XLine Quick Start Guide

Document No. 80-18392   Issue 9

Current Issue :-   Issue 9, 23 October 2014
Previous Issues :-   Issue 8, 8th December 2012
                        Issue 7, 4th November 2011

Document No 80-18392-9

If your query has not been answered by this Manual, or you require further information, please email Heber Customer Support: support@heber.co.uk

The latest version of this User Manual and other technical information can be found on the Heber website: www.heber.co.uk

The information contained in this User Manual is believed to be accurate and reliable. However, Heber Ltd. assumes no responsibility for its use, and reserves the right to revise the documentation without notice.

Precise specifications may change without prior notice.

All trademarks are acknowledged.

HEBER LIMITED
Belvedere Mill
Chalford Stroud
GL6 8NT
UK

Tel +44 (0) 1453 886000
Fax +44 (0) 1453 885013
Email   support@heber.co.uk
Website   www.heber.co.uk

CONTENTS

LIST OF TABLES

This page is intentionally left blank.

1   SCOPE OF THIS MANUAL

This manual acts as both a Hardware Reference Manual and Software Installation Guide for boards in the Heber XLine family of USB control boards.

The manual is intended to cover both 32-bit and 64-bit installations on both Windows and Linux. Where some aspect of the manual is specific to 64-bit versions, these will be noted within square brackets, thus [.....]. Files specific to 64-bit versions have '_x64' added to the equivalent 32-bit file name.

The XLine family consists of three different control boards, all of which are supported by this manual. These are:
- X10 board – a slower predecessor of the X10i board, now deprecated.
- X10i – a USB-based high speed general purpose IO control interface.
- X15 – an offshoot from the X10i, with greatly enhanced security and other facilities.

Nucleus N500 and N1000 PC System boards embed identical X10i hardware to that found on the XLine peripheral board as part of the Nucleus System board itself. As a result any references to X10i in this manual apply equally to the X10i part of the Nucleus family members.

A common set of Development Kit Software and reference manuals support these different products and this manual is one component part of that set. Where differences exist between X10/X10i/X15 products, these differences are highlighted at the relevant points throughout the manual.

The second chapter describes the Software Installation process for the Development Kit software components, demonstration programs and device drivers, for both Windows and Linux operating systems.

Then follows a chapter describing the collection of Demonstration Programs that illustrate the simplicity of writing software for the XLine family of boards and demonstrates the capabilities of the products.

This manual ends with a Hardware Reference section, in which the layouts of the boards themselves and of the wiring connections of all connectors are described.

NOTE:
From October 2014 the XLine support software has been enhanced so that either 32-bit or 64-bit XLine applications can be run on 64-bit (Windows or Linux) platforms. The installation of a 64-bit development kit now automatically installs both the 32-bit and 64-bit development kit components. To help avoid confusion, 64-bit related files have '_x64' added to the filenames. The Linux 'file' utility and its Windows equivalent can also be used to identify whether an executable files or DLL/.so file is compiled for a 32-bit or 64-bit platform.

Downloaded 32-bit development kits continue to operate, as previously, on 32-bit (Windows or Linux) platforms only.

2   INSTALLING THE DEVELOPMENT KIT

This chapter describes the installation of the XLine Development Kit software components, to begin with for members of the Windows operating system family (Windows 2000, Windows XP, Windows Vista and Windows 7) and later in the chapter describes the installation process under the Linux operating system.

Whether Windows or Linux is the choice of operating system, a common directory structure is used for both Windows and Linux versions of the development kit. By default (which can be overridden at installation time) the development kit will be installed under "C:\Heber\Xline" for Windows and under "/usr/src" is suggested under Linux. The directory structure created beneath the installation directory consists of:

- **docs**: All XLine documentation is located within this directory. The documents are saved in Adobe PDF format; the reader can be downloaded from http://www.adobe.com if required.
- **driver**: The device driver components required to drive the XLine boards are located in this directory; for Windows this includes .SYS and .DLL components, while for Linux it contains the .c source file components of the kernel driver.
- **include**: Contains the API header files of which 'fflyusb.h' and 'unlockio.h' are the most important.
- **lib**: Contains the API library file fflyusb.lib and others in the case of Windows, or library fflyusb.so and others in the case of Linux. XLine programs you create must be linked against these libraries.
- **samples**: Contains various C++ demonstration programs along with the corresponding source code. These can be compiler either for Windows or Linux.
- **samplesc**: Programs in this directory demonstrate how to write application programs for the c-language, which by implication shows the mechanisms for using XLine products under Visual Basic, c#, etc.
- **bin**: This directory is only included in the Windows installation structure and contains the XLine Windows Diagnostic Program 'x10diag'.

