PCI90 I2C Communications Adapter

Issue 1.1
06/03/2007

Welcome to the Calibre PCI90 fixed voltage I²C adapter. This adapter provides full I²C bi-directional compatibility as either a master or slave from within a Windows 95/98 or Windows NT or Windows 2000 or Windows XP environment.

If you have any queries relating to this or any other I²C product supplied by Calibre please visit our web site www.calibreuk.com***.***

For technical support please e-mail techsupport@calibreuk.com ***or send your queries by fax to (44) 1274 730960, for the attention of our I²C Technical Support Department.***

COPYRIGHT

Copyright (c) 2007 Calibre UK Ltd
Cornwall House, Cornwall Terrace
Bradford, BD8 7JS. UK.
E-mail: sales@calibreuk.com
Web site www.calibreuk.com
All World-wide Rights Reserved

Issue 1.1 06/03/2007

Calibre operates a policy of continued product improvement, therefore specifications are subject to change without notice as products are updated or revised.

E&OE.

CALIBRE

Contents

CALIBRE

INTRODUCTION

1.1. General Introduction

The I^2C Communications Adapter is a PCI bus PC interface card designed to fit any IBM PC compatible. Based on the Philips PCF8584 bus controller, it features full I^2C bi-directional compatibility as either a master or slave. I^2C connections are made via a 9 way "D" socket. This product complies with the requirements of EEC Directive 89/336 for EMC and is CE marked.

1.2. Packing List

Your I^2C Communications Adapter is supplied with the following items:-

A. I^2C CD ROM
B. The PCI90 plug-in card (the actual adapter)

1.3. Configuring the Adapter

NOTE
MANY COMPONENTS ON THE ADAPTER CARD ARE STATIC SENSITIVE. OBSERVE NORMAL STATIC SENSITIVE PRECAUTIONS WHEN HANDLING THE CARD!

The adapter is supplied in a standard configuration which should suit most applications. However, some features are link selectable. Read the following section to change the configuration.

1.4. Bus Termination and Protection

In 2006 the PCI I2C adapters were redesigned so that they complied with the RoSH Directive PCBs earlier than 2.1 are not RoHS compliant version 2.1 and later are. At this time improvements were made to the manufacturability of the product and to the output protection. None of the changes made to the I2C adapters in anyway functionally alters the product, but the jumper settings are different this section details both settings please check the PCB to determine which version of PCB you have. If you are unsure please contact you sales representative.

PCB Issue up to and including 2.0

Normally the system to which the I^2C Communications Adapter is to be connected should already have master pull up resistors fitted to the SCL and SDA lines. If this is not the case, LK10 and LK12 can be used to connect 4K7 pull up resistors to the 5V supply on these lines. The standard configuration is with these resistors disconnected.

The SCL and SDA lines are protected by 100R series resistors before exiting the adapter via the 9 way "D" socket. Upstream of the series resistors, the SCL and SDA lines are pulled up with high value resistors (10K). These resistors can be linked out via links 9 and 11 although to ensure that the I^2C Bus is in a defined state even if no other devices are connected the links should be left on. This is the standard configuration.

LK13 and LK14 connect optional protection diodes to the SCL and SDL lines. When selected, these lines are clamped to the 0V and +5V lines giving protection against transients. If these diodes are connected, the external I^2C system will not function if the adapter is connected but not powered up. The standard configuration is with these diodes disconnected.

PCB Issue 2.1 and later

Normally the system to which the I^2C Communications Adapter is to be connected should already have master pull up resistors fitted to the SCL and SDA lines. If this is not the case, LK5 and LK6 can be used to connect 4K7 pull up resistors to the 5V supply on these lines. The standard configuration is with these resistors disconnected.

The SCL and SDA lines are protected by 100R series resistors before exiting the adapter via the 9 way "D" socket. Upstream of the series resistors, the SCL and SDA lines are pulled up with high value resistors (10K). These resistors can be linked out via links 3 and 4 although to ensure that the I^2C Bus is in a defined state even if no other devices are connected the links should be left on. This is the standard configuration.

