.	Byte [RW]	**0x0A4C** (2636)	**0x0A4D** (2637)	**0x0A4E** (2638)	**0x0A4F** (2639)
Input Reading (PV)	Process Variable Reading.	Float [R]	**0x0A1C** (2588)	**0x0A1E** (2590)	**0x0A20** (2592)	**0x0A22** (2594)
Input Compensation	Compensation Reading. (where applicable)	Float [R]	**0x0A24** (2596)	**0x0A26** (2598)	**0x0A28** (2600)	**0x0A2A** (2602)
Input Status	Input Valid Flag.	Boolean [R]	**0x077B** (1915)	**0x077C** (1916)	**0x077D** (1917)	**0x077E** (1918)

6.2.1 Types and Ranges

The Sensor types are listed below: -

Register Value	Sensor Type	Description	Compensation	Sensor Range (°C)	Sensor Range (°F)
0	None	No Sensor allocated	N/A	N/A	N/A
1	B	Thermocouple type B	No	0 to 1800 °C	32 to 3272 °F
2	E	Thermocouple type E	Yes	0 to 600 °C	32 to 1112 °F
3	J	Thermocouple type J	Yes	0 to 800 °C	32 to 1472 °F
4	K	Thermocouple type K	Yes	-50 to 1200 °C	-58 to 2192 °F
5	L	Thermocouple type L	Yes	0 to 800 °C	32 to 1472 °F
6	N	Thermocouple type N	Yes	-50 to 1200 °C	-58 to 2192 °F
7	R	Thermocouple type R	Yes	0 to 1600 °C	32 to 2192 °F
8	S	Thermocouple type S	Yes	0 to 1600 °C	32 to 2192 °F
9	T	Thermocouple type T	Yes	-200 to 250 °C	-273 to 482 °F
10	RTD – 3 wire	PT100 temperature sensor, 3 – wire.	No	-200 to 800 °C	-273 to 1472 °F
11	RTD – 2 wire	PT100 temperature sensor, 2 – wire.	No	-200 to 800 °C	-273 to 1472 °F
12	Linear – Sensor	Linear Input	No	0 mA	50 mA

N.B.1 Type B accuracy not specified below 100 °C/ 212 °F.
N.B.2 Maximum Sensor lead resistance 100 Ω (including PT100 3 –wire).

6.2.2 Reading the PV

The PV reading is reported by the CALogix controller as a Floating point number. This value has a possible range of ± 1000, 000, 000 (± 1.0 E 9).
If the PV is outside the ranges of the instrument the PV inquiry will report the **Range-Limit** value of +1.0 E 12.
The limit value will be reported under the following conditions.

1. The instrument is in its initial sampling stages after power up (approx. 15 seconds).
2. The input level exceeds the limit of the input processor hardware.
3. The input value is lower than the minimum sensor range.
4. The temperature compensation value is outside the range of the instrument.

In the case where the input value is lower than the minimum sensor range a hysteresis of 10 °C above this value is applied to the PV before it will report the temperature reading again. This insures that the system will not cycle between reporting a negative value and a large positive value at the lower sensor ranges.

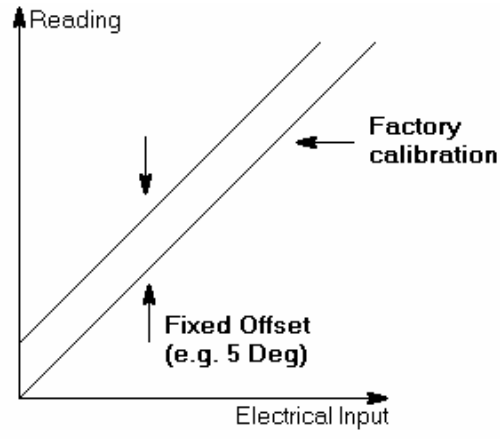
The Input Compensation Reading gives the current value of the compensation applied to the PV. The value shown when a PT100 3 - wire is selected represents the *TBD.*

During normal operation 10 readings are taken every seconds. Since the compensation reading changes are much slower these readings are only sampled every 5 seconds. During power-up however, 5 compensation readings are sampled continuously to allow the Input Control System of CALogix to reach a steady reading in the shortest possible time.

6.2.3 PV Offset

There may be occasions when the user wishes to apply an offset to the standard calibration to take account of known errors within the process, for example, a known sensor error or a known error due to the positioning of the sensor. In these instances it is possible to apply a user-defined offset.

PV Offset applies a single offset over the full sensor range of the controller. It has the effect of *moving* the calibration curve up or down, about a central point (the Y axis) as shown in the example below: -



The data parameter value supplied to the PV Offset register is a float type and is formatted as the same type of units as PV units (e.g. °C / °F).

6.2.4 Input Filter Control

The Input readings are filtered to remove electrical noise and disturbances from the input source. The filter must provide a smooth signal output reading but react readily to genuine signal step changes. To best accomplish this requirement a Banded-Average IIR digital filter algorithm is utilised.

Using this filter method the input signal is averaged for a user defined number of samples but only while the input sample value is within a user defined band. Outside the band the averaging filter will be re-seeded with the new sample values. This allows a smooth (averaged) signal to be obtained when the input signal is not changing greatly, but will also allow a quick response to sudden step changes (e.g. exothermic process)

Two registers control the filter operation: -

Input Average	This is the number of samples to average whilst the input signal is within band. A value of zero stops the averaging filter and will just return the sample, as input.
Input Band	This is the Band size. This is scaled according to the input sensor type selected. If a Thermocouple or Linear Sensor has been selected the Band size is in multiples of 10uV, as applied to the input. If the sensor

type is a PT100 then the Band size is specified in multiples of 10mΩ units.
The Band extends equally above and below the averaged sample value.

Warning: *The default Band & Average values have been optimised for industrial process control environments. These values should ONLY be changed for special applications.*

6.2.5 Input Status

The Input Status register reports the Status of the Input Hardware.

Two conditions are reported as follows: -

Input Emergency: Reported if the input to the CALogix module exceeds the limits of the hardware or the sensor has become open-circuit (Sensor-Break).

Input Ignored: Reported whilst the input system is acquiring enough new samples at power-up to provide a steady reading. This status flag wall also be set, during the pre-sample period after a *Sensor Type* change has been made.

The Input Status byte is ***bit*** encoded as follows: -

Bit 7 – 2	Bit 1	Bit 0
Not Allocated	*Input Ignored*	*Input Emergency*

N.B. It is possible to get both Input Emergency AND Input Ignored bits set together (e.g. during Sensor changes with an unconnected sensor).

6.3 Linear Input Parameters

When the *Sensor Type* register is set to Linear-Sensor (12) the CALogix unit will treat the input as scaleable with a single linear range with no compensation. Under these circumstances additional registers are utilised to provide configuration of the required linear scaling.

These are as follows: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
Linear Input Low	Linear sensor input minimum value. (scaled in mV)	Float [RW]	**0x0A78** (2680)	**0x0A7A** (2682)	**0x0A7C** (2684)	**0x0A7E** (2686)
Linear Input High	Linear sensor input maximum value. (scaled in mV)	Float [RW]	**0x0A88** (2696)	**0x0A8A** (2698)	**0x0A8C** (2700)	**0x0A8E** (2702)
Linear Scale Low	Scaled minimum reading.	Float [RW]	**0x0A80** (2688)	**0x0A82** (2690)	**0x0A84** (2692)	**0x0A86** (2694)
Linear Scale High	Scaled maximum reading.	Float [RW]	**0x0A90** (2704)	**0x0A92** (2706)	**0x0A94** (2708)	**0x0A96** (2710)

6.3.1 Input Range

The maximum and minimum values that can be utilised by the Linear-Sensor function is determined by the hardware of the CALogix module. The range of the instrument is 50mV. This is shown below: -

1. Minimum Input value: - **0mV**
2. Maximum Input value: - **50mV**

Since the values can only range from 0 to 50mV the parameter supplied to the *Linear Input Low* and *Linear Input High* are expected in mV.

