
Contents

Overview	3
Features	3
System Set Up	4
Analogue Command Set Summary	5
Primary Commands.....	7
Open a 5101 Board	8
Set a Single Input Type	9
Read a Single Analogue Input	10
Close a 5101 Board.....	11
Simple Example Programs.....	12
The p5101hal.dll.....	12
Inclusions	12
Example	12
Advanced Commands	14
Identify Module Type	14
Read Module Status Register.....	17
Set Input Averaging.....	19
New Reading Available	21
Set a Single Analogue Output	22
Last Error Number.....	23
Last Error String	24
Reset Error Number	25
Advanced Example Programs	26

Commands for Multiple 5101 Boards	27
Error Codes	28
General Errors.....	28
Analogue Module Errors.....	29
Critical Warnings	30

Overview

Features

System Set Up

1. Install the PC interface device.
 - If using the **Control-it™ 5101** board, ensure it is installed and running correctly (refer to the **5101 Users Manual** for further details).
 - If using a serial port converter, ensure it is running correctly (refer to the appropriate **Users Manual** for further details).
2. Connect the **Control-it™** remote modules as described in the appropriate **Installation Manual**.
3. Use the P5000Demo program as described in the **Installation Manual**, to ensure the correct operation of the system.

Analogue Command Set Summary

Description	Command	Page
Set-Up		
Open a communication channel	openDevice(deviceNum)	
Close a communication channel	closeDevice(deviceNum)	
Identify a module type	moduleType(moduleNum)	
Read a module status register	moduleStatusRegister(moduleNum)	
Set upper scan boundary (serial port only)	setUpperBoundary(lastModule)	
Set lower scan boundary (serial port only)	setLowerBoundary(firstModule)	
Inputs		
Set type	setType(moduleNum, positionNum, type)	
Identify type	aGetType(moduleNum, positionNum)	
Read input	aIn(moduleNum, positionNum)	
Set input averaging	aAverage(moduleNum, positionNum, desiredState)	
New reading available	newReading(moduleNum, positionNum)	
Outputs		
Set output	aOut(moduleNum, outputNum, outputCurrent)	
Errors		
Last error number	getLastErrNum()	
Last error string	getLastErrStr()	
Reset error number	resetErrNum()	
Custom Commands (serial port only)		
Send a custom serial	specialFunction(string _SpecCommand,	

command	_SpecToSend, _SpecToRec, function * _SpecialCommandCallback)	
Multiple Interface Commands		
When more than one High-Speed card or converter is used	Above commands begin with M_, and deviceNum is first parameter, e.g. M_aIn(deviceNum, moduleNum, positionNum)	

Primary Commands

This section describes the primary DLL function calls that relate to the **Control-it™ 5050**.

Each call uses an intuitive name that reflects the operation it performs and is the same for C/C++, Delphi and Visual Basic.

More advanced calls are described in a later section.

Some sample programs are included in the next section and on the disk provided. They will help you understand how to use the driver more quickly.

Glossary of terms used

Module:	A remote module e.g. Control-it™ 5030 .
moduleNum:	The number of the target module from 0 to 30. The module numbers represent the module addresses set by jumpers on the modules themselves. Refer to the Installation Guide in this manual for more detail.
Position:	Each digital module has 16 unique positions. These can either be inputs or outputs. The position numbers range from 0 to 15.
positionNum:	The number of the target position on the remote module from 0 to 15.
Input:	A position on a module that can be read but not written to. Writing to it will produce an error.
Output:	A position on a module that can be written to. Reading it will return the current state that has been requested.

Open a 5101 Board

Description

Provides access to the 5101 board. deviceNum is a two byte, short integer, which is the 5101 board number, as set with the on-board Dip switches, plus 100. Therefore, board number 00 (the factory default setting) would be opened using deviceNum 100, and any extra boards would be 101, 102, etc.

This call must be made prior to all other calls.

Modules Supported

5101

Syntax

```
openDevice(deviceNum);
```

Returns

No Return

See Also

Close a 5101 Board (page 11)

Set a Single Input Type

Description

Sets and stores the input type of each position in non-volatile memory on the module as per Table 7, (refer to **Connecting Inputs & Outputs** in the **Installation Manual**). This must match the Input Type Jumper settings as described in **Jumper Configuration** in the **Installation Manual**.

