

---

# ***Contents***

<b>Overview .....</b>	<b>3</b>
Features .....	3
<b>Quick Start .....</b>	<b>4</b>
Set Up .....	4
First Example .....	4
Second Example .....	6
<b>Command Construction .....</b>	<b>8</b>
Start Character [>] .....	8
Module Address [AA] .....	8
Position Field [PPPP] .....	9
Checksum [SS] .....	10
Carriage Return [<CR>] .....	11
<b>Returns .....</b>	<b>12</b>
<b>Error Codes .....</b>	<b>13</b>
<b>Communication Parameters .....</b>	<b>14</b>
<b>Analogue Command Set Summary .....</b>	<b>15</b>
<b>Set Up Commands .....</b>	<b>17</b>
Power Up Clear     A .....	17
Reset            B .....	18
Set Turn-around Delay     C[Data] .....	19
Set Analogue Watchdog Delay   D[data] .....	20
Identify Controller Type    F .....	21
<b>Output Commands .....</b>	<b>22</b>

---

Write 4-20mA outputs	J[positions][data]	22
<b>Input Commands</b>		<b>24</b>
Input Reading Speed		24
Set Input Type	k[positions]	25
Read Analogue Inputs	L[positions]	27
Read Temperature Inputs	l[positions]	29
<b>Input Averaging Commands</b>		<b>31</b>
Start Input Averaging	T[positions][data]	31
Read Average Complete Bits	i	33
Read Input Average Data	U[positions]	34
Read Average Temperature Inputs	o[positions]	35
<b>Reference Tables</b>		<b>40</b>
Hexadecimal – Binary – Decimal Conversion Table		40
ASCII Character Table (Reduced)		41

---

## ***Overview***

OPTO22<sup>1</sup> or OPTOMUX<sup>1</sup> is the command set protocol used for controlling the **Control-it™** modules via the PC's serial port.

### ***Features***

- ASCII based protocol widely used in industry.
- Transmission speed from 1200 to 115.2k baud.
- External connection to PC serial ports.

---

<sup>1</sup> OPTO22 and OPTOMUX are trademarks of OPTO 22

---

## Quick Start

This section uses two example commands to briefly explain the construction and use of the Optomux command set.

A terminal emulator program (e.g. Telix or HyperTerminal) is used to manually construct and send the commands.

A reduced ASCII table, and a Decimal – Binary – Hexadecimal conversion table are provided on pages 40 and 41.

### Set Up

- Set up the **Control-it™** module as described in the Installation Guide in this manual.
- Connect your **Control-it™** module to the PC's serial port using an RS232 to RS485 converter such as the **Control-it™ 5001**.
- Set the module address to 0.

If you wish to use another address you will need to substitute it in the following examples.

- Ensure that the terminal emulator program is correctly configured as per table 5 below.

Data Parameter Set Up	
Com Port	Matched to the port where the converter is connected
Baud Rate	Matched to the setting on the module (refer to the Installation Manual)
Data Length	8 bit
Parity	No parity
Stop Bits	1

Table 5 – Data parameter set up.

### First Example

The first command to try is the 'A', or Power Up Clear, command.

---

Not only is it a simple one to construct, it is also the first command required by all **Control-it™** remote modules.

- In your terminal emulator program type the following characters after each other:
  - > [‘greater than’ sign (ASCII char 63)]
  - 0 [zero (ASCII char 48)]
  - 0 [zero (ASCII char 48)]
  - A [upper case letter (ASCII char 65)]
  - ? [question mark (ASCII char 63)]
  - ? [question mark (ASCII char 63)]
- The command will now look like this:

**>00A??**

‘>’ is the first character of every command. It tells the remote module that a new command is starting and to listen.

‘00’ is the address, in Hexadecimal, of the remote module.

‘A’ is the power up clear command.

‘??’ replaces the checksum which will be discussed later.