Thus to program Linear Input Low register to 23.5 mV send the value **23.5** (N.B. Do NOT send 0.0235).

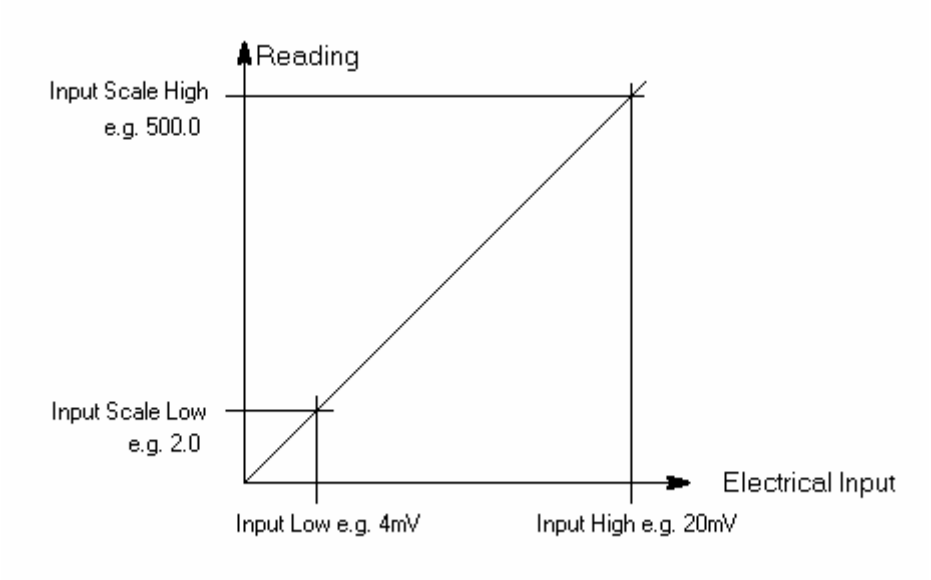
6.3.2 Linear Scaling

Using the Linear Scale Low and Linear Scale High registers the PV can be scaled as required. Each of the registers will accept a value in the range –10000 to +10000. It is possible to have to have negative scaling, where the Linear Scale Low value is greater than the Linear Scale High value.

The Linear Sensor control system will automatically allow 10% overrun above and below the scaled range of the sensor. In addition a hysteresis of 2% of range at the lowest end of the scaling will be applied to ensure the PV reading will not cycle at the lowest values. Outside these limits the PV will report the **Range-Limit** value of +1.0 E 12.

6.3.3 Linear Input Example

The graph below shows an example of Linear Sensor scaling, where it is required to display 2.0 when the input is 4mV and 500.0 when the input is 20mV.



The Linear Sensor will have 10% overrun above and below the maximum and minimum values. This means: -

Maximum reading: *500.0 + ((500.0 – 2.0) * 0.1) = 549.8*
Minimum reading: *2.0 – ((500.0 – 2.0) * 0.1) = -47.8*

6.4 PID Control Parameters

The CALogix PID module implements up to two PID controllers.
The first set-point (SP1) is normally associated with Heat control and a second set-point (SP2) controller can be used for Cooler applications. Under the Heat-Cool configuration the second set-point is not independent but is relative to the first (Heat) set-point.

The following table gives details of the PID Control registers: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
Setpoint 1 (Heat SP)	SP1 or Heat setpoint.	Float [RW]	**0x07D0** (2000)	**0x07D2** (2002)	**0x07D4** (2004)	**0x07D6** (2006)
Setpoint 2 (Cool SP)	SP2 or Cool setpoint.	Float [RW]	**0x07D8** (2008)	**0x07DA** (2010)	**0x07DC** (2012)	**0x07DE** (2014)
SP1 Offset	Control offset or Manual reset for SP1 / Heat.	Float [RW]	**0x07E0** (2016)	**0x07E2** (2018)	**0x07E4** (2020)	**0x07E6** (2022)
SP2 Offset	Control offset or Manual reset for SP2 / Cool.	Float [RW]	**0x07E8** (2024)	**0x07EA** (2026)	**0x07EC** (2028)	**0x07EE** (2030)
SP1 Band	Proportional control band for SP1 / Heat.	Float [RW]	**0x07F0** (2032)	**0x07F2** (2034)	**0x07F4** (2036)	**0x07F6** (2038)
SP2 Band	Proportional control band for SP2 / Cool.	Float [RW]	**0x07F8** (2040)	**0x07FA** (2042)	**0x07FC** (2044)	**0x07FE** (2046)
SP1 Mode	Control Mode for SP1 / Heat	Byte [RW]	**0x0800** (2048)	**0x0801** (2049)	**0x0802** (2050)	**0x0803** (2051)
SP2 Mode	Control Mode for SP2 / Cool	Byte [RW]	**0x0804** (2052)	**0x0805** (2053)	**0x0806** (2054)	**0x0807** (2055)
SP1 DAC	Derivative approach control for SP1 / Heat.	Byte [RW]	**0x0808** (2056)	**0x0809** (2057)	**0x080A** (2058)	**0x080B** (2059)
SP2 DAC	Derivative approach control for SP2 / Cool.	Byte [RW]	**0x080C** (2060)	**0x080D** (2061)	**0x080E** (2062)	**0x080F** (2063)
SP1 Integral Time	Integral time for SP1 / Heat.	Word [RW]	**0x0810** (2064)	**0x0811** (2065)	**0x0812** (2066)	**0x0813** (2067)
SP2 Integral Time	Integral time for SP2 / Cool.	Word [RW]	**0x0814** (2068)	**0x0815** (2069)	**0x0816** (2070)	**0x0817** (2071)
SP1 Derivative Time	Derivative time for SP1 / Heat.	Byte [RW]	**0x0818** (2072)	**0x0819** (2073)	**0x081A** (2074)	**0x081B** (2075)
SP2 Derivative Time	Derivative time for SP2 / Cool.	Byte [RW]	**0x081C** (2076)	**0x081D** (2077)	**0x081E** (2078)	**0x081F** (2079)
SP1 Der Sens	Derivative sensitivity for SP1 / Heat.	Byte [RW]	**0x0820** (2080)	**0x0821** (2081)	**0x0822** (2082)	**0x0823** (2083)
SP2 Der Sens	Derivative sensitivity SP2 / Cool.	Byte [RW]	**0x0824** (2084)	**0x0825** (2085)	**0x0826** (2086)	**0x0827** (2087)
SP1 Output Address	SP1 Control Block Output Address select	Byte [RW]	**0x0830** (2096)	**0x0831** (2097**)**	**0x0832** (2098)	**0x0833** (2099)
SP2 Output Address	SP2 Control Block Output Address select	Byte [RW]	**0x0834** (3000)	**0x0835** (3001**)**	**0x0836** (3002)	**0x0837** (3003)

6.4.1 Control Modes

The control modes register provides the various types of control algorithm that can be used by the controller. In addition this register is used to start the Auto-tune process.

The following table gives the settings: -

Register Value	Mode	Description
0	PARK	In Park mode the controller becomes inactive. The control output will normally not call for heat OR cool.
1	ON-OFF	On-Off mode provides the simplest control system available. In this mode the 'Band' register provides the Hysteresis.
2	P	Proportional only control.

3	PD	Proportional with Derivative action control.
4	PI	Proportional with Integral action control.
5	PID	Full PID control algorithm.
6	TUNE	Start Tune process. Tune is at 75% of Setpoint.
7	ATSP	Start Tune process at Setpoint.
8	TUNE-OFF	Abort running Tune process.

N.B. TUNE modes are covered in more detail in section 6.5

6.4.2 Setpoint Adjustment

The control setpoint con be adjusted to any desired value within the range of the currently selected Sensor. Control operation outside these limits are not advised due to non-guaranteed accuracy of the input PV readings.

