ICARC Fox UART

William Robison

February 10, 2026

This is the start of a manual for the ICARC FOX 4-port UART. It covers the 102-73226 series boards.

It is a work-in-progress right now so suggestion for updates may be sent to **kc0jfq@n952.ooguy.com**.

Full size documents may be found here: http://n952.ooguy.com/HamRDF

LaTeX Source Files:

0.		//home/wtr/Fox_Tx73181/trunk/FOX_UART.tex
1.	27	//home/wtr/Fox_Tx73181/trunk/FOX_ICARC_zNEO_Hardware.tex

Contents

List of Figures

List of Tables

Chapter 1

Glossary of Terms

There is an attempt being made to use some terms in this document in a precise manner. Some of the discussions become a bit muddled when terms are used casually.

1.1 Fart

A type of gaseous emission.

1.2 xxx

xxx is a type of moonshine liquor.

Chapter 2

Motivation

Debugging things

Programming them as well

2.1 Requirements

These are required for operation.

2.2 Desirements

These are desired features.

Chapter 3

Theory of Operation

The system consusts of a motherboard with a 4 channel USB UART and up to four daughterboards that provide the interface circuitry to the outside workd.

3.1 102-73226-2 Motherboard

Using the FTDIchip FT4232 USB-UART.

May be bus powered or self powered.

When self powered the regulatr may provide up to 10 watts (2 Amps).

3.2 102-73226-21 DE9F

Female (sockets) DE-9 connector IBM PC-AT serial port pinouts.

Jumpers to allow switching between DTE and DCE configuration.

3.3 102-73226-26 DE9M

Male (pins) DE-9 connector IBM PC-AT serial port pinouts.

Jumpers to allow switching between DTE and DCE configuration.

3.4 102-73226-41 ZiLoG Programmer

This daughterboard implements the half duplex serial programming interface to ZiLOG zNEO and eZ8 devices.

Used a tri-state buffer to drive the data pin to allow for higher speeds.

3.5 102-73226-41 Z8/eZ8/zNEO Programmer

This daughterboard is a dual-wide implementation of the 102-73226-41 programming card for use with the FT2232.

It adds a speed-up circuit to allow reliable operation at higher bit rates.

3.6 102-73226-62 ICOM CI-V

Logic level serial port using a 3.5mm TRS (stereo) headphone jack.

Jumpers allow configuration for full duplex operation to provide a serial interface to the Fox Transmitters or as a half duplex TS (mono) interface to allow accessing ICOM radios.

3.7 102-73226-63 CI-V Radio Programmer

Logic level serial port providing both a 3.5mm TRS (stereo) headphone jack and a 2.5mm TRS (stereo) headphone jack.

Extensive jumper array allow assigning any pinout to the 3.5mm/2.5mm jacks.

This board may be configured as a standard ICOM CI-V interface (half-duplex logic level) or as an interface for programming common hand-held tranceivers.

Added from the 102-73226-62 board is a 2.5mm stereo audio jack and a large jumper array. With both a 3.5mm and 2.5mm jack, the board provides for using pre-made audio cables to connect a handheld tranceiver.

The jumper array allows connecting any pin on the audio jack to either the **TxD** line or the **RxD** line. Dedicated jumpers allow the sleeve or the audio jack to be connected to ground.

Similar to the 102-73226-62, there is a means of using either the **RI*** net or the **TXDEN** net to enable the **TxD** line driver. ON the 102-73226-63 board this is available in the form of a jumper to make selection a bit easier to deal with. The **TxD** line driver may also be installed as a simple open-drain driver if speed is not an issue.

3.8 102-73226-67 CI-V Serial Card (FT2232)

This mimics the 102-73226-63 board in a dual-width card for use wit a FT2232 populated board.

Same jumper array as the 102-73226-63 with slightly more feature.

This dual-with board **must** occupy the **Channel-1/Channel-0** positions on the uART Motherboard. It is mechanically incompatible with the **Channel-3/Channel-2** position. The **Channel-A** position on the daughtercard, carrying the serial channel signals, mates with the **Channel-0** position on the motherboard.

This card is more-or-less compatible with the FT4232 chip. All of the modem control signals are correctly positioned. The **TXDEN** net moves from the **Channel-A** over to the **Channel-B** connector, becoming unusable. Jumpers are provided to configure the board for operation with the FT4232 device.

Speed-up logic improves edge transitions on the transmit data line. When the **TxD** line changes state, the tri-state buffer (on the **TxD** line) is enabled for about 50nS to provide a solid driving signal to the target system. After this short period expires, the open-drain driver hols the line in a low state, or a pull-up holds the line in a hi state.