- Add the **<CR>** [carriage return character (ASCII char 13)] by pressing Enter.  
<CR> is the last character on all commands, telling the remote module that the command is complete.
- The module will acknowledge this command by returning an ‘A’ character.  
  
If this does not happen check both your module and terminal program set-ups.
- The module is now cleared and ready to receive more commands.

---

## Second Example

The second command to try involves measuring a dc voltage applied to the 5050.

The 'L', or Read Analogue Inputs, command allows us to specify the positions to be read and returns a 4-character hex number for each.

- Connect a dc voltage, between 0 and 10V, to position 7 as described in **Connecting Inputs & Outputs** in the **Installation Manual**.

**NB** We are using the 0 – 10V range as this is the factory default setting.

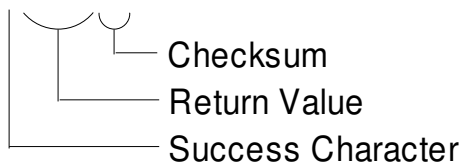
- In your terminal emulator program type the following command:

**>00L8014<CR>**

Remember, <CR> is added by pressing the ENTER key.

- The return will be

**A1dddss**



If this does not happen check both your module and terminal program set-ups.

If the return value is 0000, the polarity of the voltage is reversed.

- Convert the last three digits of the return value (ddd) from hex to decimal and multiply the result by 0.0025.

**This is the voltage applied to position 7.**

Taking another look at the command:

- '>', '00' and '<CR>' are the same as for the previous example.

- 'L' is the Read Analogue Inputs command.
- '80' is the position fields, and indicates which positions, 0 – 15 (or 0 – 7 for differential modules), are to be read. A binary number is created by placing a '1' at each position to be read, and is then converted to Hexadecimal for use in the command. Table 6 demonstrates this.

Position Field calculation for Second Example																
Positions	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Hexadecimal	0				0				8				0			

Table 6 - Position Field calculation for Second Example

For more detail on this refer to **Position Field**, page 9.

- '14' is the checksum for this command. This is the last two characters of the sum of the hexadecimal ASCII values of each character in the command, excluding the '>' start character. Table 7 demonstrates this.

Checksum calculation for Second Example								
Command	>	0	0	L	8	0	Sum	Checksum Used
ASCII (Hex)	N/A	30	30	4C	38	30	= 114	14

Table 7 - Position Field calculation for Second Example

For more detail on this refer to **Checksum**, page 10.

---

## **Command Construction**

The Optomux commands are made up of individual ASCII characters, each with a special purpose.

The general command from the PC takes the following general form:

**>AACPPPPDDSS<CR>**

Component	Description
>	The start character. (ASCII char 62)
AA	The address of the target module in Hexadecimal from 00 to FF.
C	The Command set character. Refer to specific command for further details.
PPPP	The position field used with some commands to define the target position(s).
DD	The data or modifier used with some commands. Refer to the specific command description for further details.
SS	The command Checksum.
<CR>	The end character - Carriage Return. (ASCII char 13)

### ***Start Character [>]***

The first character of every command is always the 'greater than' sign (ASCII char 62).

- This alerts the **Control-it™** module that a new command is starting and to listen.

### ***Module Address [AA]***

This is the Hex address of the target module.

- The decimal address set by the Address Jumpers on the module must be converted to its hexadecimal equivalent for

---

use in the command.

A hex to decimal conversion table is provided on page 40.

- As this is always 2 characters, address' 0 to 15 (decimal) are expressed as 00 to 0F (hex).

### ***Position Field [PPPP]***

The position field contains between 0 and 4 hex digits, each representing 4 positions on the module.

- 0 hex digits control all positions.
- 1 hex digit controls positions 0 to 3.
- 2 hex digits control positions 0 to 7
- 3 hex digits control positions 0 to 11
- 4 hex digits control positions 0 to 15

In its binary form each bit represents a position on the module with the least significant bit (LSB) representing Position 0. By placing a '1' at the bit or bits that represent the target position(s), the field number (in Hex) can be calculated.