By default the Input Type is set to 2 (0-10Vdc).

This command can also turn unused positions Off, thereby increasing the module's update rate.

The input type must be set before the input can be read.

Table 7 - Input Types			
Input Type Character	Input Type	Input Type Character	Input Type
0	Channel is Off	4	J 0 - 760°C
1	+/- 1V	5	K 0 - 1000°C
2	+/- 10V	6	T 100-400°C
3	+/- 20mA	7	Return raw count

Modules Supported

5050

Syntax

```
setType(moduleNum, positionNum, type);
```

- moduleNum & positionNum are unsigned short integers.
- type is a character.

Returns

No return.

Read a Single Analogue Input

Description

Reads the state of a single analogue input position.

A position's input type must be set before it can be read. By default this is 0-10Vdc. See Set a Single Input Type (page 9).

Modules Supported

5050

Syntax

aIn(moduleNum, positionNum);

- moduleNum & positionNum are unsigned short integers.

Returns

Returns a short integer (15 bit signed value). Scaling can then be applied as per Table 8 below.

Table 8 - Input Type and Scaling		
Input type	Steps size	Steps / unit
+/- 1V	40uV	25000/Volt
+/- 10V	400uV	2500/Volt
+/- 20mA	1uA	1000/mA
J 0 – 760°C	0.1 °C	10/°C
K 0 - 1000°C	0.1 °C	10/°C
T 100-400°C	0.1 °C	10/°C
Return raw count	A/D converter count. 15bits + sign ¹	

See Also

Set a Single Input Type (page 9)

¹ The Raw Count is not automatically zeroed

Close a 5101 Board

Description

Closes the board and frees the memory space.

This command must be called before terminating the control program to avoid system errors.

Modules Supported

5101

Syntax

```
closeDevice(deviceNum);
```

Returns

No Return.

See Also

Open a 5101 Board (page 8)

Simple Example Programs

A short example program, written in C++, Delphi & Visual Basic, is provided to aid you in learning the commands. The files are on the Technman CD-ROM under \PCSoft\High Speed Examples.

The p5101hal.dll

The p5101hal.dll contains all the commands used by the **Control-it™** modules and must be accessible by your application program. This means either placing it in the same directory as the application or including it in path command in your autoexec.bat.

Inclusions

Each program requires a specific external file to be included for using the .dll calls.

C++

p5101hal.h (on CD-ROM under \PCSoft\High Speed Examples\C++)

p5101hal.lib must also be linked during compiling. (On CD-ROM under \PCSoft\High Speed Examples\C++)

Delphi

p5101hal.pas (on CD-ROM under \PCSoft\High Speed Examples\Delphi)

Visual Basic

p5101hal.bas (on CD-ROM under \PCSoft\High Speed Examples\Visual Basic)

Example

(On CD-ROM under \PCSoft\High Speed Examples\C++\CAnalogue.cpp)

The C++ example is included here as a guide for using the simple commands detailed so far. Their use is identical in Delphi and Visual Basic but the remaining code may differ.

This example repeatedly reads analogue input 0 on a **Control-it™ 5050** at address 0.

```
#include "p5101hal.h"
```

Must be included for function calls.

```
#include <conio>
#include <iostream>
```

```
int main(){
```

```
    open5101Hal();
```

Open access to the 5101 Board.
This **MUST** be done before any other call to the board.

```
    clrscr();
    int moduleNum = 0;
    int positionNum = 0;
```

```
    char Type = 2;
    int scaling = 2500;
```

```
    setType(moduleNum, positionNum, Type);
```

Set the input type to 2 (0-10Vdc).
The input type **MUST** be set before it can be read.

```
    for (int repeat = 0; repeat < 10000; ++repeat)
    {
```

```
        int rawCount = aIn(moduleNum, PositionNum);
```

Use **aIn** to sample the selected position.

```
        float rC = rawCount;
        float volts = rC / scaling;
        cout << "The raw count is " << rawCount;
        cout << "\nwhich is " << volts << " Vdc \r";
    }
```

```
    close5101Hal();
```

Close the 5101 Board.
This **MUST** be done before terminating the program to avoid system errors.

```
}
```

Advanced Commands

These commands provide more flexibility above those detailed in the Primary Commands section.