If part of this capability/feature is not needed, some of the parts need not be populated.

The board is provisioned with two TRS (stereo) audio connectors to match that seen on most current handheld tranceivers. One being a standard 3.3mm jack and the other a 2.5mm jack. A jumper array allows any of the three pins to be connected to either the **TxD** line or the **RxD** line.

An additional pair of jumpers allows either *sleeve* to be connected to ground. If the neede ground net does not arrive on the sleeve of either jack, then a wire-wrap jumper may be installed.

3.9 102-73226-81 Isolated ICOM CI-V

Optically isolated version of 102-73226-62.

3.10 102-73226-91 Isolated RS485

Optically isolated RS485 interface.

Chapter 4

Configuration

in the file system.

4.1 Conf: 102-73226-2 Motherboard

Step 1

4.2 Conf: 102-73226-21 DE9F

Step 1

4.3 Conf: 102-73226-26 DE9M

Step 1

4.4 Conf: 102-73226-41 ZiLoG Programmer

Step 1

4.5 Conf: 102-73226-46 Z8/eZ8/zNEO Programmer

Step 1

4.6 Conf: 102-73226-62 ICOM CI-V

Step 1

4.7 Conf: 102-73226-63 ICOM CI-V

Step 1

4.8 Conf: 102-73226-67 ICOM CI-V

507 Step 1

4.9 Conf: 102-73226-81 Isolated ICOM CI-V

515 Step 1

4.10 Conf: 102-73226-91 Isolated RS485

523 Step 1

Chapter 5

Build Documents

The build documents are produced using *Eagle* V6.5.0.

All of these documents are produced using a combination of *Eagle* scripts, *Eagle* user language programs, and bash shell scripts.

5.1 Build: 102-73226-2 Motherboard

Step 1

5.2 Build: 102-73226-21 DE9F

Step 1

5.3 Build: 102-73226-26 DE9M

Step 1

5.4 Build: 102-73226-41 ZiLoG Programmer

Step 1

5.5 Build: 102-73226-46 ZiLoG Programmer

Step 1

5.6 Build: 102-73226-62 ICOM CI-V

Step 1

5.7 Build: 102-73226-63 CI-V Radio Programmer

Step 1

5.8 Build: 102-73226-67 CI-V Serial Card (FT2232)

601 Step 1

5.9 Build: 102-73226-81 Isolated ICOM CI-V

609 Step 1

5.10 Build: 102-73226-91 Isolated RS485

617 Step 1

5.11 Assembly

Start with the *Master Build Record* as a guide to assembling the board. Install the ceramic capacitors and the resistors first as these are relatively small and do not hinder access to the remaining parts.

638

Parts locations on the schematic and on the circuit board are listed in the *Master Build Record*. The schematic is a sheet.quadrant notation and the circuit board locations are from the *Pick & Place Zero Point* shown on the *PWB* (Printed Wiring Board) drawing. Note that the dimensions are taken from the top, so reverese direction on the bottom.