6.4.3 PID Control Parameters

For P, PD, PI and PID control the following registers are provided for SP1 and SP2. These registers give the controlling parameters for PID control.

Band This setting is used in both proportional and ON-OFF control modes to select the size of the proportional band in proportional mode, or the hysteresis band size in the case of ON-OFF. The parameter will have a float value in PV units.

DAC The Derivative Approach Control (*DAC*) parameter is used to define the band over which the derivative action will work, centred on the setpoint. The setting is a multiplying factor of the proportional band, and allows the warm-up characteristics of the system to be controlled independently of the normal control loop. The range is from 0 to 15.
The encoding format is shown below: -

Derivative Band Size = (DAC value / 2) + 0.5

Thus: -
Minimum Derivative band size = 0.5 x *Band*
Maximum Derivative band size = 8.0 x *Band*

Offset / Manual Reset This parameter is used to offset the hysteresis or proportional band away from the setpoint to counteract mismatches between control inputs of a system and the system losses.

Integral Time This parameter sets up the integral time in *minutes* and is only used in PI and PID control modes.
The range of the parameter is 1 to 1000 in 0.1 minutes steps. The value zero is invalid.

Derivative Time This parameter sets up the derivative time in *seconds* and is only used in PD and PID control modes.
The range of the parameter is 1 to 200 in 1 second steps. The value zero is invalid.

Derivative

Sensitivity The Derivative Sensitivity (*DerSens*) is used to control how often the derivative control mode samples the process value to calculate the derivative contribution to the control loop.
It is expressed as a multiplying factor to be applied to the setting of the D*erivative Time* in order to give a shorter derivative sample time than the D*erivative Time* itself. The range is from 0 to 15.
The encoding format is shown below: -

*Derivative Sample Multiplier = (DerSens value * 0.1) + 0.1*

Thus: -
Minimum Derivative Sample Time = 0.1 x *Derivative Time*
Maximum Derivative Sample Time = 1.6 x *Derivative Time*

6.4.4 ON – OFF control and Hysteresis

The simplest control method available in the CALogix system is ON – OFF control. This is implemented by the controller turning ON the output when the process value is below the Setpoint, and turning OFF when the process value rises above the setpoint. In order to avoid the situation where the process value hovers around the Setpoint (as is required), with the output device switching rapidly as the PV rises and falls marginally, hysterisis is added to the OFF state. Thus the output stays OFF until the process value has fallen below the Setpoint by the size of the hysteresis band.
The hysteresis band value is set up in the *Band* register. This can have a range from 0 to the maximum range limit of the current sensor selector. The value is a float expressed in the same units as the PV.

6.4.5 Integral Action and Manual Reset

Using Proportional control only, errors between the required temperature setting and the actual temperature may build up in the overall control system due to mismatches between the control input and the process being controlled. Such a mismatch may occur if for instance a heater is slightly too powerful or weak for a given process at the programmed cycle time. Unless the heater and cycle time are an exact match for the heat loss of the system when the output is at 50% duty cycle, there will be a tendency for the system to sit at some other temperature than the Setpoint.

This situation can be treated manually using the Offset / Manual Reset register. In the case where Offset / Manual Reset has been given a non-zero value the effect is to move the Proportional band in relation to the Setpoint, so that when the process value is at Setpoint, it will no longer represent a proportioning value of 50%.
A negative value of Offset / Manual Reset will move the proportional band down, making the control output reduce its proportion by comparison with the case of no offset for the same process value. A positive value moves the band up in relation to the Setpoint, resulting in the output increase its proportion for a given process value.

When PI or PID control modes are selected Integral action is used to automatically adjust the Offset / Manual Reset value. This has the effect of automatically tracking the error between Setpoint and PV, moving the proportional band about the Setpoint to counteract any tendency for the temperature to drift away from the Setpoint.

Under this regime the Offset / Manual Reset value does not contribute to the control algorithm. In stead the Offset / Manual Reset value will report the offset to the Proportional band, which is being applied due to the Integral action.

Integral action operates within a band around the Setpoint, which is dictated by the size of the Proportional band. This allows the Integral Offset applied to proportional band to swing from 0 to 100% above or below the Setpoint (i.e. the Integral band size is 2 x Proportional band size). Outside these band limits the Integral control will apply the maximum rate of movement to drive the band towards the PV as quickly as possible.

When ever the PI or PID control mode is set within the *Control Mode* register the Integral Offset value will be initialised to 25 % of the Proportional Band value.

6.4.6 Heat-Cool Mode

Heat-Cool operation is a special mode of operation involving both SP1 and SP2 outputs. SP1 is assumed to control a heater, and SP2 controls a cooler. The SP2 proportional or hysteresis band is assumed to be above the SP1 Setpoint, but may lie completely within the SP1 proportional or hysteresis band. The central feature of Heat-Cool operation is that the bands of SP1 and SP2 are linked together, a movement of SP1 band moving the SP2 band by the same amount.
Thus Manual Reset or Integral action operating on the SP1 band will effect the position of the SP2 band. A further feature is that during Heat-Cool mode, the integral band movement limits (size) are modified to allow that the SP1 and SP2 bands to exert fully ON and fully OFF action at the Setpoint.

N.B. To select Heat-Cool operation the SP2 controller must be selected into Cool mode. This is achieved by setting the *SP2 Alarm Mode* settings to Cool. See section 6.7.

When Integral operation is selected for SP2 or Cool (i.e. *SP2 Mode* set to PI or PID) a modified Integral action takes place. Since the Integral operation of SP1 is linked to the band of SP2 the system will switch its Integral Time to the new SP2 Integral Time when the PV has risen above the upper limit of the SP1 band. Using this method of Integral Rate Switching enables the system to adapt to the different losses involved in a heating or a cooling system.

See Application Note for more info on Heat-Cool operation.

6.4.7 SP1/SP2 Output Address select

The CALogix system allows any Control block output to be directed to any available physical output on any module fitted. Although this flexibility exists it is normal to select ONLY outputs on the module that is providing the Control functionality.

The output allocation is encoded as follows: -

Module	Output 1	Output 2	Output 3	No Output Allocated
Module 1	0	4	8	12
Module 2	1	5	9	13
Module 3	2	6	10	14
Module 4	3	7	11	15

Example: - To connect the output of SP1 control block on Module 3 to Output 2 on Module 3 the *SP1 Output Address* register must be programmed with the value –

SP1 Output Address = 6

6.5 AutoTune

The CALogix incorporates an automatic tuning process, which can tune the settings of the PID parameters to the load being controlled. There are two modes of operation, selected using either TUNE to tune the system below the SP1 setpoint or ATSP to tune the operation at the setpoint temperature. The selection of which of the two tune modes to use will depend on the situations the controller is used in. Since the tune process relies on generating overshoots of process values above the tuning setpoint, it may not be advisable to tune certain applications at the operating setpoint, in case of damage to the load or its contents. The Tune-at-setpoint (ATSP) operation is intended for tuning an application if it is already up to operating temperature and the rate of heat loss is low, which would mean long a wait in the case of dropping to a 75% setpoint. It may also be used for tuning low temperatures close to the ambient, where the temperature drop does not need to be so large.

The AutoTune operation is only applicable to loads whose process value increase when the SP1 output is logically ON and decrease when it is logically OFF. AutoTune assumes that there is no power limiting applied to the SP1 output when performing its calculations.

The AutoTune cycle is started by setting the *Control Mode* register to the appropriate value, either TUNE or ATSP. Once started, AutoTune will run to completion or until a failure condition halts the process. At this point the controller will return to the control mode of operation in use before the AutoTune cycle was invoked. If the *Control Mode* register is programmed to the TUNE-OFF value during this process the AutoTune cycle will abort and return the controller to the previous control mode.

6.5.1 Tune Data

The tuning algorithm uses a modified one-shot, Ziegler-Nicholls tuning cycle to obtain information about the dynamic performance of the load at the setpoint, including the impulse response in terms of forced heating and the rate of heat loss due to cooling effect at the ambient temperature.