1.5. Connecting the Adapter to your System

All external connections are made via a 9 way “D” socket:

Pin	Normal Mode
1	0V
2	0V
3	0V
4	0V
5	0V
6	SDA (Bi-directional)
7	+5V
8	SCL (Bi-directional)
9	NC

1.6. Bus Capacitance Limitations/Cable Choice

The maximum allowable capacitance on the I^2C bus in normal mode depends on the value of the SCL and SDA master pull-ups, but never exceeds 400pF. Refer to Phillips Technical Handbook Book 4 Parts 12a and 12b for further details. Care should be taken in choosing a length and type of interconnecting cable, which will not exceed this limit.

For most systems with a distance of a few metres between the I^2C Communications Adapter and the target system, screened cable is NOT recommended, as it is likely to introduce too much capacitance. However, the EMC performance of an unscreened cable is always potentially poorer than a screened one. The Adapter's EMC performance even with an unscreened cable is good - but this may not be true of the target system! If you are in any doubt at to the best way to connect up your system with EMC in mind please contact your supplier or Calibre for advice.

1.7. +5V Power Supply

Pin 7 on the “D” connector is connected directly to the PC +5V power rail. Power for external circuitry can be drawn from here, but care should be taken never to short it to 0V or to exceed 250mA loading. It is short circuit and overload protected by a self-resetting thermal fuse but prolonged shorting could cause the ICA90 to generate an excessive amount of heat inside your computer.

CALIBRE

INSTALLING THE ADAPTER UNDER WINDOWS 9x

2.1. Introduction

This section details the installation of the PCI90 I^2C communications adapter under Windows 95® / 98®.

The appearance of the dialog boxes during the installation of new hardware varies depending on the version of Windows 95 and Windows 98. Consequently this section is slit into 3 parts

Windows 95 versions 950 and 950a
Windows 95 version 950b and 950c
Windows 98

Which version of Windows you have installed on your machine by click on the **My Computer** icon with the right mouse button and then selecting **Properties**.

2.2. Installing the Adapter

Turn off your computer and disconnect it from the mains power supply. Remove the PC cover and then a PCI slot blanking plate.

Insert the card into the PCI slot, ensure that it is fully home and screw the adapter panel into the PC.

Replace the PC cover and reconnect the PC to the mains power supply. Turn the PC on.

2.3. Windows 95 Versions 950 and 950a

When Windows 95 restarts it should detect the new hardware (if this is not so see section 2.6), select

"Driver from disk provided by hardware manufacturer" then click **OK**

When you see the **Install from Disk** screen, insert the CD_ROM into the drive. Select the drive letter appropriate to your CD – ROM drive (usually D) browse the CD for the \cd_pci directory. The click **OK**, Windows will install the drivers for the adapter.

NOW SHUTDOWN YOUR PC

On restarting you PC click on the **My Computer** icon with the right mouse button and then selecting **Properties**. Select **Device Manager**, expand the **CaDrv** and the select **I2C Driver for PCI Devices** Click on **Properties** is the installation was successful the **Device Status** will say "**This device is working properly**".

2.4. Windows 95 Versions 950b and 950c

When Windows 95 restarts it should detect the new hardware (if this is not so see section 2.6), and the Driver Wizard should be displayed.

Insert the CD_ROM into the drive. Follow the wizard instructions, if the wizard fails to find the driver select **Other Location**. Then follow the wizard instructions to install the unknown device from the CD ROM.

NOW SHUTDOWN YOUR PC

On restarting you PC click on the **My Computer** icon with the right mouse button and then selecting **Properties**. Select **Device Manager**, expand the **CaDrv** and the select **I2C Driver for PCI Devices** Click on **Properties** is the installation was successful the **Device Status** will say "**This device is working properly**".

2.5. Windows 98

When Windows 98 restarts it should detect the new hardware (if this is not so see section 2.6), and the Driver Wizard should be displayed.

Select Search for the best driver for your device and click **Next.**

When the next screen is displayed select **CD-ROM drive** and **Specify a location**. Browse the CD for the \cd_pci directory, select the CaDrv.inf file and click **Next**.

The next screen tells you that Windows has found the driver click **Next**. On the final screen click **Finish**.

NOW SHUTDOWN YOUR PC

On restarting you PC click on the **My Computer** icon with the right mouse button and then selecting **Properties**. Select **Device Manager**, expand the **CaDrv** and the select **I2C Driver for PCI Devices** Click on **Properties** is the installation was successful the **Device Status** will say "**This device is working properly**".