5.12 Schematics

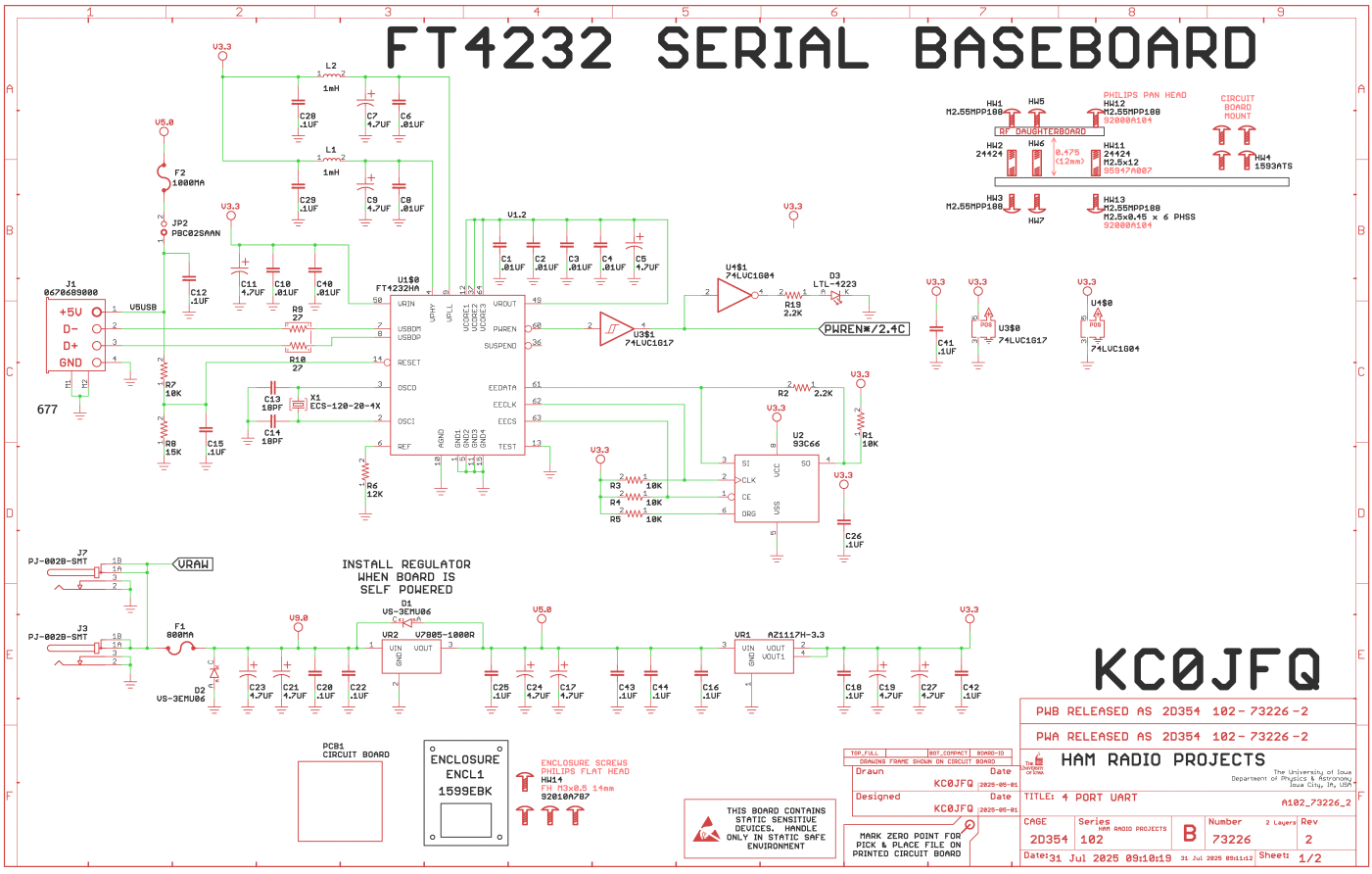
Most logic gates have their power pins represented on a separate symbol to reduce clutter around the pins of the logic gate. The power symbol is a bit elaborate looking to force, as much as possible, the power pins to be connected to the correct rails following a rotation or mirroring operation.

661 You will also note that **no** schematic symbols are arranged to look like their packaging. The schematic should **not** represent the board layout, that is not its function. The schematic provides the connection netlist for the circuit board.

The fist page of the schematic has some additional *clutter* around the title block. These simply document that various documentation artifacts appear on the circuit board drawing.