All the data used by the tuning algorithm is made available for the user to inspect.

The following table gives details of the Tune Data registers: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
AutoTune CtA	Approach time A	DWord [R]	**0x07D0** (2000)	**0x07D2** (2002)	**0x07D4** (2004)	**0x07D6** (2006)
AutoTune CtB	Approach time B	DWord [R]	**0x07D8** (2008)	**0x07DA** (2010)	**0x07DC** (2012)	**0x07DE** (2014)

AutoTune *Ct1*	Quarter time 1.	DWord [R]	**0x07E0** (2016)	**0x07E2** (2018)	**0x07E4** (2020)	**0x07E6** (2022)
AutoTune *Ct2*	Quarter time 2.	DWord [R]	**0x07E8** (2024)	**0x07EA** (2026)	**0x07EC** (2028)	**0x07EE** (2030)
AutoTune *Ct3*	Quarter time 3.	DWord [R]	**0x07F0** (2032)	**0x07F2** (2034)	**0x07F4** (2036)	**0x07F6** (2038)
AutoTune *Ct4*	Quarter time 4.	DWord [R]	**0x07F8** (2040)	**0x07FA** (2042)	**0x07FC** (2044)	**0x07FE** (2046)
AutoTune *Os1*	Overshoot 1	Float [R]	**0x0800** (2048)	**0x0801** (2049)	**0x0802** (2050)	**0x0803** (2051)
AutoTune *Us1*	Undershoot 1	Float [R]	**0x0804** (2052)	**0x0805** (2053)	**0x0806** (2054)	**0x0807** (2055)
AutoTune *Os2*	Overshoot 2	Float [R]	**0x0808** (2056)	**0x0809** (2057)	**0x080A** (2058)	**0x080B** (2059)
AutoTune Cycle Time	AutoTune Cycle Time. See section ????	Byte [R]	**0x080C** (2060)	**0x080D** (2061)	**0x080E** (2062)	**0x080F** (2063)
AutoTune Band	AutoTune Proportional Band.	Float [R]	**0x0810** (2064)	**0x0811** (2065)	**0x0812** (2066)	**0x0813** (2067)
AutoTune Integral Time	AutoTune Integral time. See section **6.4.3** for encoding format.	Word [R]	**0x0814** (2068)	**0x0815** (2069)	**0x0816** (2070)	**0x0817** (2071)
AutoTune Derivative Time	AutoTune Derivative time. See section **6.4.3** for encoding format.	Byte [R]	**0x0818** (2072)	**0x0819** (2073)	**0x081A** (2074)	**0x081B** (2075)
AutoTune Der Sens	AutoTune Derivative sensitivity. See section **6.4.3** for encoding format.	Byte [R]	**0x0820** (2080)	**0x0821** (2081)	**0x0822** (2082)	**0x0823** (2083)
AutoTune DAC	AutoTune Derivative Approach Control. See section **6.4.3** for encoding format.	Byte [R]	**0x0824** (2084)	**0x0825** (2085)	**0x0826** (2086)	**0x0827** (2087)
AutoTune Flags	AutoTune Flags: - Update cycle time from AutoTune. Tune Fail. See text.	Byte [RW]	**0x0830** (2096)	**0x0831** (2097**)**	**0x0832** (2098)	**0x0833** (2099)

6.5.2 Flags

The AutoTune Flag register provided both feedback and configuration of the AutoTune process. Using the AutoTune Flag register, the user can decide whether to automatically assign the AutoTune calculated Proportional cycle time to the PID controller. In some circumstances this may not be desirable due design limitations of the heaters or the power supply of the system.

The other feature of the AutoTune Flag register is to report on the status of the AutoTune process. If the AutoTune process fails for some reason (i.e. Quarter times are too small) then the AutoTune Failed bit will be set. This bit will be reset when a new AutoTune cycle is set running or alternatively whenever the AutoTune Flag register is written to.

N.B. *AutoTune cycle is completed when the Control Mode register reports that the controller has reverted back to normal operation.*

The encoding is show below: -

Bit 7 – 2	Bit 1	Bit 0
Not Used	0 = AutoTune Successful 1 = AutoTune Failed. *(Read Only)*	0 = Don't use tuned cycle time 1 = Use tuned cycle time

6.6 Output Block Parameters

The output block provides control of the physical output that has been assigned to the PID controller. Since there is 3 outputs to each Output Block there will correspondingly be 3 sets of registers for each parameter type. In the table below descriptions for the Output 2 and 3 have been left blank for clarity.

The following table gives details of the Output Block registers: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
Output 1 Cycle Time	Proportion Cycle Time	Byte [RW]	**0x0838** (2104)	**0x0839** (2105)	**0x083A** (2106)	**0x083B** (2107)
Output 2 Cycle Time			**0x083C** (2108)	**0x083D** (2109)	**0x083E** (2110)	**0x083F** (2111)
Output 3 Cycle Time			**0x0840** (2112)	**0x0841** (2113)	**0x0842** (2114)	**0x0843** (2115)
Output 1 Emergency Action	Output action on input failure etc.	Bool [RW]	**0x0844** (2116)	**0x0845** (2117)	**0x0846** (2118)	**0x0847** (2119)
Output 2 Emergency Action			**0x0848** (2120)	**0x0849** (2121)	**0x084A** (2122)	**0x084B** (2123)
Output 3 Emergency Action			**0x084C** (2124)	**0x084D** (2125)	**0x084E** (2126)	**0x084F** (2127)
Output 1 Status	Output State (ON or OFF).	Bool [RW]	**0x085C** (2140)	**0x085D** (2141)	**0x085E** (2142)	**0x085F** (2143)
Output 2 Status			**0x0860** (2144)	**0x0861** (2145)	**0x0862** (2146)	**0x0863** (2147)
Output 3 Status			**0x0864** (2148)	**0x0865** (2149)	**0x0866** (2150)	**0x0867** (2151)
Output 1 Manual Power	Manual Output Power.	Byte [RW]	**0x0868** (2152)	**0x0869** (2153)	**0x086A** (2154)	**0x086B** (2155)
Output 2 Manual Power			**0x086C** (2156)	**0x086D** (2157)	**0x086E** (2158)	**0x086F** (2159)
Output 3 Manual Power			**0x0870** (2160)	**0x0871** (2161)	**0x0872** (2162)	**0x0873** (2163)
Output 1 Minimum Power	Minimum Output Power.	Byte [RW]	**0x0874** (2164)	**0x0875** (2165)	**0x0876** (2166)	**0x0877** (2167)
Output 2 Minimum Power			**0x0878** (2168)	**0x0879** (2169**)**	**0x087A** (2170)	**0x087B** (2171)
Output 3 Minimum Power			**0x087C** (2172)	**0x087D** (2173**)**	**0x087E** (2174)	**0x087F** (2175)
Output 1 Maximum Power	Maximum Output Power.	Byte [RW]	**0x0880** (2176)	**0x0881** (2177)	**0x0882** (2178)	**0x0883** (2179)
Output 2 Maximum Power			**0x0884** (2180)	**0x0885** (2181**)**	**0x0886** (2182)	**0x0887** (2183)
Output 3 Maximum Power			**0x0888** (2184)	**0x0889** (2185**)**	**0x088A** (2186)	**0x088B** (2187)
Output 1 Inhibit	Inhibit output address value (prevents output contention).	Byte [RW]	**0x088C** (2188)	**0x088D** (2189)	**0x088E** (2190)	**0x088F** (2191)
Output 2 Inhibit			**0x0890** (2192)	**0x0891** (2193**)**	**0x0892** (2194)	**0x0893** (2195)
Output 3 Inhibit			**0x0894** (2196)	**0x0895** (2197**)**	**0x0896** (2198)	**0x0897** (2199)
Output 1 Emergency Flag	Reports if the output is in normal operation or not.	Bool [R]	**0x0898** (2200)	**0x0899** (2201)	**0x089A** (2202)	**0x089B** (2203)
Output 2 Emergency Flag			**0x089C** (2204)	**0x089D** (2205**)**	**0x089E** (2206)	**0x089F** (2207)
Output 3 Emergency Flag			**0x08A0** (2208)	**0x08A1** (2209**)**	**0x08A2** (2210)	**0x08A3** (2211)
Output 1 Action	Output invert operation action.	Bool [R]	**0x0850** (2128)	**0x0851** (2129)	**0x0852** (2130)	**0x0853** (2131)
Output 2 Action			**0x0854** (2132)	**0x0855** (2133)	**0x0856** (2134)	**0x0857** (2135)
Output 3 Action			**0x0858** (2136)	**0x0859** (2137)	**0x085A** (2138)	**0x085B** (2139)
Output 1 Power	Reports the Output Power.	Byte [R]	**0x08A4** (2212)	**0x08A5** (2213)	**0x08A6** (2214)	**0x08A7** (2215)
Output 2 Power			**0x08A8** (2216)	**0x08A9** (2217**)**	**0x08AA** (2218)	**0x08AB** (2219)
Output 3 Power			**0x08AC** (2220)	**0x08AD** (3221**)**	**0x08AE** (2222)	**0x08AF** (2223)
Output 1 Diagnostics	Output operations count; stored in NVR (non reset-able).	DWord [R]	**0x08B0** (2224)	**0x08B2** (2226)	**0x08B4** (2228)	**0x08B6** (2230)
Output 2 Diagnostics			**0x08B8** (2232)	**0x08BA** (2234**)**	**0x08BC** (2236)	**0x08BE** (2238)
Output 3 Diagnostics			**0x08C0** (2240)	**0x08C2** (2242**)**	**0x08C4** (2244)	**0x08C6** (2246)