If no position characters are sent, the assumed value is FFFF (all 1's).

A hex to binary conversion table is provided on page 40.

### **Examples:**

Table 8 shows three Position Field examples.

- On the left are the example module positions to be targeted.
- The centre section lays out the module positions and constructs a binary number by placing a 1 at each target position.
- The right hand column is the Hex equivalent of this binary number for use in the Optomux command.

Position Field Examples					
Target Positions	Module Positions				Position Field (Hex)
	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	
3				1 0 0 0	8
1 & 6			0 1 0 0	0 0 1 0	42
6, 7 & 14	0 1 0 0	0 0 0 0	1 1 0 0	0 0 0 0	40C0

Table 8 - Position Field Examples

### **Checksum [SS]**

The Checksum is your protection against corrupted messages being received as valid messages.

The Checksum is calculated by summing the ASCII decimal values of each character, converting the result to Hex and using the last two digits. The '>' is not included in the Checksum.

- For purposes of testing, the Checksum can be replaced with "??" in all commands.
- For final use in production equipment the Checksum must be calculated and sent.

#### **Example:**

Command	>	0	9	U	1	2	3	4
ASCII value (decimal)	Not used	48	57	85	49	50	51	52

- Adding all the values gives  $48 + 57 + 85 + 49 + 50 + 51 + 52 = 392$
- $392$  (decimal) =  $188$  (hex) therefore the checksum is 88

- 
- The complete command with the Checksum is now

>09U123488<CR>

### ***Carriage Return [<CR>]***

<CR> is always the last character sent. It tells the **Control-it™** module that the command is completed.

- The carriage return is executed by adding ASCII char 13 immediately after the checksum in the command.
- In a manual set up, such as with Telex or Hyperlink, the <CR> is added when the ENTER key is pressed.

---

## **Returns**

A successfully completed command will return an 'A' and <CR>

- If the command required data to be returned then an 'A', data, checksum, and <CR> is returned

i.e. A[data][checksum]<CR>

An unsuccessful command will return an error code indicating the problem.

- A complete list of error codes is provided in the next section.

---

## Error Codes

Errors will occur if a command is constructed incorrectly (e.g. wrong command character or incorrect data) or there is a problem with transmission (e.g. checksum error or message time out).

The return from the module upon detecting an error is

N0x            where x is the relevant error number.

The red WD/ERR LED on the module itself will also display this number, as repeated flashes.

Table 9 list the error codes and their meanings.

Opto22 Error Codes		
Return	Name	Description
N00	Power Up Clear Expected	A command other than 'A' was attempted after power-up or power failure.  Once this error is received, the next command will be executed normally.
<b>Important</b> Receiving the N00 error means the module has undergone a power-up sequence and will have reset all characteristics to default. These will need to be re-initialised.		
N01	Undefined Command.	The Command character is not a legal character.
N02	Checksum Error.	The checksum did not match the sum of the characters.
N03	Buffer Overrun.	Command contained more than 16 characters.
N04	Non-Printable ASCII Character Received.	Command ignored
N05	Data Field Error.	[positions], [data], [modifier] or [type] numbers are out-of-range or not recognised.
N06	RS485 Watchdog Timeout Error.	No activity on RS485 line before watchdog time set by D or m command has timed-out.
N07	Specified Limits Invalid	

Table 9 – Opto22 Error Codes

---

## Communication Parameters

Opto22 & Optomux uses the communication parameters shown in table 10.

Communication Parameters	
Baud Rate	Matched to the setting on the module (refer to <b><i>Jumper Configuration</i></b> in the <b>Installation Manual</b> )
Data Length	8 bit
Parity	No parity
Stop Bits	1

Table 10 – Optomux Communication Parameters

---

## **Analogue Command Set Summary**

General Command:

>[Address]**Char**[Additional Structure][Checksum]<**CR**>

(l/c) indicates lower case character.