2.6. Plug and Play Does Not Find Hardware

From the Control Panel select **Add New Hardware**. Follow the instruction and when prompted choose **Other Devices** and **Have Disk**. Browse the CD for the \cd_pci directory select the Cadrv.inf then continue to follow the displayed instruction.

When the installation is complete **SHUTDOWN THE PC.**

On restarting you PC click on the **My Computer** icon with the right mouse button and then selecting **Properties**. Select **Device Manager**, expand the **CaDrv** and the select **I2C Driver for PCI Devices** Click on **Properties** is the installation was successful the **Device Status** will say "**This device is working properly**".

2.7. Windows 9x File Location

If you are going to use the dynamic link library to write your I2C protocol software click on the PCI_UTILS follow the instructions to unzip the files (if you do not have an unlock code please contact sales@calibreuk.com). Then

1)Copy the Cali2c32.Lib from the \lib directory into your compilers \lib

2)Copy the cali2c32.Dll from the \lib directory into \WINDOWS\SYSTEM

INSTALLING THE ADAPTER UNDER WINDOWS NT4

3.1. Introduction

This section details the installation of the PCI90 I^2C communications adapter under Windows NT4.

3.2. Installing the Adapter

Turn off your computer and disconnect it from the mains power supply. Remove the PC cover and then a PCI slot blanking plate.

Insert the card into the PCI slot, ensure that it is fully home and screw the adapter panel into the PC.

Replace the PC cover and reconnect the PC to the mains power supply. Turn the PC on.

3.3. Installing the Device Driver

TO INSTALL A DRIVER YOU MUST HAVE ADMINISTRATOR PRIVILAGES.

Copy all the following files from the CD (located in the \cd_pci directory, to c:\winnt\system32\drivers

1) CaDrv.sys - The device driver
2) CaDrv.ini - The device driver initialisation file
3) Regini.exe - This a Microsoft registration program

From a DOS window, change the directory to the \system32\drivers (usually c:\winnt\system32\drivers). Then type **regini cadrv.ini <CR>** this will register the driver.

NOW SHUTDOWN YOUR PC

3.4. WinNT4 File Location

If you are going to use the dynamic link library to write your I2C protocol software click on the PCI_UTILS follow the instructions to unzip the files (if you do not have an unlock code please contact sales@calibreuk.com). Then

1)Copy the Cali2c32.Lib from the \lib directory into your compilers \lib

2)Copy the cali2c32.Dll from the \lib directory into \WINNT\SYSTEM

INSTALLING THE ADAPTER UNDER WINDOWS 2000 and XP

4.1. Introduction

This section details the installation of the PCI90 I^2C communications adapter under Windows 2000 and XP. The drivers for these operating systems are located in the \CD_PCI\W2k Drivers and \CD_PCI\XP DRIVERS folders on the CD. You MUST use the driver appropriate to you operating system DO NOT try to us the Windows NT or Windows 98 drivers these will not work.

Both these operating systems are plug and play you must NOT run the regini applications on these operating systems.

4.2. Installing the Adapter

Turn off your computer and disconnect it from the mains power supply. Remove the PC cover and then a PCI slot blanking plate.

Insert the card into the PCI slot, ensure that it is fully home and screw the adapter panel into the PC.

Replace the PC cover and reconnect the PC to the mains power supply. Turn the PC on.

4.3. Installing the Device Driver

TO INSTALL A DRIVER YOU MUST BE THE ADMINISTRATOR.

Wait for the wizard to find the new hardware, if the operating system does not recognise new hardware select add new hardware via the control panel.

VERY IMPORTANT: Sometimes the wizard selects an unsuitable driver option found a c:\windows\inf\oem1.inf this file is nothing to do with Calibre and MUST NOT be used.

The solution is to select the "Display list" option. From the next page select Have Disk select the CAWDM13.INF file located on the CD in \CD_PCI

5.1. Introduction

Each utility is documented in a standard format which lists its name, usage, function and effect on the adapter is given. The adapter should be setup prior to any data transfer.