6.6.1 Output Cycle Time

This register programs the duty cycle of the controller and is used by the PID controller to provide the proportion of heat/cool required by the load.

The register has encoding as follows

Register Values	Cycle Time (seconds)
0	Invalid *(the controller will not accept a value of 0)*
1 – 99	0.1 to 9.9 sec / step size 0.1 second.
100 – 171	10 – 81 sec / step size 1 second.

6.6.2 Proportional ON and OFF times

In order to save the output devices from switching for very short periods and eroding relay contacts, the Proportioning algorithm ensures that the minimum period for OFF or ON duty cycle is 2.5 %. This is done by rounding the duty cycle to either 0 % or 2.5 % and accumulating the residue, which is then applied to the next cycles proportion. This operation is only applied to outputs with cycle times set to 5 seconds or more, since it assumed that a cycle time less than this will only be used on Solid State device (SSd) outputs, which will not suffer from short switching times.

6.6.3 Output Action and Output Status

Two registers *Output Action* and *Output Status* provide configuration and status on the condition of the output under normal conditions. Under normal operating conditions the output is ON when the control output is 'Active'.

These are both Boolean type registers in that they have only two possible settings: -

The encoding is shown below: -

Output Action:	0 = Output is NOT inverted, (i.e. Output ON when active). 1 = Output IS inverted, (i.e. Output OFF when active).
Output Status:	0 = The output is OFF (relay open, SSd denergised) 1 = The output is ON (relay closed, SSd energised)

6.6.4 Emergency Action and Emergency Flag

Two registers *Emergency Action* and *Emergency Flag* provide configuration and status on the condition of the output under Emergency conditions. The Emergency condition is when the input to the PID controller connected to the output has gone into failure.

These are both Boolean type registers in that they have only two possible settings: -

The encoding is shown below: -

Emergency Action:	0 = Set output to OFF when in Emergency condition. 1 = Set output to ON when in Emergency condition.
Emergency Flag:	0 = Normal output operation

1 = Output is in the Emergency condition as programmed by the *Emergency Action* register.

6.6.5 Output Power

Four registers provide configuration and status of the output power settings of the controller. The output power is a measure of duty cycle that is applied to the output at a given cycle time.
As an example, with a *Output Cycle Time* setting of 10 seconds and a power output of 25 %, the output would go ON for 2.5 seconds and OFF for 7.5 seconds.

Manual Power: If this register is programmed with a value greater than **zero** it will over-ride the controller function and constantly output the value of power specified. The value of Manual power will still be subjected to the Maximum and Minimum limits.

Maximum Power: This provides a Maximum power setting that the controller will output. The value must be greater that the Minimum Power setting.

Minimum Power: This provides a Minimum power setting that the controller will output.
The value must be less than the Maximum Power setting.

Output Power: This reports (read only) the actual output power being applied to the output.

These registers are all encoded as a percentage with range 0 – 100%. Value greater than 100 % will be truncated to 100 %

6.6.6 Output Inhibit

These registers provide the means to inhibit the activation of any output if the current output is already ON. This is sometimes useful where two outputs are being used (i.e. Heat/Cool application) but the plant cannot tolerate both outputs being ON at the same time. In the case of setting the Inhibit action the register *Name* implies the *Master* control output and the register *Value* is the *Slave* address of the inhibited output.

Example: when Output 1 is ON inhibit Output 3, module 1 from operating.

Output 1 Inhibit register **(0x088C)** = 8 (Output 3, Module 1)

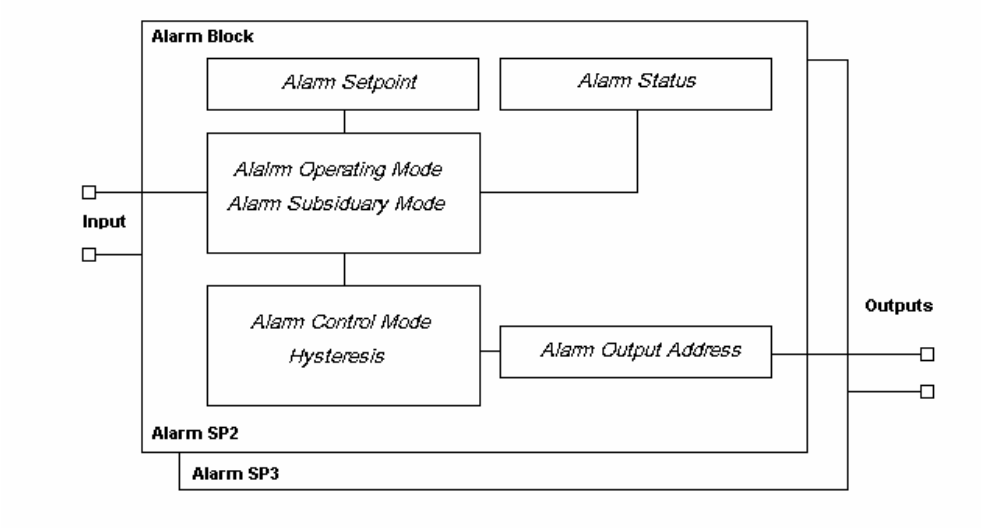
N.B. Any output can inhibit any other output within the CALogix system, but normally only outputs within the same module should be inhibited.
Care should be taken when programming the *Output Inhibit* register since an output can also be programmed to inhibit itself, causing the output not to function.

The output inhibit address values are is encoded as follows: -

Module	Output 1	Output 2	Output 3	No Output Allocated
Module 1	0	4	8	12
Module 2	1	5	9	13
Module 3	2	6	10	14
Module 4	3	7	11	15

6.7 Alarms

A block diagram of the Alarm processor is shown below: -



Alarms are only available for second and third Setpoints (SP2 and SP3). The input to the Alarm Block is fed from the Input Block as detailed in section 6.2 and the outputs are fed to the Output Control Block as detailed in section 6.6.