2.1   Installing under Windows 2000, XP, Vista and Win7

2.1.1   Installation
The most up-to-date versions of development kits can be obtained from:
    http://www.heber.co.uk/support/downloads

The installation process under all versions of Windows is very similar and differs only in the way Windows recognizes and loads the XLine device drivers at the end of the installation process.

The installation process is described as a "software-first installation" where the software installation is run BEFORE the XLine hardware is attached, and only when the software is fully installed is XLine hardware connected for the first time.

It is therefore appropriate to disconnect all XLine devices already connected by USB cables BEFORE launching the software installation process.

This installation process has four phases:
- User decides where Development Kit is installed and where Heber menus are added to Start Menus.
- Library, driver, demonstration samples and documentation files are copied to folder chosen by User.
- The Device Driver components are installed into the Windows operating system software.
- XLine hardware is plugged in, detected and association between device driver and XLine hardware is confirmed.

The installation is performed by running the "xline_setup.exe" ["xline_setup_x64.exe"] program (eg. by double-clicking on xline_setup.exe). For the majority of installations, simply follow the on-screen instructions and, if appropriate, accept the default values for installation directory, menu, etc.

Note that during the Device Driver installation phase which follows the software installation phase, both Windows Vista and Win7 present an additional dialog screen which is not presented on either Win2K or

WinXP. This dialog confirms the Device Driver Certification status and presents an "Install" button to proceed with installation.

Following the software and driver installation process, the further actions you may need to perform depend on the PC system you are using, and on the version of the Windows operating system being used.

For NUCLEUS systems it will be necessary to Shutdown and Restart the system. This is necessary because of the need to trigger a new Windows Plug-and-Play detection cycle for the X10i hardware embedded on the Nucleus system board itself.

On systems other than Nucleus, new Plug-and-Play detection cycles will be triggered by connecting USB cables to each XLine board.

System using Windows Vista and Windows 7 operating systems, on detecting each instance of XLine hardware (the embedded hardware in the case of Nucleus) and/or each USB-connected XLine device, will automatically load the appropriate device drivers without any further intervention (the most that will be seen are a series of small message boxes reporting the hardware detection and loading of the appropriate drivers).

Systems running Windows 2000 and Windows XP will need some manual assistance the first time the new or re-connected XLine hardware is detected on specific USB ports. These systems will present a series of "Found New Hardware" dialogs up to two times for each XLine board detected (once on detecting "X10i Firmware Loader" and again on detecting "Xline X10i Driver"). Each time the Found New Hardware dialog appears:
- To the question about using Windows Update, select "No, not this time".
- To the question about which software to install select "Install the software automatically".

Provided no errors were encountered during the above steps, your XLine hardware devices and Development Kit are now ready for use. If you wish to double-check the correct installation of hardware, navigate through the following menus:
   Windows Start menu - Control Panel – System – Device Manager – Universal Serial Bus Controllers

You should see as many instances of "Xline X10i Driver" in Device Manager as there are XLine boards installed on your system.

You may now run the Demonstration programs provided and/or compile and run new XLine application programs on your XLine hardware.

2.1.2   Uninstalling under Windows

If you wish to uninstall the XLine software, or if you wish to delete the current installation and re-install a fresh new installation, navigate to Windows Control Panel – Add and Remove Programs [Programs and Features -- Programs] and select Xline 32/64-bit Development Kit – Uninstall. You should also uninstall the Heber Windows Driver Package(fflyio) from this list.

If re-installing the Development Kit to replace an earlier installation, remember to backup any modified files that you wish to save from the old installation directory before proceeding.

2.2   Installing under Linux

2.2.1 Extracting the Development Kit

The most up-to-date versions of development kits can be obtained from:
        http://www.heber.co.uk/support/downloads

Many of the following installation steps require 'root' privileges, therefore either log in as root and/or add the command 'sudo' at the start of all of the following command lines.