5.2. Function Prototypes

If you are using 'C' or 'C++' copy the file CALI2C32.H into the directory containing your project and add the line:

```
#include "CALI2C32.H"
```

The following functions are implemented in the windows libraries:-

```
extern __declspec(dllimport) int WINAPI setup (int, int, int);
extern __declspec(dllimport) int WINAPI sendaddress (int, int, int);
extern __declspec(dllimport) int WINAPI writebyte(int, int);
extern __declspec(dllimport) int WINAPI readbyte(int, int);
extern __declspec(dllimport) int WINAPI sendstop(int);
extern __declspec(dllimport) int WINAPI restart (int, int, int);
extern __declspec(dllimport) int WINAPI getstatus(int);
extern __declspec(dllimport) int WINAPI recover(int);
extern __declspec(dllimport) int WINAPI slavelastbyte(int);
extern __declspec(dllimport) int WINAPI dllissue(void);
```

If you are using Visual Basic copy the file CALI2C32.BAS into the directory containing your project and add the file CALI2C32.BAS to your project:

The following functions are implemented in the windows libraries:-

```
Public Declare Function setup% Lib "cali2c32.dll" (ByVal Board%, ByVal
OwnAddress%, ByVal Sclk%)
Public Declare Function sendaddress% Lib "cali2c32.dll" (ByVal Board%, ByVal
slaveaddress%, ByVal Setnack%)
Public Declare Function restart% Lib "cali2c32.dll" (ByVal Board%, ByVal
slaveaddress%, ByVal Setnack%)
Public Declare Function writebyte% Lib "cali2c32" (ByVal Board%, ByVal
wrdata%)
Public Declare Function readbyte% Lib "cali2c32.dll" (ByVal Board%, ByVal
Setnack%)
Public Declare Function sendstop% Lib "cali2c32.dll" (ByVal Board%)
Public Declare Function getstatus% Lib "cali2c32.dll" (ByVal Board%)
Public Declare Function recover% Lib "cali2c32.dll" (ByVal Board%)
Public Declare Function slavelastbyte% Lib "cali2c32.dll" (ByVal Board%)
Public Declare Function dllversion% Lib "cali2c32.dll" ()
```

NOTE A type is defined in cali2c32.bas to help passing parameters to the DLL, if you do not wish to use this local variables MUST be declared as static

5.3. Function Description

5.3.1. setup

Function specification int setup(int board, int ownaddress, int sclk)

Parameters are:

int board

This parameter .is reserved for future use and should always be set to zero (0).

int ownaddress

This is the I2C address to which the adapter is to respond in slave mode. This forms the upper 7 bits of the 8 bit address, the lowest bit being the read(1) or write(0) bit. This means that if ownaddress = 57H the card will respond to a write address of AEH and a read address of AFH.

int sclk

This is the clock rate (bit rate for the I^2C serial bus) when operating as a master.

Value of sclk	Approximate SCL-kHz
0	90
1	45
2	11
3	1.5

Parameters returned: If the software fails to find the driver error code 9000H is returned otherwise the status is returned.

Prerequisites: None.

Functional description: This function characterises the PC and initialises adapter ready for I2C transfers.

5.3.2. sendaddress

Function specification: Int sendaddress(int board, int slaveaddress, int setnack)

Parameters are:

int board

This parameter .is reserved for future use and should always be set to zero (0).

int slaveaddress

This is the address to be accessed via the I2C, e.g. A0H.

int setnack

This controls whether the adapter transmits an acknowledge down the I2C bus on reception of a byte. The last byte received during a transfer must not be acknowledged, in all other cases acknowledge must be enabled. If setnack = 0 then acknowledge is enabled, if setnack = 1 then acknowledge is disabled. Therefore, if a read (odd numbered) address is being sent AND only 1 Byte is to be read, setnack should be set to = 1; in all other cases it must be clear = 0.

Parameters returned: ***int ErrCode***.

If the transfer time out occurs error code 8001H is returned otherwise the status is returned.

Prerequisites: The adapter must be configured by running **setup**.

Functional description: The function waits for the bus to be free. Then sends the slave address with the appropriate acknowledge.

The acknowledge is set ready for the data transfer after the address and hence in read mode (odd address being sent) if only one byte is to be read the setnack parameter must equal 1. If more than one byte is to be read or if in write mode (even address being sent) then setnack must equal 0.