Each call uses an intuitive name that reflects the operation it performs and is the same for C/C++, Delphi and Visual Basic.

Identify Module Type

Description

Identify the type of module at the given moduleNum.

The module's status register is accessed, returning a code for the I/O type while masking the Last Error and other Status bits.

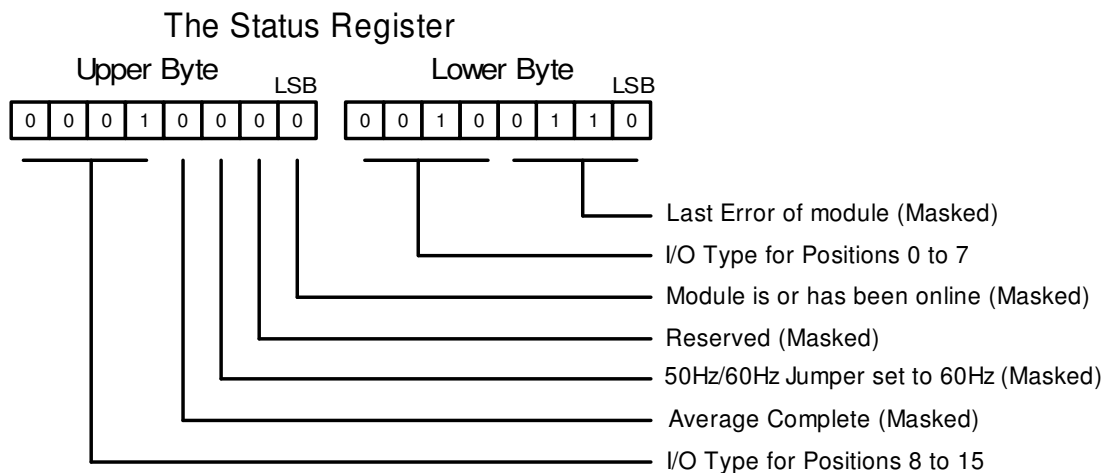


Figure 22 - Layout of the status register.

Modules Supported

5020, 5030, 5040, 5050

Syntax

```
moduleType(moduleNum);
```

- moduleNum is an unsigned short integer.

Returns

Returns a two-byte integer. The upper nibble of each byte identifies the position type.

- The lower byte refers to module positions 0 to 7
- The upper byte refers to module positions 8 to 15, including outputs, if fitted.

The lower nibbles of each byte (the Last Error & Status bits) are masked.

Table 9 details the various returns and their meanings.

Table 9 - Position Type		
Hex Return	Byte	Description
00	Lower	Offline
10 & 20	Either	Digital (refer 5020 manual)
30	Upper	2 Analogue Outputs
40	Either	8 Analogue Inputs
50	Upper	2 Analogue Outputs and 8 Analogue Inputs
60-F0		Reserved

Putting the position-type returns together reveals the module type, as detailed in Table 10.

Table 10 - Module Type			
Hex Return	I/O Configuration	Description	Module Type
0000		No board found	
0040	8 Inputs 0 Outputs	Differential Input module with no Outputs.	5050/8
3040	8 Inputs 2 Outputs	Differential Input module with 2 Outputs.	5050/8/OUT
4040	16 Inputs 0 Outputs	Single-Ended Input module with no Outputs.	5050/16
5040	16 Inputs 2 Outputs	Single-Ended Input module with 2 Outputs.	5050/16/OUT

See Also

Read Module Status Register (page 17)

Read Module Status Register

Description

Same as 'Identify Module Type' but with the lower nibbles unmasked, which contain the remote module's Last Errors and Status.

Modules Supported

5020, 5030, 5040, 5050

Syntax

```
moduleStatusRegister(moduleNum);
```

- moduleNum is an unsigned short integer.

Returns

Returns a two-byte integer as per 'Identify Module Type'.

The upper nibble of each byte contains the I/O type, as per 'Identify Module Type'.

The lower nibble of the LOWER byte contains the Last Error as per Table 11.

Table 11 - Status Register Lower Byte.	
Hex Byte	Last Error
x0	No error
x2	Checksum error for data arriving at PC
x6	Communication time-out error

The lower nibble of the UPPER byte contains module status information as per Table 12.