<b>Description</b>	<b>Char</b>	<b>Additional Structure</b>	<b>Page</b>
<b>Set-Up</b>			
Power Up Clear	A		17
Reset	B		18
Set Turn-around Delay	C	[data]	19
Set Analogue Watchdog Delay	D	[positions][data]	20
Identify Remote Module Type	F		21
<b>Output</b>			
Write 4-20mA Outputs	J	[positions][data]	22
<b>Input</b>			
Set Input Type	k (l/c)	[positions][data]	25
Read Analogue Inputs	L	[positions]	27
Read Temperature Inputs	l (l/c)	[positions]	on page 29
<b>Input Averaging</b>			
Start Input Averaging	T	[positions][data]	on page 31
Read Average Complete Bits	i (l/c)		33
Read Input Average Data	U	[positions]	on page 34

---

Read Average Temperature Inputs	o (l/c)	[positions]	on page 35
------------------------------------	---------	-------------	------------------

---

# ***Set Up Commands***

## ***Power Up Clear***

**A**

### USE

Informs the Control-it 50x0 that a normal power up has occurred.

### STRUCTURE

A

### COMMENTS

When power is applied to the Control-it 50x0 modules they expect the first command to be an 'A' command. If this is not so then a N00 error is returned and the command sent will not be carried out.

If during normal operation an N00 error is received by the controlling computer then an unexpected power down and power up has occurred on a remote module.

### RETURNS

A<CR>

---

**Reset****B**USE

Resets the Control-it 50X0 back to power up conditions.

STRUCTURE

B

COMMENTS

Power on or reset causes the following actions to be performed

- All outputs turned off
- All positions then configured as inputs
- Turn around delay = 0
- Counters / duration timers cancelled
- Latches cleared
- Timer resolution = 10ms

RETURNS

A<CR>

---

**Set Turn-around Delay****C[Data]****USE**

Causes the Control-it 50x0 to wait for a time before responding to commands.

**STRUCTURE**

C[Data]      NB - Data field must be included.

**COMMENTS**

The (data) causes the following delays

<b>Data</b>	<b>Function</b>
0	Turn around delay = 0
1	10mSec
2	100mSec
3	500mSec

**RETURNS**

A<CR>

---

**Set Analogue Watchdog Delay****D[data]****USE**

Cause the Control-it 50X0 to monitor activity on the RS485 bus and take the specified action if there is no activity for the programmed time.

This command has no affect on the Control-it 5040 as it only has inputs.

**STRUCTURE**

D[data]                    NB - Data field must be included.

**COMMENTS**

The (data) field contains a single ASCII digit between 0 and 7 that specifies the time interval and desired action. Times and actions are shown below.

<b>(Data)</b>	<b>Time</b>	<b>ACTION</b>
0		Watchdog disabled
1	10 secs	Turn all outputs OFF
2	1 minute	Turn all outputs OFF
3	10 minutes	Turn all outputs OFF
4		Watchdog disabled
5	10 secs	Turn output 0 ON, all other outputs OFF *
6	1 minute	Turn output 0 ON, all other outputs OFF *
7	10 minute	Turn output 0 ON, all other outputs OFF *

\* Only the Control-it 5020 has a relay in positions 0 to 7.

**RETURNS**

A<CR>

---

***Identify Controller Type***

***F***

USE

The Control-it 5050 will always report itself as an analogue controller  
i.e. the response is 01.

STRUCTURE

F

RETURN

A0161<CR>

---

# Output Commands

**Write 4-20mA outputs**

***J[positions][data]***

USE

Sets the current that the 4 – 20mA outputs will deliver.

STRUCTURE

J[positions][data]

COMMENTS

[positions] contains 4 hex digits. All 4 digits must be present.

The 5050 has two 4-20mA outputs, at 14 & 15, so the following are valid numbers for [positions].