The function waits for the address to be sent. Should a time-out occur during the sending of an address then an error code 8001H is returned, otherwise the status is returned.

5.3.3. writebyte

Function specification: Int writebyte(int board, int wrData)

Parameters are:

int board

This parameter .is reserved for future use and should always be set to zero (0).

int wrData

This is the byte of data to be written.

Parameters returned: ***int ErrCode***.

If the transfer time out occurs error code 8004H is returned otherwise the status is returned.

Prerequisites: Adapter must be configured using **setup**, start and write address sent by **sendaddress**.

Functional description: The function writes the data to the adapter and then waits for it to be sent. Should a time-out occur during the sending of the data then error code 8004H is returned, otherwise the status is returned.

Writebyte is compatible with both master write and slave write modes.

5.3.4. readbyte

Function specification: Int readbyte(int board, int setnack)

Parameters are:

int board

This parameter .is reserved for future use and should always be set to zero (0).

int setnack

This controls whether the adapter transmits an acknowledge down the I2C bus on reception of a byte. The last byte received during a transfer must not be acknowledged, in all other cases acknowledge must be enabled. If setnack = 0 then acknowledge is enabled, if setnack = 1 then acknowledge is disabled. Therefore, if the LAST BUT ONE byte is to be read, setnack should be set to =1; in all other cases it is to be set = 0 (in the case of reading 1 byte only, the acknowledge will have been disabled by sendaddress and so should now be enabled again after reading the data, hence setnack = 0 for reading a single byte of data).

The first read from the adapter following a write to it will result in the data that was written being returned. This data MUST be read and discarded before real data can be read, DO NOT count this extra read when considering whether or not to acknowledge.

Parameters returned ***int I2CData***

If a time-out occurs the ErrCode 8005H is returned, otherwise the data is returned.

Prerequisites Adapter must be configured using **setup**, start and read address sent by **sendaddress**.

Functional description If setnack is 1 the function writes 40H to the control register to establish the correct acknowledge procedure.

The data is read from the adapter.

Readbyte is compatible with both master read and slave read modes.

5.3.5. sendstop

Function specification Int sendstop(int board)

Parameters are: ***int board***

This parameter .is reserved for future use and should always be set to zero (0).

Parameters returned ***int ErrCode***.

If the transfer time out occurs error code 8002H is returned otherwise the status is returned.

Prerequisites Adapter must be configured using **setup.** Should normally only be used at the end of a transmission. Correct acknowledge sequence must have been applied if the transmission was a read.

Functional description Instructs the adapter to send a stop code and wait for it to be sent.

Should a time-out occur during the sending of a stop then an error code 8002H is returned, otherwise the status is returned.

5.3.6. restart

Function specification Int restart(int board, int slaveaddress, int setnack)

Parameters are: ***int board***

This parameter .is reserved for future use and should always be set to zero (0).

int slaveaddress

The address to be accessed via the I2C, e.g. A1H.

int setnack

This controls whether the adapter transmits an acknowledge down the I2C bus on reception of a byte. The last byte received during a transfer must not be acknowledged, in all other cases acknowledge must be enabled. If setnack = 0 then acknowledge is enabled, if setnack = 1 then acknowledge is disabled. Therefore, if a read (odd numbered) address is being sent AND only 1 Byte is to be read, setnack should be set to = 1; in all other cases it must be clear = 0.

Parameters returned ***int ErrCode***

If the transfer time out occurs error code 8003H is returned otherwise the status is returned.

Prerequisites Adapter must be configured using **setup.** A start and slave address must have previously been sent using **sendaddress**.

Usually a data pointer would already have been written using **writebyte.**

Functional description	Sends a start code and the slave address specified and presets the acknowledge status depending on the value of setnack. The acknowledge is set ready for the data transfer after the address and hence in read mode (odd address being sent) if only one byte is to be read the setnack parameter must equal 1. If more than one byte is to be read or if in write mode (even address being sent) then setnack must equal 0. The function waits for the address to be sent. Should a time-out occur during the sending of an address then an error code 8003H is returned, otherwise the status is returned.

5.3.7. getstatus

Function specification*:*	Int getstatus(int board)
Parameters are:	***int board*** This parameter .is reserved for future use and should always be set to zero (0).
Parameters returned	***int I2Cstatus*** The current value of the bus status is returned.
Prerequisites	Adapter must be configured using **setup.**
Functional description	The function reads status word from the adapter and returns it.