Next decide on a suitable location to install the XLine Development Kit (/usr/src is recommended) and copy the archive 'xlinedevkit.tar.gz' ['xlinedevkit_x64.tar.gz'] to that directory. Change into that directory and then unzip the supplied tar archive:

- `cd /usr/src`
- `tar –xzvf xlinedevkit.tar.gz  [xlinedevkit_x64.tar.gz]`

For the remainder of this guide it will be assumed that the XLine Development Kit has been installed under directory /usr/src.

2.2.2 Building the kernel module under Linux 2.6.* and later

As part of installing the development kit, it is necessary to compile the Xline device driver module to match your running Linux kernel. How this is done depends on how the kernel itself was generated.

If your kernel is part of a pre-compiled package (eg. is the result of installing a standard Linux distribution package online or from DVD), you should download and install the "linux-headers package" that exactly matches this kernel (command 'uname –r' will report the kernel version currently running). By default the headers package will be installed in '/usr/src/linux-headers-<kernel_version>'.

On the other hand your kernel may be a specially built custom version where certain features have either been added or removed and the kernel re-generated. Building a custom kernel requires downloading and installation of the full Linux source tree and running the kernel configuration process. By default the Linux source tree is installed under '/usr/src/linux-<kernel-version>'.

The Xline driver compiling process below need to include a reference either to the linux-headers path or Linux source tree path; the following commands assume that linux-headers are being used, use the Linux source tree path instead if running a custom kernel.

- `cd /usr/src/xlinedevkit/driver/`
- `make –C /usr/src/linux-headers-$(uname -r)/  SUBDIRS=`pwd` modules`

Note that –C is uppercase-C and the characters surrounding pwd are backticks.

If the build completes without error, the file xline.ko will have been created in the local directory. To install this driver into the currently running kernel, issue the command:

- `make –C /usr/src/linux-headers-$(uname -r)/  SUBDIRS=`pwd` modules_install`

`modules_install` will copy the newly created xline.ko file to directory:

        `/lib/modules/<current_kernel_ver>/extra`

The `modules_install` instruction also automatically runs program 'depmod', so Linux will know where to find the XLine driver when an Xline board is plugged in or the 'modprobe' command is used (see below).

2.2.3 Installing the fflyusb Shared Libraries

To obtain software access to the XLine board, a user library fflyusb.so [fflyusb_x64.so] is supplied - this should be copied to one of the common Linux Shared Library directories (e.g. /usr/lib or /usr/local/lib). A symbolic link should also be created. If you intend to work on both 32-bit and 64-bit Xline software you should copy and symbolically link BOTH 32-bit and 64-bit library files.

- `cd /usr/src/xlinedevkit/lib`
- `cp –a fflyusb.so  /usr/lib`
- `ln –s /usr/lib/fflyusb.so  /usr/lib/libfflyusb.so`

- `cp –a fflyusb_x64.so  /usr/lib`
- `ln –s /usr/lib/fflyusb_x64.so   /usr/lib/libfflyusb_x64.so`

Note the third command (ln) is: l for Lima, n for November. These libraries are required to compile your application programs.

2.2.4 XLine Device Permissions under Linux

The XLine device drivers are managed by the 'udev' mechanism in Linux. Under this mechanism, when an XLine board is plugged in to the system, the device automatically appears under directory '/dev' in the system. The first XLine board connected will be labelled '/dev/Xline_0', the next board will be '/dev/Xline_1', and so fourth (note upper-case X).

So that normal users in addition to root user have access to the xline board(s), device-permissions for the Xline devices must be set by adding an additional rule to the udev system. This is done as follows:

- Check if the file '`/etc/udev/rules.d/50-udev.rules`' exists already; if not, create the file.
- Add the following line to the file:
  `KERNEL=="Xline_?",SUBSYSTEM=="Xline",MODE="666"`

Save the file changes, then disconnect and reconnect the XLine board for the changes to take effect.

'50-udev.rules' is a file intended for generic udev rules; note that the first two terms in the rule use double-equals (==) and the third term is a value assignment with single-equals (=); 666 grants Read/Write access to Owner, Group and Global.

2.2.5 Starting and stopping the XLine Driver

When all of the above steps have been completed, disconnecting and reconnecting the Xline board should cause the Xline device driver to be loaded automatically each time.