6.7.1 Alarm Control Parameters

The following table gives details of the Alarm Control registers: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
SP2 Alarm Operating Mode	Alarm Operational Mode	Byte [RW]	**0x0A9C** (2716)	**0x0A9D** (2717)	**0x0A9E** (2718)	**0x0A9F** (2719)
SP3 Alarm Operating Mode		Byte [RW]	**0x0AA8** (2728)	**0x0AA9** (2729)	**0x0AAA** (2730)	**0x0AAB** (2731)
SP2 Alarm Subsidiary Mode	Alarm Subsidiary Mode.	Byte [RW]	**0x0AA0** (2720)	**0x0AA1** (2721)	**0x0AA2** (2722)	**0x0AA3** (2723)
SP3 Alarm Subsidiary Mode		Byte [RW]	**0x0AAC** (2732)	**0x0AAD** (2733)	**0x0AAE** (2734)	**0x0AAF** (2735)
SP2 Alarm Control Mode	Alarm Control Mode.	Byte [RW]	**0x0804** (2052)	**0x0805** (2053)	**0x0807** (2054)	**0x0808** (2055)
SP3 Alarm Control Mode		Byte [RW]	**0x0ACC** (2764)	**0x0ACD** (2765)	**0x0ACE** (2766)	**0x0ACF** (2767)
SP2 Setpoint	Alarm Setpoint	Float [RW]	**0x07D8** (2008)	**0x07DA** (2010)	**0x07DC** (2012)	**0x07DE** (2014)
SP3 Setpoint		Float [RW]	**0x0AB4** (2740)	**0x0AB6** (2742)	**0x0AB8** (2744)	**0x0ABA** (2746)
SP2 Hysteresis	Hysteresis or Proportional Band value.	Float [RW]	**0x07F8** (2040)	**0x07FA** (2042)	**0x07FC** (2044)	**0x07FE** (2046)
SP3 Hysteresis		Float [RW]	**0x0ABC** (2748)	**0x0ABE** (2750)	**0x0AC0** (2752)	**0x0AC2** (2754)
SP2 Alarm Output Address	Alarm output address	Byte [RW]	**0x0834** (2100)	**0x0835** (2101)	**0x0836** (2102)	**0x0837** (2103)
SP3 Alalrm Output Address		Byte [RW]	**0x0AC4** (2756)	**0x0AC5** (2757)	**0x0AC6** (2758)	**0x0AC7** (2759)
SP2 Alarm Status	Alarm Status and Reset.	Byte [RW]	**0x0AA4** (2724)	**0x0AA5** (2725)	**0x0AA6** (2726)	**0x0AA7** (2727)
SP3 Alarm Status		Byte [RW]	**0x0AB0** (2736)	**0x0AB1** (2737)	**0x0AB2** (2738)	**0x0AB3** (2739)

6.7.2 Types of Alarm

The types of alarm are programmed using the *Alarm Operation* register. The output of the alarm block is either *ACTIVE* or *NOT-ACTIVE.* The output of the Alarms are *ON* (closed) when *NOT-ACTIVE* and *OFF* (open) when *ACTIVE*.

The following table gives the settings: -

Register Value	Mode	Description
0	NONE	An Alarm is not selected for this Setpoint. The Alarm will remain not active and no output action will occur.
1	DVHI	The output go ACTIVE when the input moves *ABOVE* the main Setpoint SP1 by the deviation specified in the *Alarm Setpoint* register (SP2 or SP3).
2	DVLO	The output go ACTIVE when the input moves *BELOW* the main Setpoint SP1 by the deviation specified in the *Alarm Setpoint* register (SP2 or SP3).
3	BAND	The output go ACTIVE when the input moves ABOVE or *BELOW* the main Setpoint SP1 by the deviation specified in the *Alarm Setpoint* register (SP2 or SP3). This deviation sits equally either side of the main Setpoint SP1.
4	FSHI	The output go ACTIVE when the input moves *ABOVE* the by the full-scale value specified in the *Alarm Setpoint* register (SP2 or SP3).
5	FSLO	The output go ACTIVE when the input moves *BELOW* the by the full-scale value specified in the *Alarm Setpoint* register (SP2 or SP3).
6	COOL	Heat-Cool Mode is selected. See section 6.4.6. This is only applicable to SP2 setting the *SP3 Alarm Operation* register to this setting will have no effect.
7	EOP	Event Output from Programmer. See section 7.

6.7.3 Subsidiary Modes

The Subsidiary Modes provide additional features to the standard alarm operation. Any of these subsidiary modes can be applied to all but COOL and EOP alarm operational modes.

The following table gives the settings: -

Register Value	Mode	Description
0	NONE	No subsidiary mode is applied to the Alarm operation selected.
1	LTCH	*Alarm Latch.* Once the Alarm output goes *ACTIVE* it is latched in the ACTIVE state even if the condition that caused the alarm is removed.
2	HOLD	*Alarm Hold.* The alarm will remain *NOT-ACTIVE* even if the input is within the alarm condition, until the input is returned to the non-alarm condition. In this state the alarm is in the *'hold-off'* condition. After the *hold-off* has been cleared the alarm operation will respond as normal.
3	LTHO	Both *Alarm Hold and Latch*. The *Hold-off* must be triggered first before the *Alarm Latch* mode becomes active.

6.7.4 Control Modes

The Control mode provides configuration of how the output will behave when the Alarm has been triggered and is in the *ACTIVE* state.

The following table gives the settings: -

Register Value	Mode	Description
0	PARK	In Park mode the controller becomes inactive. The control output will normally not call for heat OR cool.
1	ON-OFF	ON -OFF mode provides the simplest control system available. In this mode the *Hysteresis* register is used to control the switching characteristics of *ACTIVE* alarms.
2	PROP	Proportional control. In this mode the *Hysteresis* register is used to provide a *Band* around the alarm activation point where the output will control.
3	NLIN	This is a special mode used to modify the Proportional control in Cool mode only.

6.7.5 Setpoint & Hysteresis

The Alarm Setpoint can be of two types: -

Deviation Setpoint: The Alarm Setpoint is relative to the main Setpoint SP1. To calculate the Alarm activation point the operation mode must be known.

For DVHI: Alarm activation point = SP1 + SP2
For DVLO: Alarm activation point = SP1 - SP2
For BAND: Alarm activation point = SP1 - SP2 or SP1 + SP2

Full-Scale Setpoint: The Alarm Setpoint is the actual activation point.

The Setpoint values are always written in the same units as the PV.

The Hysteresis must be a positive non-zero value, expressed in the same units as the PV. This value represents the switch point Hysteresis when used in ON – OFF control mode and a Proportion Control Band when used in PROP mode.

When used in ON –OFF mode the Hysteresis area will placed above or below the Alarm activation point, depending on Operational mode. When used in PROP control mode the Band is placed with equal spread either side of the Alarm activation point, with the 50 % control point at the Alarm activation point.

6.7.6 Output Configuration

The Alarm Output Address register allows the user to direct the Alarm output to any of the physical outputs on the CALogix system. Although this flexibility exists it is normal to select ONLY outputs on the module that is providing the Alarm Control functionality.

The output allocation is encoded as follows: -

Module	Output 1	Output 2	Output 3	No Output Allocated
Module 1	0	4	8	12
Module 2	1	5	9	13
Module 3	2	6	10	14
Module 4	3	7	11	15

N.B. Allocation of Alarm outputs is subject to the physical output *NOT* being connected to a Control output. If for example an attempt is made to connect SP2 Alarm output to Output 1, Module 1, but this Output is already ***owned*** by SP1 PID Control Block then the request will not be serviced and the register will remain unmodified. Care should always be taken when setting ***ANY*** address allocation to ensure the parameter has been accepted.

6.7.7 Alarm Status and Reset

An Alarm Status register provides feedback on the various phases of activation of the Alarm. Additionally this register provides a method of resetting Latched Alarms.