5.3.8. recover

Function specification	Int recover(int board)
Parameters are:	***int board*** This parameter .is reserved for future use and should always be set to zero (0).
Parameters returned	***int ErrCode.*** If the bus recovery failed error code 8006H is returned otherwise the status is returned.
Prerequisites	Adapter must be configured using **setup.**
Functional description	This function issues two consecutive stop commands on the bus, with a delay in between. It then clears the adapter registers and reads the status. This should normally set the adapter into a known idle state when a bus error or other problem has occurred. If the status does not indicate bus free or the Bus Error bit is still set then 8006H is returned otherwise the status is returned.

5.3.9. slavelastbyte

Function specification	int slavelastbyte(int board)
Parameters are:	***int board*** This parameter .is reserved for future use and should always be set to zero (0).
Parameters returned	The function always returns 0.
Prerequisites	Adapter must be configured using **setup.** This function would normally only be called following the end of a transmission in slave write mode - when the adapter is being read as a slave, by another master, *not when writing to a slave using the adapter.*
Functional description	This function is used when the adapter is a slave being read by a master elsewhere on the bus - the adapter is in slave write mode. The function must be called immediately after the master indicates the last byte has been read (by not acknowledging that byte). This function is required to clear the I^2C data lines so that the master can send a stop signal.

The Real-Time Bus Monitor

Before attempting to run the monitor program ensure that the device drivers are installed correctly in accordance with this manual.

The program WINMONITOR.EXE runs in a DOS window and is a completely non-invasive real-time bus monitor which records activity on an I^2C-bus, post-processes the data and stores the results in an ASCII file. This file can be printed out or examined with a word-processor as required.

During the monitoring of the I2C bus the program may miss part of a transfer this is due to the operating system servicing other hardware and / or applications. To minimise the amount of data missed we recommend that you close as many windows applications as possible especially all windows messaging applications, e-mail and FindFast. We also recommend that you run the application from a PII 350MHz PC (or better), note the application will run on a lesser specification PC but you are likely to miss more data.

The following mnemonics are used by the monitor program in the ASCII file which it produces.

SaXX Start code followed by address, acknowledged.

SnXX Start code followed by address, not acknowledged.

DaXX Data byte, acknowledged.

DnXX Data byte, not acknowledged.

STOP Stop code.

BUS ERROR A bus error has occurred. This is non-fatal to the monitor but a small amount of data may have been lost whilst recovery was taking place.

Appendix A I²C Communications Adapter Status Codes

This is an eight bit register, read using the getstatus routine. Each individual bit has its own meaning as follows:

Bit 7 (MSB) - The PIN Bit

The PIN bit "Pending Interrupt Not" is a read-only flag which is used to synchronise serial communication. Each time a serial data transmission is initiated (by sendaddress routine or setting STA bit) the PIN will be set (= 1) automatically. After successful transmission of one byte (9 clock pulses, including acknowledge), this bit will be automatically reset (= 0) indicating a complete transmission. When the ENI bit (enable interrupt) is also set, the PIN triggers an external interrupt via the selected IRQ line when PIN is reset. When in receiver mode, the PIN is also reset on completion of each received byte. In polled applications, the PIN bit is tested (using the getstatus routine) to determine when a serial transmission has been completed. In receiver mode, the PIN bit is tested (using the readbyte function). When the PIN becomes set all other status bits will be reset, with the exception of the BB (not Bus Busy) bit.

In short, when transmitting data, if PIN = 0 then the data has been sent, if PIN = 1 then it has not. When receiving data, if PIN = 0 then there is unread received data ready to read, if PIN = 1 then either the data received has already been read, or no data has yet been received.

Bit 6 - Not Used This bit is not currently used and will always = 0.

Bit 5 - The STS Bit

When in slave-receiver mode (i.e. transmission initiated by a master elsewhere on the I²C bus), the flag STS = 1 when an externally generated Stop condition is detected, otherwise STS = 0. This flag is used only in slave-receiver mode.