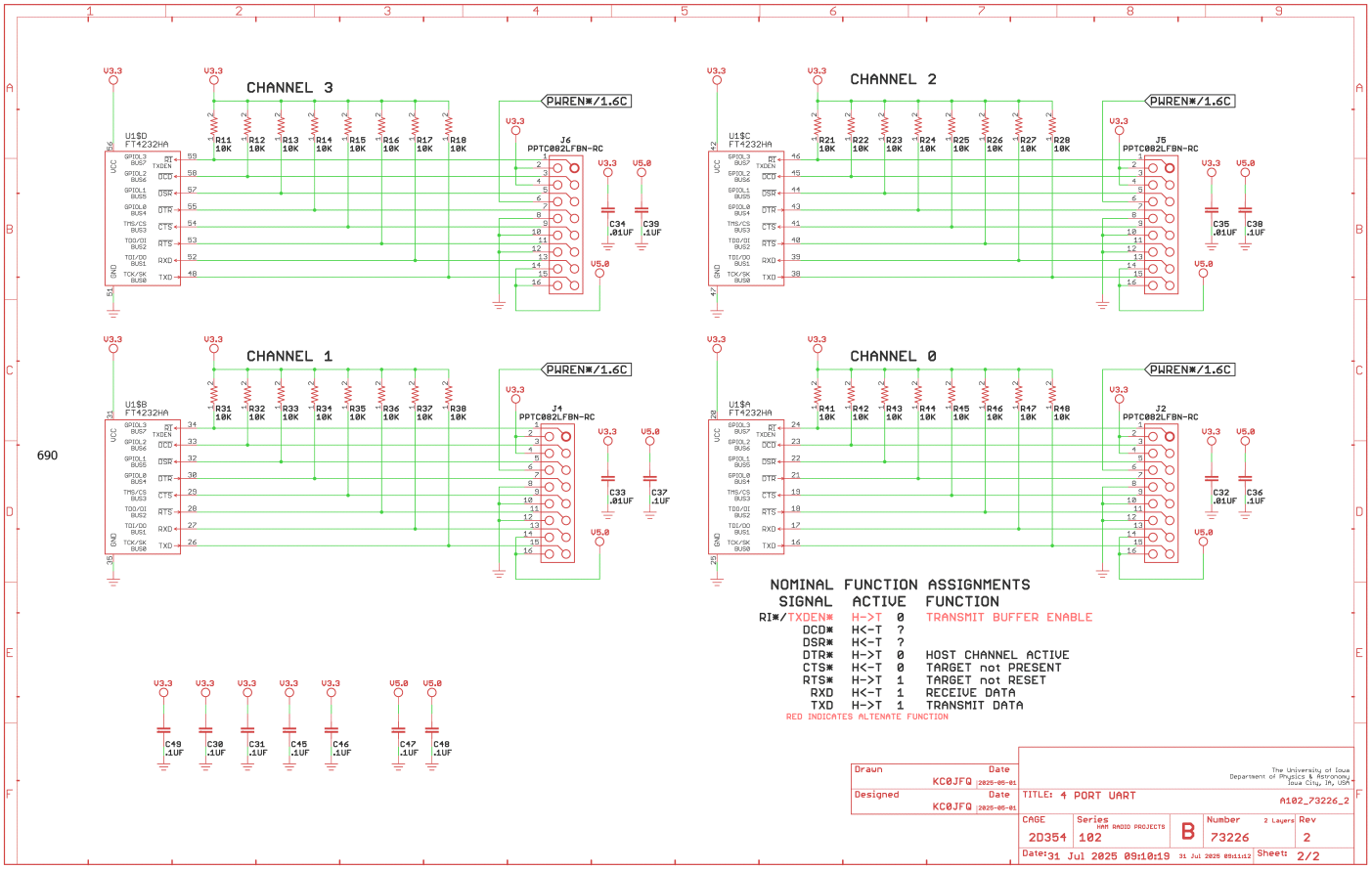
5.13 Sch: 102-73226-2 Motherboard

FTDIchip Quad UART



Dual Power Connectors to piggyback on USB Hub 12V supply.

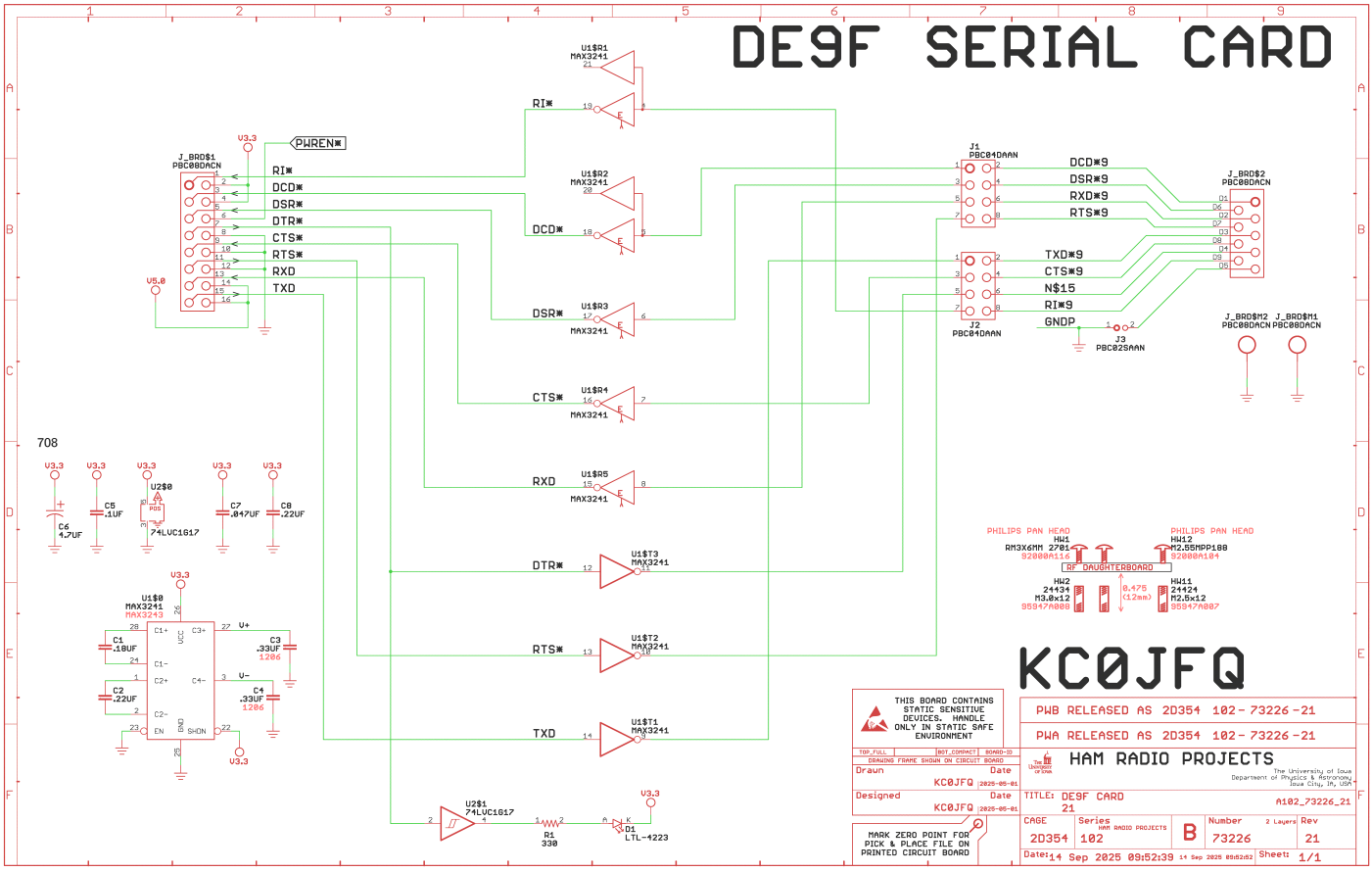
The four UART serial ports. Idenitcal pinout and mechanical interface.