Without connecting the board, you can also load the Xline driver from the default /lib/module directory using command:
- `modprobe xline`

An alternative mechanism that doesn't expect the driver to be installed in the default /lib/module directory is to use the command:
- `insmod <path>/xline.ko`

To confirm that the driver is loaded you can do either of the following:
- Issue the command '`lsmod`' to display all loaded device drivers. An entry for 'xline' confirms that the driver is loaded.
- Examine the end of the system log with the command '`less /var/log/messages`' (the location of system log does vary with Linux distribution). You should see message '`usbcore: registered new driver Xline`'.

When an XLine board is plugged in, it takes a couple of seconds for the board to download firmware and become ready for use. You can confirm that the board was recognised and is ready for operation by again checking the system log: there will be several entries relating to Heber, XLine and device /dev/Xline_0, 1, etc.

If you wish to remove the xline driver from the running Linux kernel, issue the command:
- `rmmod xline`

3 DEMONSTRATION PROGRAMS

A number of demonstration programs are supplied with the XLine Development Kit. They give examples of how to use the API calls described in the X10i Software User Manual, included as part of the Development Kit.

Within the "bin" directory is an executable diagnostic program named **X10Diag.exe** as well as a XML script entitled cctalk_4_1. This easy-to-use program will allow you to test a subset of the XLine's functionality. Please note that this program is currently only available for Windows, and will run on both 32-bit and 64-bit platforms.

Sample programs that have '_x64' in their names are compiled for 64-bit operation; the remainder are compiled for 32-bit operation.

When ready to write software for the board, the user should refer to the "sample" directory. Twelve demonstration programs are provided, each demonstrating specific features of the XLine API, and it is hoped that the supplied source code will prove useful to customers. The demonstration programs can be run on Windows and Linux operating systems, although the Linux samples may need to be recompiled on the target machine before running them.

Once running, each program responds to keyboard commands. The command keys for each program are displayed when the program is launched. Although the command keys are specific to each program, all programs can be terminated by typing "C".

Following is a list of all supplied demonstration programs along with a brief description:

- **cctalkdemo**: This program demonstrates the ccTalk mode 1 operation. A pre-defined message is sent to any connected ccTalk devices and it is possible to read or empty the ccTalk receive buffers.
- **eepromdemo**: This demonstrates EEPROM reading and writing.
- **fadedemo**: This program demonstrates lamp fading by slowly turning on all output lights and then turning them off again. This results in a "caterpillar" effect. The fade speed can be controlled by turning on or off IP16 to IP23 (the speed is increased by turning on more inputs).
- **Inpmuxdemo**: This demonstrates input multiplexing. There are four channels (corresponding to OP12-OP15) each containing 24 inputs (corresponding to IP0-IP23). The XLine board internally multiplexes these inputs and outputs together to provide 96 inputs.
- **iodemo**: This demonstrates various IO functionality.
- **paralleldemo**: This demonstrates the ability to drive parallel devices (e.g. coin hoppers).
- **randomdemo**: This demonstrates the random number generation facility (this is not available under X15).
- **reels**: This demonstrates reel spinning.
- **serialdemo**: Demonstrates serial communications. A good way to test this is to connect an RS232 null-modem cable between an XLine board and a PC and then use a program like HyperTerminal to send and receive data.
- **spidemo**: This demonstrates communication with a Starpoint Electronic Counter (SEC) device using the SPI protocol. The program displays "HEBER" plus a number (which is incremented each time you press a key) on the SEC device.
- **sramdemo**: This demonstrates SRAM memory reading and writing.
- **timedemo**: This demonstrates communication with the PIC security controller. Each time a key is pressed the Real Time Clock value displayed will be updated. The timedemo program also reports important version information about: the version of the driver, version of the API library, version of the PIC, etc. etc.
- **authenticatedemo**: This demonstrates the unlocking procedure for the X15 board only.

3.1 Compiling sample programs

NOTE: The running of any application programs on the Xline board requires that either the Development Kit version of the security library, or a customer's own custom security library be linked in. Where designs in the past were supplied with a 32-bit security library only ('unlockio.lib'), please ask Heber Ltd. to supply an appropriate 64-bit security library ('unlockio_x64.lib') to match your custom design for 64-bit development.