Bit 4 - The BER Bit

The BER (Bus Error) bit. BER = 1 when a misplaced Start or Stop has been detected, otherwise BER = 0. This can be quite serious since the I²C devices on the bus may be left in an undefined state after a bus error has occurred - in some circumstances the only way to get the bus going again may be to reset all the I²C devices on it.

Bit 3 - The LRB/ADO Bit

The LRB (Last received Bit) / ADO (Address 0 "General Call" Address Received) bit. This dual function status bit holds the value of the last received bit over the I²C bus when AAS (Bit 2) = 0. Normally this will be the value of the slave acknowledge; thus checking for slave acknowledgement is done via testing of the LRB bit. When AAS (Bit 2) = 1 ("Addressed As Slave"), the I²C Communications Adapter has been addressed as a slave and the ADO bit will = 1 if the slave address received was the "General Call" address. For further information on the "General Call" Address, see the Philips data books referenced in Section 8 of this User Manual.

Bit 2 - The AAS Bit

The AAS ("Addressed As Slave") bit. When acting as a slave-receiver, this flag is set = 1 when an incoming address over the I²C bus matches the value defined by the setup routine, or if the slave address received was the I2C bus "General Call" address (00 Hex). In all other circumstances, AAS = 0.

Bit 1 - The LAB Bit

The LAB (Lost Arbitration) bit. This bit is set = 1 when, in multimaster operation (more than one master present on the I²C bus) arbitration is lost to another master on the I²C bus. In all other circumstances, LAB = 0.

Bit 0 - The BB Bit

The BB (not Bus Busy) bit. This is a read-only flag indicating when the I²C bus is in use. BB = 0 indicates that the bus is busy, and access is not possible (unless of course it is busy because the I²C Communications Adapter itself has control of the bus). This bit is set = 1 by Stop conditions and reset = 0 by Start conditions. In short, BB = 1 means that the bus is free and a new transmission can be started.

CALIBRE

THE MOST COMMONLY ASKED I2C QUESTIONS

General Questions

Question **Will my adapter run I2C clock speeds greater than 90KHz?**
Answer At the moment your adapter is limited by the Bus Controller chip fitted, to a maximum of 90KHz as a master and 100KHz as a slave.

Question **I get corrupted transfers why is this?**
Answer The most likely reason for corrupted transfers is either incorrect bus termination or excessive capacitance - see the manual for details.

Question **Do you have software to talk to my.....?**
Answer Unfortunately there are too many I2C devices for us to be able to offer complete solutions - although we can supply a windows based application called WINI2C which is designed for those just starting I2C or wishing to perform simple I2C tasks, please contact our sales team or look on our web site, www.calibreuk.com for further information.

Question **I am trying to read from a device, the first time my software works fine but when I try again I can't get anything what's wrong?**
Answer Please check that you are changing the value of Setnack in accordance with the manual, it is likely that you have not made Setnack 1 for the last **AND** last but one bytes being read.

Windows Questions

Question **My software cannot find the adapter. Your Windows software reports that it cannot configure the adapter. Why is this?**
Answer Have you registered the device driver?.

Question **I think I have registered the driver how can I find out if I have?**
Answer You need to inspect the registry as follows

Windows 95 START - Run regedit
HKEY_LOCAL_MACHINE
|
|--SYSTEM
|
|--CurrentControlSet
|
|--Services
|
|--Class
|
|--Cadrv
|
|--0000

Windows NT START - Run regedit
HKEY_LOCAL_MACHINE
|
|--SYSTEM
|
|--CurrentControlSet
|
|--Services
|
|--CaDrv
|
|--Eum
|--Parameters

Question **I have read the manual and still cannot get the communications to run. What do I do next?**

Answer Check that you have fully implemented the protocol between the adapter and the other I2C devices see the device manufacturers data sheet for details.

Check that the software you have written is logically and syntactically correct - this is probably the most common cause of software faults we have to deal with.

Send us the following details:-

1)The link settings of the adapter.
2)A sketch of the relevant I2C hardware including the location of bus termination.
3)The type and speed of processor within your PC and which operating system, you are running.
4)Brief software listings, or which Calibre software you are running.
5)The serial number of your I2C adapter, or when you purchased it.

PLEASE EMAIL YOUR QUERY TO: techsupport@calibreuk.com
OR FAX YOUR QUERY TO: 44-1274-730960

We will endeavour to help you.