[positions]	Output
4000	14
8000	15
C000	14 & 15

[data] contains the value of current to be output.

- The range is from 320H to FA0H (800 to 4000).
- There are 200 counts per mA

i.e. each step has a current value of 0.005mA

**NB** Because the 4-20mA outputs are completely isolated they will not respond to commands until voltage is placed across the appropriate Output. When voltage is applied the default current is 4mA.

---

### EXAMPLE

>0FJ40007D02F<CR>

The Output position is 14 [4000(hex)].

The Output current is 7D0(hex) = 2000(dec)

$$2000 \times 0.005\text{mA} = 10\text{mAmps.}$$

0F is the module address and 2F is the Checksum.

---

# ***Input Commands***

## ***Input Reading Speed***

The 5050 module uses the MAX132 ADC, a very accurate multi-slope integrating analogue-to-digital converter, with automatic 50Hz or 60Hz noise rejection. This makes the 5050 ideal for measuring a wide range of signal types including low-level signals such as temperature probes.

### Input Refresh Time

To achieve this precision and hum rejection the 5050 requires 1/10 second to convert an input. Since the 5050 constantly cycles around all enabled inputs, the refresh time for each becomes:

$$\text{Refresh time} = 1/10 \text{ sec} \times \text{No. of Enabled Inputs.}$$

For example, if 4 inputs are enabled the refresh time for each input will be 0.4 seconds.

For this reason, it is strongly recommend that all unused inputs are disabled using the ***Set Input Type*** command.

### Data Request Time

As each input is sampled, the result is stored in memory on the module. When the host computer requests a reading, this stored value is returned immediately rather than waiting for a real-time conversion, releasing the RS485 bus much sooner.

---

## ***Set Input Type***

***k[positions]***

### USE

Enables specified positions and defines their input type, enabling temperature, voltage and current to be read directly. Can also disable positions to improve overall sampling rate.

### STRUCTURE

k[positions][data]

### COMMENTS

- This command must be used multiple times to set different input types across the same module.
- Disabling inputs that are not used will speed up the input scan time within the 5050 module.
- Once the input type has been set, it is stored in non-volatile memory on the module and will remember its setting even after power down.
- When setting up inputs it may be necessary to change the jumpers on the 5050.

[positions] indicates the module positions to be set as described in ***Position Field*** (page 9).

- 0 indicates position to be ignored
- 1 indicates position to be set

[data] is a single ASCII character that defines the input type as follows

Data	Input Type
0	Disable input
4	J type temperature probe

---

5	K type temperature probe
8	T type temperature probe
D	0 to 1V input
E	0 to 10V input
F	4 to 20mA input

---

## ***Read Analogue Inputs***

## ***L[positions]***

### USE

Returns 4 HEX characters for each position specified.

- If a temperature probe is fitted then use the l command (lower case L).

### STRUCTURE

L[positions]

### COMMENTS

[positions] indicates the module positions to be read as described in ***Position Field*** (page 9).

- 0 indicates position to be ignored
- 1 indicates position to be read

### RETURN

A[data][checksum]<CR>

[data] returns values as 4 digit hex numbers in the range 1000H to 1FFFH.

Therefore:

$$\text{Input Value} = ([\text{data}] - 1000\text{H}) \times \text{Resolution of Range}$$

where Resolution is found in the following table:

Range	Resolution	Steps / Unit	Input limits
0 to 1V	0.00025V	4000 / V	0 – 1.02375V
0 to 10V	0.0025V	400 / V	0 – 10.2375V
0 to 20mA	0.005mA	200 / mA	0 – 20.475mA

- 
- Counter values are returned in sequence from highest to lowest position.
  - An attempt to read a position that has not been set up using the **Set Input Type** command (page 25) will return ????.
  - Reading positions 8 to 15 on an 8 input 5050 will return ????
  - If the input is –ve then a 0000H is returned.
  - If the input is over range then a value >1FFFH is returned.