3.1.1 Building samples under Windows

The following description assumes that the Windows version of the development kit was installed under folder C:\heber\Xline. Adjust this path as necessary to suit your installation.

As an example, building the 'reel' project will be described, but the building of all samples will follow the same steps. Explanations apply to Visual C++ version 6 but similar steps should be followed for other versions/compilers.

- Copy the reel folder into your working directory.
- Create a new project in Visual C++ and add "reels.cpp" to the source file list.
- Access the setting of your project (project\setting). Select "C/C++" tab and pre-processor drop down menu from there. In the additional include directory, insert the path of XLine include files (C:\heber\Xline\include as well as C:\heber\Xline\samples)
- Select the "Link" tab and the input drop down menu from there. In the additional library path, insert the path of the XLine library files. (C:\heber\Xline\lib) and specify which library you need to identify in the project option (both fflyusb.lib [fflyusb_x64.lib] and the customer-specific secured unlockio.lib [unlockio_x64.lib] file supplied by Heber need to be present)
- Press 'OK' to change settings.

Rebuilding the project should now compile reels.cpp correctly.

As an alternative for compiling the programs, sample makefiles (makefile.win [makefile_x64.win]) are provided.

3.1.2 Building sample under Linux

NOTE: The running of any application programs on the Xline board requires that either the Development Kit version of the security library, or a customer's own custom security library be linked in. Where designs in the past were supplied with a 32-bit security library only ('unlockio.so'), please ask Heber Ltd. to supply an appropriate 64-bit security library ('unlockio_x64.so') to match your custom design for 64-bit development.

The 'samples' directory contains a file 'makefile.linux' ['makefile_x64.linux']. This script, as shipped, builds all of the demonstration program in one operation. The file can be altered to suite your needs; eg. to compile just one sample program. To build the programs, execute the command 'make –f <makefile_name>'. The sample programs, and each customer's own application programs will need to be built using the customer-specific secured library 'unlockio.o' ['unlockio_x64.o'] supplied by Heber if running on customer-specific secured X10i hardware, rather than the Development Kit hardware.

4 CONNECTING DEVICES TO THE XLINE BOARD

The XLine board is fitted with the following connectors:

Table 1 - Table of Connectors

Ident	Type	Function
P1A	USB Type B	USB (connectors in parallel)
P2	34W Header	I/O 1
P3	50W Header	I/O 2
P4	20W Header	ccTalk Channels A & B
P5	4W AMP	+12V Power in. (Hard Disk style connector)
P6	16W Header	I/O 3
P7	3.5mm Stereo Jack Socket	Audio In

The connectors on the XLine board are arranged as follows:
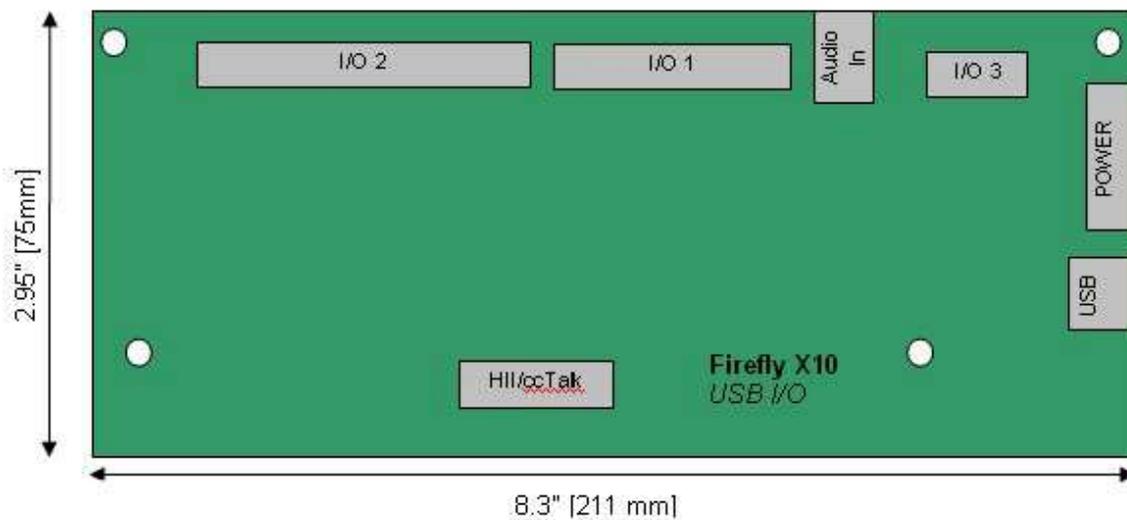
Note:
The X10, X10i and X15 boards have the same set of connectors which are fully pin compatible, however the layout of the connectors is different on the X15 board.