Table 12 - Status Register Upper Byte.		
Bit	State	Explanation
0	0	Module is offline.
	1	Module is, or has been, online.
1	-	Reserved.
2	0	50/60Hz Noise rejection jumper set to 50Hz .
	1	50/60Hz Noise rejection jumper set to 60Hz .
3	0	Average incomplete.
	1	Average complete.

See Also

Identify Module Type (page 14)

Set Input Averaging

Description

Turn averaging on or off for a single position.

When the AVERAGE ON bit is set, the 5050 reads the same input and averages over 10 samples. The average is built up over the 10 samples and can be read at any time. The formula below is used to compute average.

$$\text{AVERAGE} = (((N-1) * \text{OLD AVERAGE}) + \text{NEW READING}) / N$$

As the 5050 reads samples from its input, N increments from 1 to 10. When it reaches 10 it remains at this value. AVERAGE continues to be computed when each sample is taken.

After averaging is turned on N has the following values 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, ...

Modules Supported

5050

Syntax

aAverage(moduleNum, positionNum, desiredState);

- moduleNum & positionNum are unsigned short integers.
- desiredState can be either On, True or 1 for ON, or Off, False or 0 for OFF.

VEE Users:

There is a VEE version of this command. The syntax is

VEE_aAverage(moduleNum, positionNum, desiredState);

This command accepts desiredState in the form of a short integer, and as above 0 is no average and 1 is average. The functionality is otherwise identical.

Returns

No return.

See Also

Read a Single Analogue Input (page 10).

New Reading Available

Description

Allow checking to see if a new reading is available for an analogue position.

Ready is set to False after aIn is called, and only becomes True after a new sample is available.

This command is used to eliminate repeated aIn calls to the same sample.

Modules Supported

5050

Syntax

```
newReading(moduleNum, positionNum);
```

- moduleNum & positionNum are unsigned short integers.

Returns

Returns a Boolean.

False (0) = no new reading is available since the last aIn call.

True (1) = a new reading is available.

See Also

Read a Single Analogue Input (page 10)

Set a Single Analogue Output

Description

If the 5050 has two optional 4-20mA outputs fitted, this command sets their output current.

The outputs are located at positions (outputNum) 14 & 15. Position numbers 0 & 1 may also be used for 14 & 15 respectively.

NB: On 16 input (single ended) modules, these positions do not clash with input positions 14 & 15. aOut will write to the outputs while aIn will read from the inputs.

The output current (outputCurrent) is set in μ Amps as per Table 13.

Table 13 - Analogue Output Range		
Range	outputCurrent	Actual Current
Minimum	3500	3.5mA
Step	1	1 μ A
Maximum	20000	20mA

NB: The outputs are electrically isolated and therefore require the 4-20mA supply for power. The outputs cannot be set without it.

Modules Supported

5050

Syntax

aOut(moduleNum, outputNum, outputCurrent);

- moduleNum, outputNum & outputCurrent are unsigned short integers.

Returns

No returns.

Last Error Number

Description

Returns an integer representing the last error number encountered. If 0, there is no error.

The error number is cleared by a call to `getLastErrStr()` or `resetErrNum()`.

Modules Supported

5020, 5030, 5040, 5050

Syntax

```
getLastErrNum();
```

Returns

Refer to Error Codes (page 28) for an explanation of error returns.

See Also

Last Error String (page 24)

Reset Error Number (page 25)

Last Error String

Description

Returns a string representing the last error number encountered. Use this call to display an error message to the user.

This call clears the error stored in Last Error Number.

Modules Supported

5020, 5030, 5040, 5050

Syntax

```
getLastErrStr();
```

Returns

Refer to Error Codes (page 28) for an explanation of error returns.

See Also

Last Error Number (page 23)

Reset Error Number (page 25)

Reset Error Number

Description

Clears the error number currently stored in Last Error Number. Use this call before performing a function you wish to error check, if you have not already used `getLastErrStr()`.

Modules Supported

5020, 5030, 5040, 5050

Syntax

```
resetErrNum();
```

See Also

Last Error String (page 24)

Advanced Example Programs

These examples are provided on CD to illustrate the use of all the commands for the **Control-it™ 5101**.

NB Some of the commands are for the **Control-it™ 5020, 30 & 40** Digital modules. For an explanation of these commands, refer to the **Control-it™ 5020** User Manual.