### EXAMPLES

To read from position 6, a 0 to 1V input, at address 15 the command would be

>0FL004086<CR>

The return is

A1100??<CR>

- The data returned is 1100H.
- The input Count is 1100H – 1000H = 100H =256 (dec).
- Multiply this by the 0-1V Resolution (0.00025V)

Voltage = Count x Resolution

$$= 256 \times 0.00025V$$

$$= 0.064V$$

---

## ***Read Temperature Inputs***

**I[positions]**

### USE

Causes the 5050 to return the temperature in degrees Celsius. For 0-1V, 0-10V and 0-20mA inputs use the READ INPUT DATA (Command L)

### STRUCTURE

I[positions]

**NB** this is a lower case L, not an upper case i.

### COMMENTS

[positions] indicates the module positions to be read as described in **Position Field** (page 9).

- 0 indicates position to be ignored
- 1 indicates position to be read

### RETURN

A[data][checksum]<CR>

[data] returns values as 4-digit signed hex numbers in 2s-compliment form. i.e.

Hex: 8000, ... FFFF, 0000, 0001, ... 7FFF.  
Dec: -32766, ... -1, 0, 1, ... 32767.

- Temperature, in degrees Celsius, is calculated by converting [data] to decimal and dividing by 16. i.e.

$$\text{Temp} = \frac{\text{data (decimal)}}{16}$$

- Alternatively, the 3 most significant digits, converted to decimal, is the integer value, and the 4<sup>th</sup> digit is the 16<sup>ths</sup>.

- Temperature can be measured in three ranges as follows.

Temperature Probe Type	Minimum	Maximum
J	0 °C	760 °C
	0000H	2F80H
K	0 °C	1000 °C
	0000H	3E80H
T	-100 °C	400 °C
	F9C0H	1900H

- Returned values are from the highest to lowest module positions.
- An attempt to read a position that has not been set up using the **Set Input Type** command (page 25) will return ????.
- Reading positions 8 to 15 on an 8 input 5050 will return ????
- Channels that read below the scale of the probe type set will return a -273°C. If the input is above the probe type set the temperature returned is 2047°C

### EXAMPLES

Inputs 2 and 5, of module 15, are to be read.

>0FI0024A8<CR>

Assume the response is A1234FF7F14<CR>

The data returned is 1234H for channel 5 and FF7F for channel 2.

$$\begin{aligned} \text{So channel 5} &= 123\text{H} + 4/16 \text{ °C} \\ &= 291 + 0.25 \text{ °C} = 291.25 \text{ °C} \end{aligned}$$

$$\text{And channel 2} = (\text{FF7FH} - 10000\text{H}) = -81\text{H} = 129$$

$$\text{so temperature} = \frac{129 \text{ °C}}{16} = -8.0625 \text{ °C}$$

---

## ***Input Averaging Commands***

Input averaging is useful for the reduction of noise in data. Because the 5050 uses a MAX132 A/D converter with 50 and 60Hz rejection it is probably not necessary to average samples.

$$\text{Average} = \frac{((N - 1) \times \text{Old Average}) + \text{New Reading}}{N}$$

As samples are taken, N increments from 1 to the value specified in [data]. When it reaches that value, it remains at this value and sets the position's Input Average Complete Bit.

Average continues to be computed when each sample is taken. E.g. if [data] was 3 then N would take on the values of 1, 2, 3, 3, 3, ...

### ***Start Input Averaging***

***T[positions][data]***

#### USE

Start averaging the value of input positions over a specified number of samples.

#### STRUCTURE

T[positions][data]

#### COMMENTS

[positions] indicates the module positions to be set as described in ***Position Field*** (page 9).

- 0 indicates position to be ignored
- 1 indicates position to be set

[data] contains 0 to 4 ASCII-Hex digits specifying the number of consecutive readings to be averaged.