4.1 P2, I/O connector

This is a 34 way Header.

Table 2 – P2, I/O 1 Connector-1

Reference: P2
Type: 34W Header
Description: I/O 1

Left signal	Pin	Pin	Right signal
Open Drain Output OP0/Input IP16	1	2	Open Drain Output OP1/Input IP17
Open Drain Output OP2/Input IP18	3	4	Open Drain Output OP3/Input IP19
Open Drain Output OP4/Input IP20	5	6	Open Drain Output OP5/Input IP21
Open Drain Output OP6/Input IP22	7	8	Open Drain Output OP7/Input IP23
Open Drain Output OP8	9	10	Open Drain Output OP9
Open Drain Output OP10	11	12	Open Drain Output OP11
Open Drain Output OP12	13	14	Open Drain Output OP13
Open Drain Output OP14	15	16	Open Drain Output OP15
Input IP0	17	18	Input IP1
Input IP2	19	20	Input IP3
Input IP4	21	22	Input IP5
+12V Current Sensed	23	24	+12V Power
Ground (0V)	25	26	Ground (0V)
Loudspeaker (left)+	27	28	Loudspeaker (right)+
Loudspeaker (left)-	29	30	Loudspeaker (right)-
Ground (0V)	31	32	Ground (0V)
Left Audio Line In	33	34	Right Audio Line In

4.2 P3, I/O connector-2

This is a 50 way Header. It may be fitted with a ribbon cable assembly to jump to a 50 way 'D' Type connector on an I/O panel.

The high current outputs should use all three connections if the load will draw a high current. Otherwise, only one of the connections needs to be made. Similarly, sufficient ground connections should be used to meet the maximum load current expected.

Table 3 – P3, I/O 2 Connector

Reference: P3
**Type:**   50W Header
Description: I/O 2

Left Label	Pin	Pin	Right Label
Open Drain Output OP16	1	2	Open Drain Output OP17
Open Drain Output OP18	3	4	Open Drain Output OP19
Open Drain Output OP20	5	6	Open Drain Output OP21
Open Drain Output OP22	7	8	Open Drain Output OP23
Open Drain Output OP24	9	10	Open Drain Output OP25
Open Drain Output OP26	11	12	Open Drain Output OP27
High Current Output OP28	13	14	High Current Output OP28
High Current Output OP29	15	16	High Current Output OP28
High Current Output OP29	17	18	High Current Output OP29
High Current Output OP30	19	20	High Current Output OP30
High Current Output OP31	21	22	High Current Output OP30
High Current Output OP31	23	24	High Current Output OP31
Input IP6	25	26	Input IP7
Input IP8	27	28	Input IP9
Input IP10	29	30	Input IP11
Input IP12	31	32	Input IP13
Input IP14	33	34	Input IP15
+12V	35	36	+12V
+12V	37	38	+12V
Ground (0V)	39	40	Ground (0V)
Ground (0V)	41	42	Ground (0V)
Ground (0V)	43	44	Ground (0V)
Security Switch SW1-4 common	45	46	Ground (0V)
Security Switch SW 1	47	48	Security Switch SW 2
Security Switch SW 3	49	50	Security Switch SW 4

4.3 P4, ccTalk Channels A & B

This connector provides two ccTalk interface channels. It may be fitted with a 20 way header and the 20 way ribbon split to provide two Industry Standard 10 way ccTalk connections.