Two of the ports have a bit more room to allow for DE9 connectors.

5.14 Sch: 102-73226-21 DE9F

DE9 connector, female.

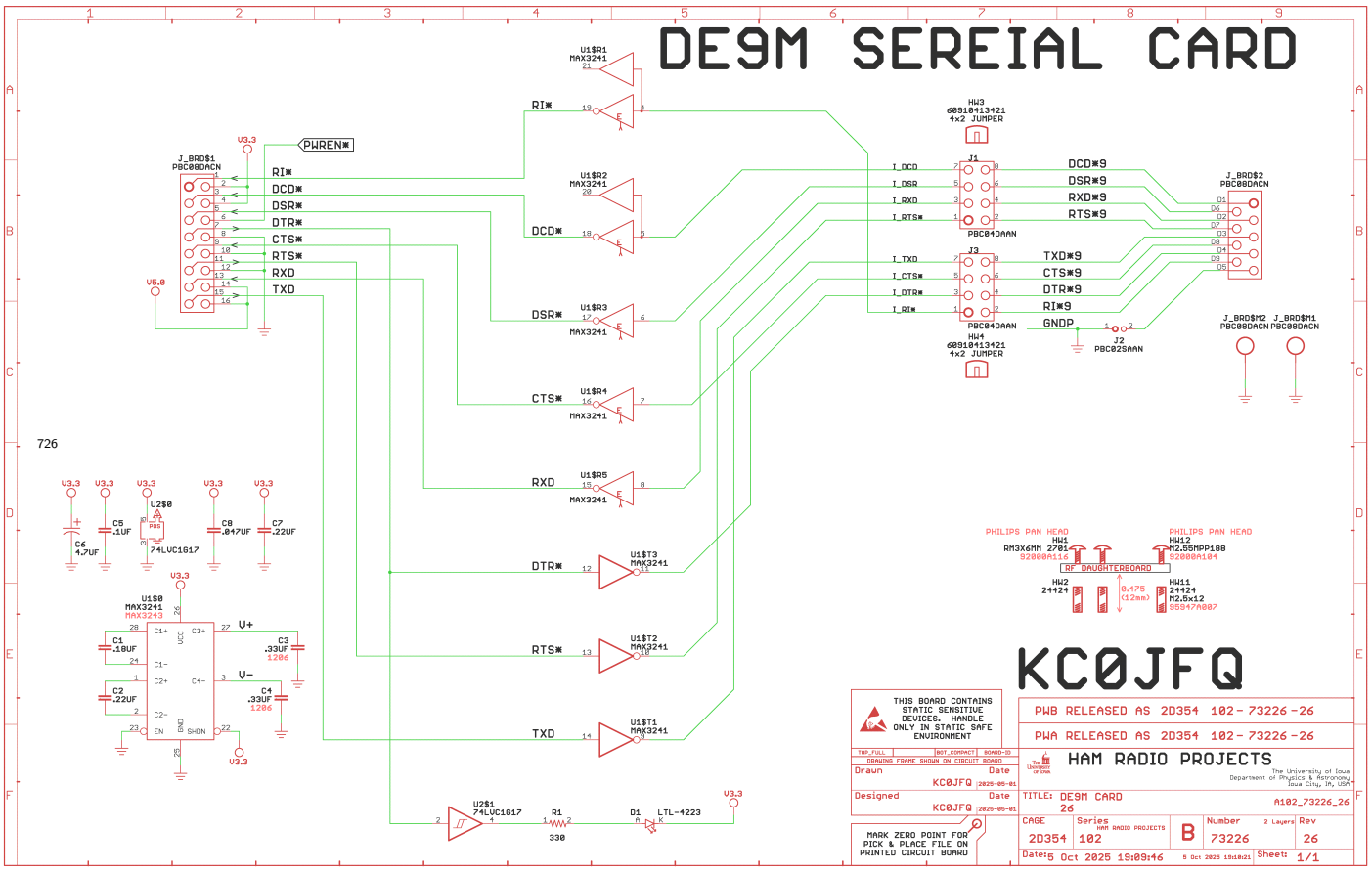


Jumper block straight through for standard pinouts.