- If there is no data field, then the averaging is switched off.
- The sample rate is 100mSec per channel.
  - The number of active channels is defined by the k command.

- 
- For example, if 4 channels are active then the sample rate is 400mSec.
  - To determine when the samples are complete use the **Read Average Complete Bits** command (page 33).
  - Reading the samples is done using the **Read Input Average Data** command (page 34) or **Read Average Temperature Inputs** (page 35).

### EXAMPLES

To set positions 2 and 3 at address 15 to sample 4 times the following command would be used.

```
>0FT000C48E<CR>
```

---

## ***Read Average Complete Bits***

*i*

### USE

Allows the PC to check which positions have completed averaging.

### STRUCTURE

*i*

### RETURN

The 5050 returns a 4-digit Hex character representing each position's ACB in the same fashion as the ***Position Field*** (page 9).

- Any positions that have completed the number of samples set with the ***Start Input Averaging*** command (page 31), will have their appropriate bits set.

### EXAMPLES

Request from module at address 15 the AVERAGE COMPLETE BITS

>0Fi55cr

Assume the response is

A005309

The data returned is 0053, indicating that positions 6, 4, 1 & 0 have reached the specified number of samples.

---

## ***Read Input Average Data***

***U[positions]***

### USE

Causes the 5050 to return the averaged input values. Use this for 0-1V, 0-10V and 0-20mA inputs only.

### STRUCTURE

U[positions]

### COMMENTS

[positions] indicates the module positions to be read as described in ***Position Field*** (page 9).

- 0 indicates position to be ignored
- 1 indicates position to be read

### Return

A[data][checksum]

[data] is in the same fashion and ranges as ***Read Analogue Inputs*** (page 27).

- If the 5050 has not completed the number of samples specified in the ***Start Input Averaging*** command (page 31) then the average so far is returned.
- If averaging has not been specified for a channel then the current value is returned.

---

## ***Read Average Temperature Inputs*    o[positions]**

### USE

Causes the 5050 to return the averaged temperatures.

### STRUCTURE

o[positions]

### COMMENTS

[positions] indicates the module positions to be read as described in ***Position Field*** (page 9).

- 0 indicates position to be ignored
- 1 indicates position to be read

### Return

A[data][checksum]

[data] is in the same fashion and ranges as ***Read Temperature Inputs*** (page 29).

- If the 5050 has not completed the number of samples specified in the ***Start Input Averaging*** command (page 31) then the average so far is returned.
- If averaging has not been specified for a channel then the current value is returned.

---

## **High Resolution Read Analogue Inputs**      *t[positions]*

### USE

This is the high resolution version of the L command. Returns 4 HEX characters for each position specified.

### STRUCTURE

*t[positions]*

### COMMENTS

*t[positions]* indicates the module positions to be read as described in **Position Field** (page 9).

- 0 indicates position to be ignored
- 1 indicates position to be read

### RETURN

A[data][checksum]<CR>

[data] returns values as 4-digit signed hex numbers in 2s-compliment form. i.e.

H	7	0	0	F	80
e	F	0	0	F	00.
x	F	0	0	F	
:	F,	1	0	F	
		,	,	,	
D	3	1	0	-	-
e	2	,	,	1	32
c	7			,	76
:	6				6.
	7,				

Therefore:

Input Value = [data] x Resolution of Range

where Resolution is found in the following table:

---

<b>Input Type and Scaling</b>		
<b>Input type</b>	<b>Steps size</b>	<b>Steps / unit</b>
+/- 1V	31.25uV	32000/Volt
+/- 10V	312.5uV	3200/Volt
+/- 20mA	0.625uA	1600/mA
Return raw count	A/D converter count. 15bits + sign <sup>2</sup>	

- Counter values are returned in sequence from highest to lowest position.
- An attempt to read a position that has not been set up using the **Set Input Type** command (page xx) will return ????.
- Reading positions 8 to 15 on an 8 input 5050 will return ????