This is a bit encoded register as follows: -

Bit 7 – 4	Bit 3	Bit 2	Bit 1	Bit 0
Spare	*Alarm Active* 0 – Alarm Off 1 – Alarm On	*Alarm Enable* 0 – Not Enabled 1 – Enabled	*Alarm Hold* 0 – Not Held Off 1 – Held Off	*Alarm Latch* 0 – Not Latched 1 – Latched

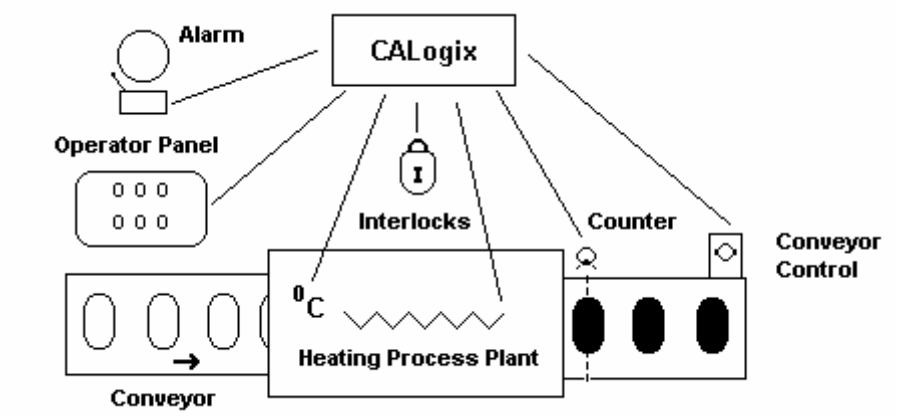
Alarm Latch: Reports if the Alarm has been latched in the ACTIVE state.
Alarm Hold: Reports if the Alarm is in the Hold Off mode of operation.
Alarm Enable: Reports if an Alarm is enabled. It will only be enabled if the Operation Mode is not NONE and an output has been allocated.
Alarm Active: Reports if the Alarm has gone into the alarm zone. The *Alarm Subsidiary* and *Control* mode do not effect the operation of this flag.

The Alarm Status register can also be used to ***Reset*** a Latched alarm or ***Set*** the Hold-Off operation. Bit 0 and Bit 1 are writable and can be manipulated by the user as required.
N.B. Bit 2 and 3 are read only.

7 CALogix Logic Module

The CALogix Logic modules provide interfaces to integrate Programmed Logic Control functions into the system operation.

An example of a temperature control system with additional logic control is shown below: -

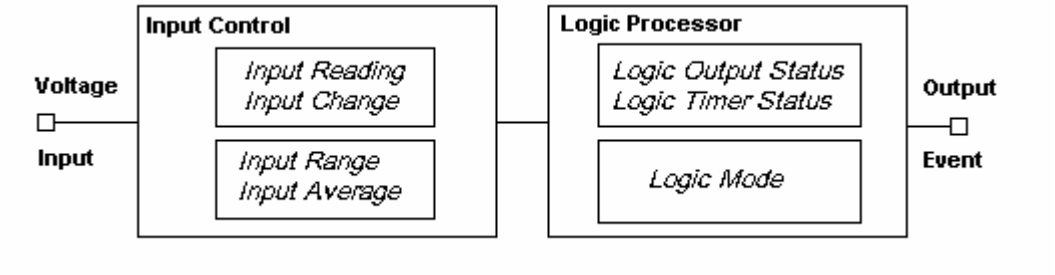


The embedded logic controller utilises a 3-input, 3-output logic module to provide the interface with the external plant. The inputs are user configurable, providing 3 operational ranges of 5, 10 and 24 volts. The input module can resolve inputs to 0.1 % of range and have a sampling period of 10 per second. The outputs are always relay type.

The embedded Logic Processor has an array of user-configurable Logic Blocks which provide such functions as Timers, Event-Counters, Comparitors and Boolean Logic. The array is programmed using the CALogix configuration package, in a proprietary CALogix Logic Module data format. The configuration of these User Define Logic Blocks is beyond the scope of this document.

7.1 Module Block Diagram

A block diagram of the Logic module controller is shown below: -



The Logic module is divided into 2 functional block. These blocks are detailed below.

7.2 Logic Input Control

This functional block provide configuration and status of the Logic Module input hardware.

The following table gives details of the Input Control registers: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
Input Reading 1	Logic Input Reading for input channel 1.	Float [R]	**0x0A1C** (2588)	**0x0A1E** (2590)	**0x0A20** (2592)	**0x0A22** (2594)
Input Reading 2	Logic Input Reading for input channel 2.	Float [R]	**0x0A24** (2596)	**0x0A26** (2598)	**0x0A28** (2600)	**0x0A2A** (2602)
Input Reading 3	Logic Input Reading for input channel 3.	Float [R]	**0x0A2C** (2604)	**0x0A2E** (2606)	**0x0A30** (2608)	**0x0A32** (2610)
Input Change 1	Input Status value for input channel 1.	Byte [R]	**0x0A34** (2612)	**0x0A35** (2613)	**0x0A36** (2614)	**0x0A37** (2615)
Input Change 2	Input Status value for input channel 2.	Byte [R]	**0x0A38** (2616)	**0x0A39** (2617)	**0x0A3A** (2618)	**0x0A3B** (2619)
Input Change 3	Input Status value for input channel 3.	Byte [R]	**0x0A3C** (2620)	**0x0A3D** (2621)	**0x0A3E** (2622)	**0x0A3F** (2623)
Input Average 1	Input Filter Averaging value for input channel 1.	Byte [RW]	**0x0A40** (2624)	**0x0A41** (2625)	**0x0A42** (2626)	**0x0A43** (2627)
Input Average 2	Input Filter Averaging value for input channel 2.	Byte [RW]	**0x0A44** (2628)	**0x0A45** (2629)	**0x0A46** (2630)	**0x0A47** (2631)
Input Average 3	Input Filter Averaging value for input channel 3.	Byte [RW]	**0x0A48** (2632)	**0x0A49** (2633)	**0x0A4A** (2634)	**0x0A4B** (2635)
Input Range 1	Input voltage range for input channel 1.	Byte [RW]	**0x0AE8** (2792)	**0x0AE9** (2793)	**0x0AEA** (2794)	**0x0AEB** (2795)
Input Range 2	Input voltage range for input channel 2.	Byte [RW]	**0x0AEC** (2796)	**0x0AED** (2797)	**0x0AEE** (2798)	**0x0AEF** (2799)
Input Range 3	Input voltage range for input channel 3.	Byte [RW]	**0x0AF0** (2800)	**0x0AF1** (2801)	**0x0AF2** (2802)	**0x0AF3** (2803)

7.2.1 Logic Input Configuration

The Input Range register provides configuration of the Logic Module input hardware.

The encoding is as follows: -

Register Value	Description
0	Invalid / A read of the *Input Range* register where no Logic Module is fitted will return **zero.**
1	0 – 5 Volt (max) input range.
2	0 – 10 Volt (max) input range.
3	0 – 24 Volt (max) input range.

The Input readings are filtered to remove electrical noise and disturbances from the input source. The filter provides a smooth signal output reading but at the expense of frequency response. The default value has been optimised for general use. For a system requiring the best frequency response the Averaging can be reduced.

To enable the Input Filter the *Input Average* register will requires a value from 1 to 255 samples. A value of zero will turn averaging OFF.

7.2.2 Reading Logic Inputs

Each Logic Module input can be interrogated to determine its voltage input and current *logical* state.

The voltage readings are returned as floating point numbers with a resolution and accuracy depending on the range selected. A one-percent over range exists on all inputs to allow for noise fluctuations at the maximum voltage for the range selected. Inputs greater than the range maximum plus the over range allowance will be truncated. No under-range allowance exist and the minimum value will always be zero.

In addition to providing the Input Reading as a voltage level, the Logic Module will interpret the value as a logic level.
This is reported in the *Logic Change* register and can only have two values: -

0 = Input LOW (OFF) Logic 0
1 = Input HIGH (ON) Logic 1

The switching point for determining whether the Logic Input is in the HIGH or LOW state is set at less-than 33 % of input max for the LOW state and higher-than 66 % of input max for the HIGH state. When the signal is within the region between the two switching points the Logic HIGH or LOW value of the *Input Change* register will remain unchanged from its previous value.