Rewire for opposite with wire-wrap.

5.15 Sch: 102-73226-26 DE9M

DE9 connector, male.

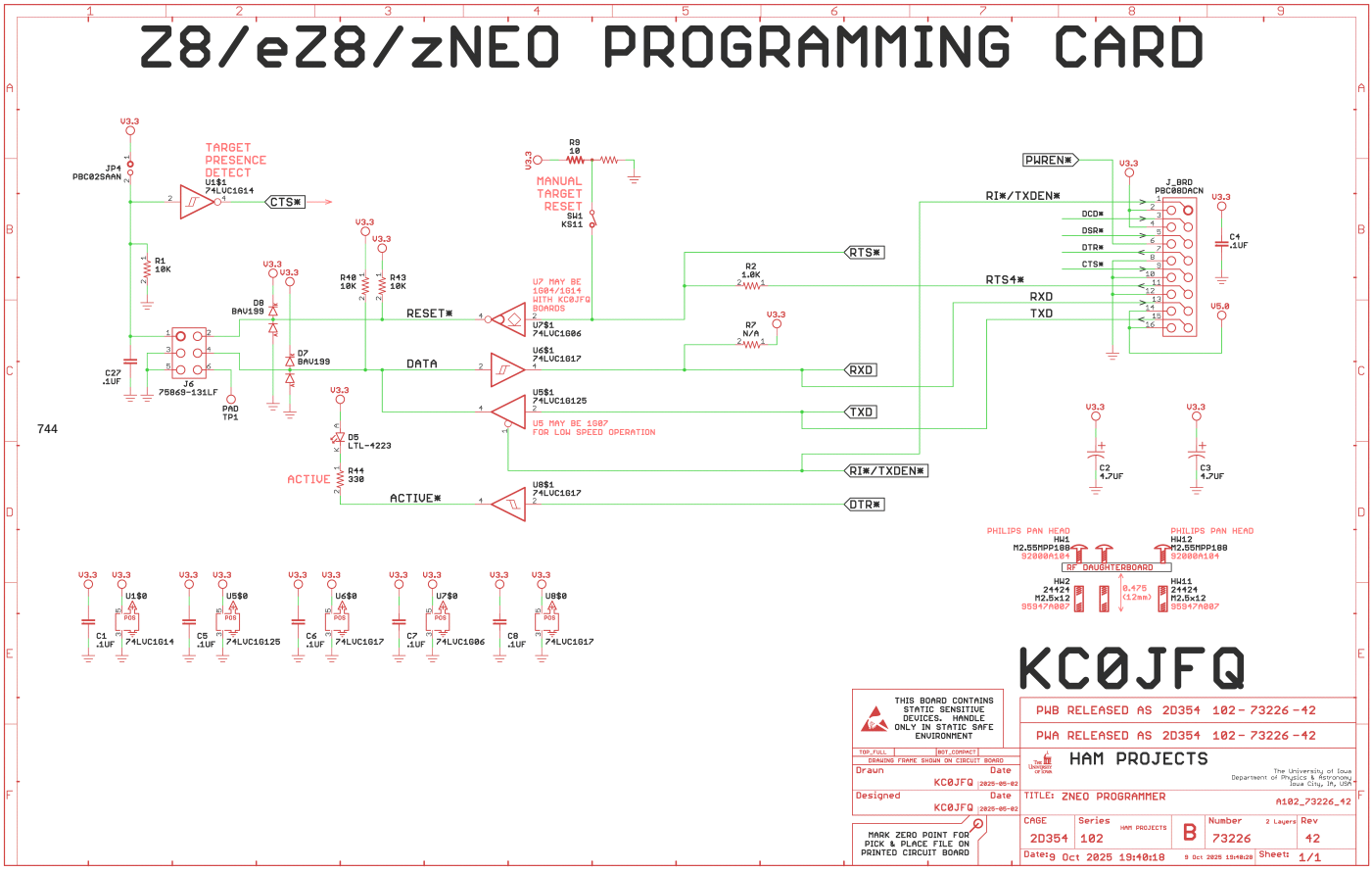


Jumper block straight through for standard pinouts.