C/C++ Advanced Example

n:\PCSoft\5101\Cadvanced.exe

Visual Basic Advanced Example

n:\PCSoft\5101\VBadvanced.exe

Delphi Advanced Example

n:\PCSoft\5101\DELadvanced.exe

where n is the letter of your CD drive.

Commands for Multiple 5101 Boards

If you have more than one **Control-it™ 5101** fitted to a PC, the commands detailed in the previous chapters can be altered to address them.

The 5101 identification number ranges from 0 to 3 and is set as detailed in the **Control-it™ 5101** Manual.

All commands are modified by the addition of the prefix: **m_** and the 5101 board number as the first argument.

Example: `aIn(moduleNum, positionNum);`

becomes

`m_aIn(boardNum, moduleNum, positionNum);`

- boardNum is a single Byte character and must be 0, 1, 2 or 3.

Each board must also be opened and closed individually using:

`m_open5101(boardNum); &`

`m_close5101(boardNum);`

In the case of board number zero, the original, single board commands can still be used i.e.

`m_aIn(0, 3, 5);`

is the same as

`aIn(3, 5);`

Error Codes

This chapter describes the various error codes encountered when calling the **Control-it™ 5101** and remote modules.

Every call to the DLL will set an error code that can be viewed using the 'Last Error Number' command from the previous chapter. In most cases this will be 0 (No error).

General Errors

No Error

Number: 0

String: Nil

No error was encountered when the call was executed.

Illegal Module Number

Number: 1

String: "Illegal board number specified. Must be between 0 and 30 inclusive."

The program has attempted to call a module number that is out of range.

- moduleNum must be between 0 and 30.

Illegal Position Number

Number: 2

String: "Illegal position number. Must be between 0 and 15 inclusive."

The program has attempted to call a position number that is out of range.

- positionNum must be between 0 and 15.

Module Not Present

Number: 3

String: "Attempted to access a module that is not present."

The program has attempted to call a module that is not present.

- Check the address jumper settings on the target module (refer Installation Guide in this manual). moduleNum must equal this setting.

Analogue Module Errors

No Analogue Output Fitted

Number: 20

String: "Attempted to write to an input."

An attempt was made to write to a module without an analogue output position.

- Check that you are writing to the correct analogue module.
- Check that you have purchased module with analogue outputs fitted.

Not an Analogue Module

Number: 21

String: "Attempted digital call to non-digital
 module."

A module was found at this address, but it is not an analogue module so calling analogue routines is not possible.

- Check the address jumper settings on the target module (refer Installation Guide in this manual). moduleNum must equal this setting.

Critical Warnings

WinDriver not Found

Number: 40

String: "Could not find WinDriver on the system"

The WinDriver for the **Control-it™ 5101** was not found.

- Check that windrvr.vxd is in the windows\system\vm32 directory for Win 95 & 98, or windrvr.sys is in the windows\system32\drivers directory for Win 2000 & NT.
- If necessary, re-install driver software from TEL CD-ROM.

Control-it™ 5101 Board not Found

Number: 41

String: "No 5101 High-Speed cards detected."

There were no PCI cards found with DevId & VendorID matching 5101.

- Ensure the card is properly fitted in the PCI slot.
- Install the card in another empty PCI slot.
- Use 'Add New Hardware' in the Control Panel to install manually.

Non-functional Control-it™ 5101 Board Found

Number: 42

String: "No functional 5101 High-Speed cards detected."

A card was found, but it was not a functional 5101. This error will occur if the last application did not close the board correctly.

- Restart the computer
- Ensure close5101Hal is called at the end of the application.

Null Handle Used

Number: 43

String: "Call made with a null handle (i.e. a closed board) "

A call was made to a board that has not opened.

- Ensure that open5101(); is the first call made to the board.
- Use getLastErrNum immediately after open5101(); to verify that the board has opened correctly. The return value must be 0.

Invalid handle Used

Number: 44

String: "Call made with an invalid handle (i.e. not between 0 - 3) "

A call was made to a board using a handle other than 0, 1, 2 or 3.

- Check that code uses the correct 5101 handle.

Wrong version of Windriver

Number: 45

Strings: "Windriver was found, but it was the
wrong version for this DLL"

Windriver version on the system is older than the expected version.
Run the set up installshield from the Technman CD-ROM.