Table 4 - P4, ccTalk Channels A & B

Reference: P4
Type: 20W Header
Description: ccTalk Channels A & B

DATA Channel A	1	2	Ground (0V)
BUSY Channel A	3	4	Ground (0V)
RESET Channel A (*Output OP8)	5	6	
+12V Power	7	8	Ground (0V)
Ground (0V)	9	10	+12V Power
DATA Channel B	11	12	Ground (0V)
BUSY Channel B	13	14	Ground (0V)
RESET Channel B (*Output OP9)	15	16	
+12V Power	17	18	Ground (0V)
Ground (0V)	19	20	+12V Power

ccTalk CHANNEL A (pins 1–10), *ccTalk CHANNEL B* (pins 11–20)
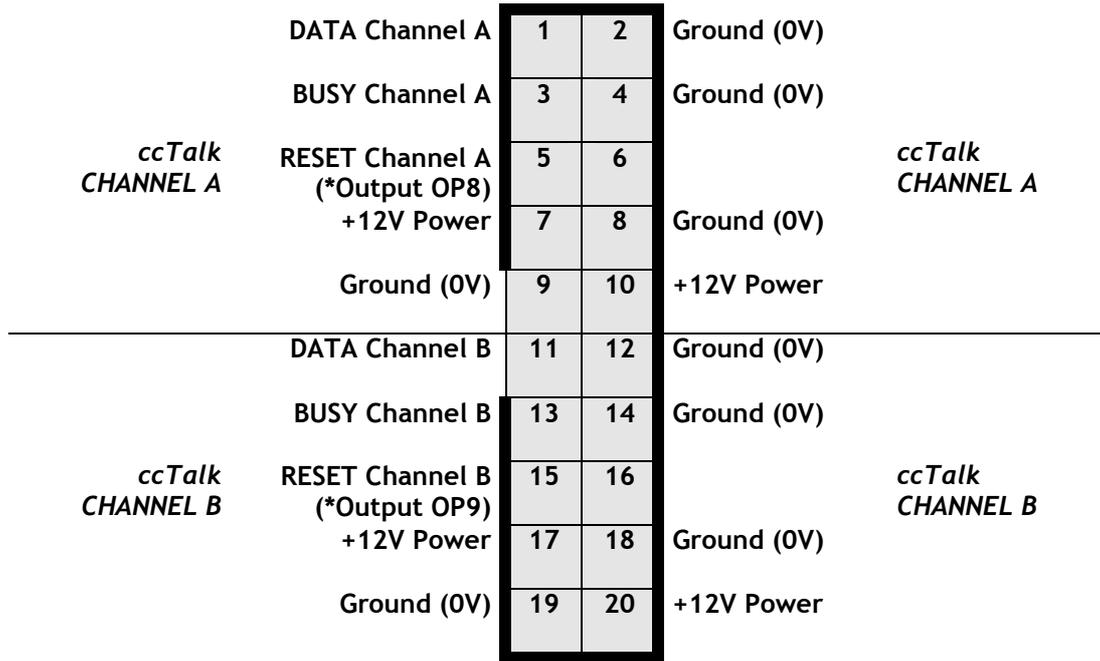
Note: The ccTalk receiver is configured for +5V operation. If the interface is operating at +12V levels, then R32 should be removed and fitted in R33 position instead. If this is not done, the ccTalk interface will work but with slightly reduced noise immunity and will be pulled up to 5V by the XLine board. This is unlikely to cause any communication problems.

4.4 P6, I/O connector-3

This is a 16 way Header.

Table 5 – P6, I/O 3 Connector

Reference: P6
Type: 16W Header
Description: I/O 3

Ground (0V)	1	2	Ground (0V)
Serial RS232 Input RXD A	3	4	Serial RS232 Output TXD A
Serial RS232 Input CTS A	5	6	Serial RS232 Output RTS A
* Serial TTL Input RXD B	7	8	* Serial TTL Output TXD B
* +12V output	9	10	* -12V output
Auxiliary CMOS Output AUX0	11	12	Auxiliary CMOS Output AUX1
Auxiliary CMOS Output AUX2	13	14	Auxiliary CMOS Output AUX3
Auxiliary CMOS Output AUX4	15	16	Auxiliary CMOS Output AUX5

+12V and –12V outputs are only available if a +12V source has been connected to P5. However, RS232 signal levels are generated on the XLine board and it is not necessary to connect a +12V power source to use RS232 signals. The +12V and –12V outputs are intended for use by a BACTA port.

4.5 P7, Audio connector

This is standard 3-wire 3.5mm stereo jack Socket. It duplicates the Audio Line-In signals on Pins 33 & 34 of Connector P2 to allow connection to Firefly 700 via a standard jack-jack lead.

Table 6 – P7, Audio-In Connector

Reference: P7
Type: 3.5mm stereo jack Socket
Description: Audio In