Rewire for opposite with wire-wrap.

5.16 Sch: 102-73226-41 ZiLoG Programmer

ZiLOG zNEO and eZ8 programmer.



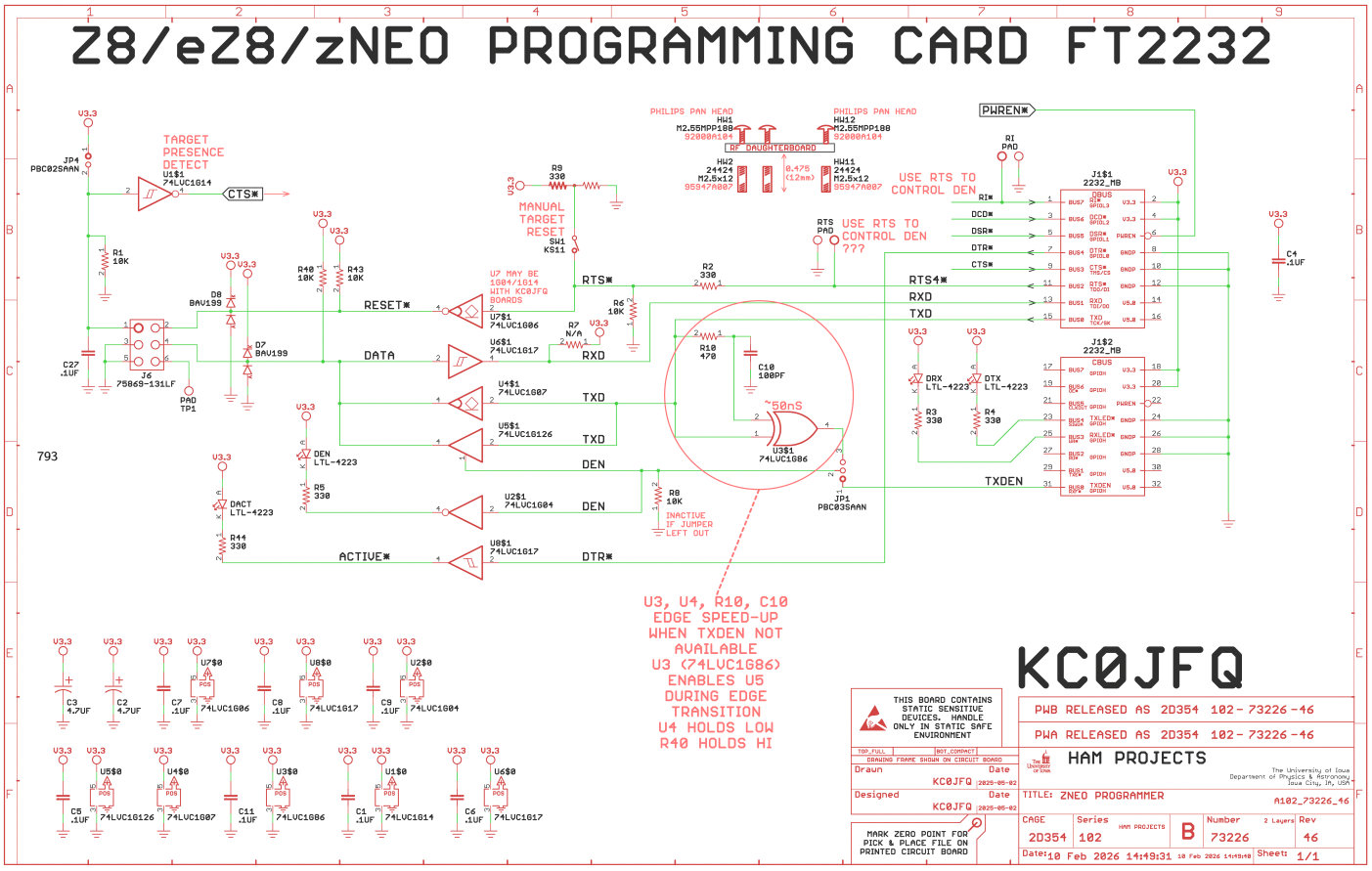
CTS* line reports when the target is attached. Pin 1 of the programming connector on the target should be connected to 3.3V.