### EXAMPLES

To read from position 6, a +/-1V input, at address 15 the command would be

```
>0Ft004086<CR>
```

The return is

```
A1100??<CR>
```

- The data returned is 1100H.
- This equals 4352 decimal
- Multiply this by the +/-1V Resolution (0.00003125V)

---

<sup>2</sup> The Raw Count is not automatically zeroed

---

$$\begin{aligned}\text{Voltage} &= \text{Count} \times \text{Resolution} \\ &= 4352 \times 0.00003125\text{V} \\ &= 0.136\text{V}\end{aligned}$$

---

## **High Resolution Read Input Average Data** *u[positions]*

### USE

Causes the 5050 to return the averaged input values. Use this for 0-1V, 0-10V and 0-20mA inputs only. This is the high resolution version of the U command (uppercase U). Returns 4 HEX characters for each position specified.

### STRUCTURE

*u[positions]*

### COMMENTS

[positions] indicates the module positions to be read as described in **Position Field** (page 9).

- 0 indicates position to be ignored
- 1 indicates position to be read

### RETURN

A[data][checksum]

[data] is in the same fashion and ranges as **High Resolution Read Analogue Inputs** (page xx).

- If the 5050 has not completed the number of samples specified in the **Start Input Averaging** command (page xx) then the average so far is returned.
- If averaging has not been specified for a channel then the current value is returned.

---

# Reference Tables

## Hexadecimal – Binary – Decimal Conversion Table

Hexadecimal – Binary – Decimal Conversion Table					
Hex	Binary	Decimal			
		Most Significant Bytes		Least Significant Bytes	
		IV	III	II	I
0	0 0 0 0	0	0	0	0
1	0 0 0 1	4096	256	16	1
2	0 0 1 0	8192	512	32	2
3	0 0 1 1	12288	768	48	3
4	0 1 0 0	16384	1024	64	4
5	0 1 0 1	20480	1280	80	5
6	0 1 1 0	24576	1536	96	6
7	0 1 1 1	28672	1792	112	7
8	1 0 0 0	32768	2048	128	8
9	1 0 0 1	36864	2304	144	9
A	1 0 1 0	40960	2560	160	10
B	1 0 1 1	45056	2816	176	11
C	1 1 0 0	49152	3072	192	12
D	1 1 0 1	53248	3328	208	13
E	1 1 1 0	57344	3584	224	14
F	1 1 1 1	61440	3840	240	15

Using the table:

- Decimal value = IV+III+II+I  
e.g. D42 (Hex) = 0+3328+64+2 = 3394 (Dec)
- Binary value is a group of four bits for each hexadecimal character.  
e.g. D42 (Hex) = 1101,0100,0010 (Bin)

---

## ***ASCII Character Table (Reduced)***

<b>Reduced ASCII Character Table</b>								
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
13	D	<CR>	74	4A	J	101	65	e
48	30	0	75	4B	K	102	66	f
49	31	1	76	4C	L	103	67	g
50	32	2	77	4D	M	104	68	h
51	33	3	78	4E	N	105	69	i
52	34	4	79	4F	O	106	6A	j
53	35	5	80	50	P	107	6B	k
54	36	6	81	51	Q	108	6C	l
55	37	7	82	52	R	109	6D	m
56	38	8	83	53	S	110	6E	n
57	39	9	84	54	T	111	6F	o
62	3E	>	85	55	U	112	70	p
65	41	A	86	56	V	113	71	q
66	42	B	87	57	W	114	72	r
67	43	C	88	58	X	115	73	s
68	44	D	89	59	Y	116	74	t
69	45	E	90	5A	Z	117	75	u
70	46	F	97	61	a	118	76	v
71	47	G	98	62	b	119	77	w
72	48	H	99	63	c	120	78	x
73	49	I	100	64	d	121	79	y