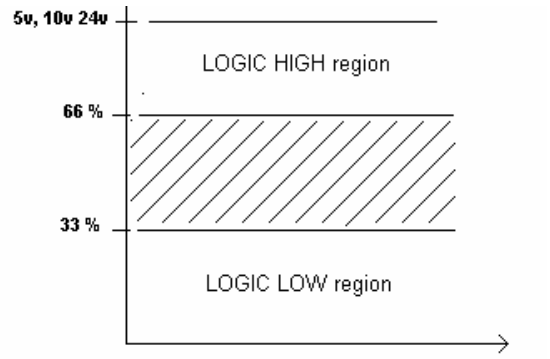


Diagram to show Logic-LOW-HIGH switching characteristics within the Logic Module.

7.3 Logic Control

Using the Logic Mode register allows the user to enable or disable the Logic Processor.
When in the *Run* state, the programmed user-defined logic block sequence will be actioned once every 100 mS. New user-defined logic block sequence are uploaded into the Logic Processor memory only when the Logic Mode register makes a transition from *Stop* to *Run* mode.

When the Logic Processor is put into the *Stop* state the user-defined logic block sequence will halt at the end of its current cycle and all the allocated outputs will be released.

The Logic Mode register is as follows: -

Register Name	Description	Data Size/Format	Default	Address
Logic Mode	Logic Control	Byte [RW]	0 (*Stop*)	**0x07CA** (1994)

Register Value	Description
0	*Stop* logic sequences and disable Logic Processor.
1	*Run* logic sequences and enables Logic Processor.

7.4 Logic Status

Whilst the Logic is running the operation of the Timers and Logic Block outputs can be monitored. A detailed description of the format of these registers is beyond the scope of this document.

Register Name	Description	Data Size/Format	Default	Address Range
Logic Timer value 1 –16	Logic Block Timer / Counter value 1 – 16	Word [R] x 16	0	**0x0AD0 to 0x0ADF**
Logic Output Status	Logic Block output status flag array (Total: 64 bit flags)	Word [R] x 4	0	**0x0AF4 to 0x0AF7**

8. Programmer

The embedded Programmer controller can utilises any of the PID module to provide automated Ramp / Soak control to for the process control.

The Programmer controller utilises an array of user-defined Program Segments, which provide such functions as Ramp, Soak, Loop etc. The array is programmed using the CALogix configuration package, in a proprietary CALogix data format. The configuration of these user-define Program Segments is beyond the scope of this document.

The following table gives details of the Programmer controller registers: -

Register Name	Description	Size/Format	Address Module 1	Address Module 2	Address Module 3	Address Module 4
Program Mode	Programmer Operational Mode.	Byte [RW]	**0x0787** (1927)	**0x0788** (1928)	**0x0789** (1929)	**0x078A** (1930)
Program Number	Program Number Range 1 to 31.	Byte [RW]	**0x0783** (1923)	**0x0784** (1924)	**0x0785** (1925)	**0x0786** (1926)
Program Segment Number	Current Program Segment Number.	Byte [R]	**0x078B** (1931)	**0x078C** (1932)	**0x078D** (1933)	**0x078E** (1934)
Program Segment Type	Current Program Segment type.	Byte [R]	**0x078F** (1935)	**0x0790** (1936)	**0x0791** (1937)	**0x0792** (1938)
Program Segment Time	Elapsed time of Ramp / Soak segment (seconds).	DWord [R]	**0x0994** (2452)	**0x0996** (2454)	**0x0998** (2456)	**0x099A** (2458)
Program Loop Count	Loop count of currently running program.	Word [R]	**0x077F** (1919)	**0x0780** (1920)	**0x0781** (1921)	**0x0782** (1922)
Program Setpoint Moving	Programmer moving setpoint.	Float [R]	**0x0799** (1945)	**0x079A** (1947)	**0x079B** (1949)	**0x079C** (1951)
Program Setpoint Start	Programmer Ramp / Soak setpoint start value.	Float [R]	**0x07A1** (1953)	**0x07A3** (1955)	**0x07A5** (1957)	**0x07A7** (1959)
Program Event Input	Set Program Event Input flag.	Word [W]	**0x099C** (2460)	**0x099D** (2461)	**0x099E** (2462)	**0x099F** (2463)

8.1 Selecting and Starting Programs

To select and run programs two registers are required.

To run any of the used-defined programs the *Program Number* must first be selected. Values ranging from 1 to 31 are valid. If a value for the *Program Number* is used which does not have a corresponding entry in the user-defined Program Segments array the *Program Number* register will be set back to **zero** when an attempt is made to run the program.

To control the operation of the program selected, the *Program Mode* register is used.

The following table gives the settings: -

Register Value	Mode	Description
0	OFF	In the *OFF* state the Programmer will not control the process or effect any outputs. If the *OFF* value is written whilst the Programmer is running the controller will *ABORT* the current program.
1	RUN	Starts a Program running.
2	HOLD	Suspend a Program but still retain control.
3	STOP	The *STOP* value is reported when the Programmer has

		finished the last Segment of its current program. This value cannot be written to the *Program Mode* register and is provided for status only. The user must write the *OFF* value after the STOP mode has been reported, to return the Programmer controller to the idle state.
4	INVALID	Invalid value.
5 or 6	HOLD-BACK	This status value will only be reported if the Program Segment running has an active HOLD-BACK segment. The user cannot set the Programmer into HOLD-BACK manually.

8.2 Program Status

When an active Program is running the Programmer controller reports the status of the Program using several read-only register as detailed below:-

8.2.1 Segment Status

Program Segment Number: Reports the current Segment number. Range 1 to 127. When the Programmer is in the *OFF* mode the *Program Segment Number* will be **zero.**

Program Segment Time: Reports the elapsed time, in seconds, of the current Program Segment. Where the Segment is a *Loop or Step* type the reported time will be **zero.**

Program Loop Count: The Loop count of the currently running program. If the *Program Mode* is *OFF* the count will be **zero.**

Program Segment Type: Reports the current Segment Type

Register Value	**Description**
0	Unknown: Program Error.
1	SPRH: Setpoint RAMP with Hold-Back.
2	SPR: Setpoint RAMP.
3	STEP: Step to new Setpoint.
4	SOAK: Remain at current Setpoint for #N secs/mins
5	LOOPS: Loop (small)
6	LOOPL: Loop (large)
7	CALL: Call another Program
10	EIP: Event Input
12	EOP: Event Output

8.2.2 Setpoint Status

Two registers track the progress of the Programmers Setpoint generator.

Program Setpoint Moving: Tracks RAMP segment Setpoint. The value is expressed in the same units as the PV. This value will be updated once per second.

Program Setpoint Start: Tracks the initial value of RAMP segments. The Program Setpoint Moving and Program Setpoint Start will have the same value for STEP or SOAK segments.

8.3 Program Event Input

The Programmer has the ability to react to external inputs either from a Logic Modules or via control from a Modbus command. The later is achieved using the *Program Event Input* register.

This register is an array of 16 bit-flags which can be set to 1 or reset to 0 as desired by the user. If a Program Segment has been set-up to react to Event-Input, the Programmer will not advance its program sequence from that segment until the appropriate bit-flag has been set.

The Event-Input Segments have values from 1 to 16 which correspond directly to the bits within the *Program Event Input* register, which range from bit 0 to bit 15.

CAL Controls

CAL Controls Ltd
Bury Mead Road, Hitchin, Herts, SG5 1RT, UK
Tel +44 (0) 1462 436161
Fax +44 (0) 1462 451801
e-mail sales@cal-controls.co.uk
www.cal-controls.com

CAL Controls Inc
1117 S. Milwaukee Av, Libertyville, IL60048, USA
Tel (847) 680-7080
Fax (847) 816-6852
e-mail sales@cal-controls.com
www.cal-controls.com