The LED, D5, indicates when the host system is driving data to the target (during programming or dumping). It will be bright during programming and dim during a dump.

We drive the data to the target using a tri-state gate to provide good drive in both directions. This is an attempt to allow a reliable connection at high bit rates.

5.17 Sch: 102-73226-46 Z8/eZ8/zNEO Programmer

ZiLOG zNEO and eZ8 programmer.

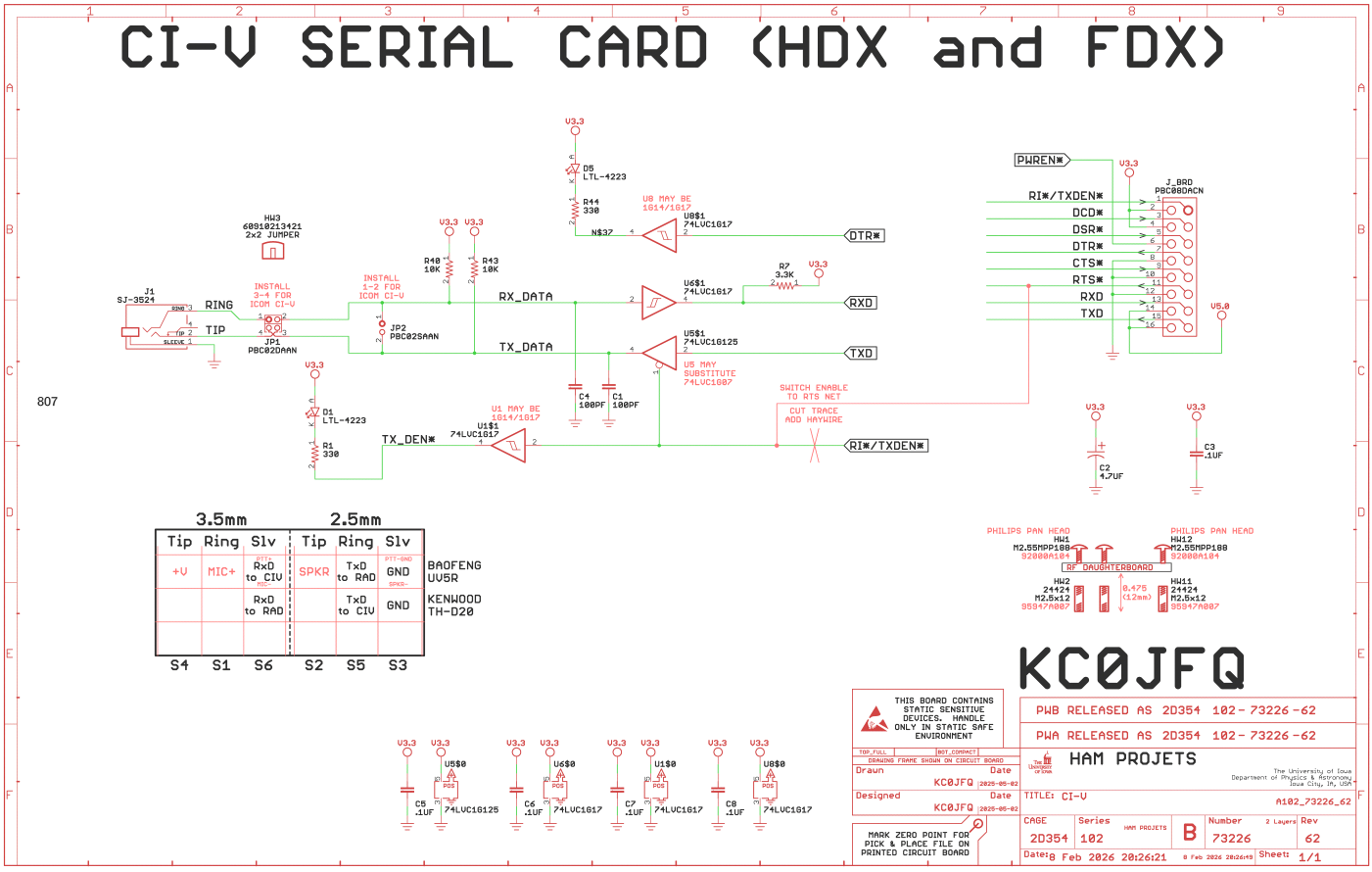


This card functions like the 102-73226-42 card while using the FT2232 configuration. This card is mounted in the **Channel-3/Channel-2** positions on the motherboard.

This can be configured to piggy-back a standard card on top of the **Channel-3** position by using a Samtec ESQ-108-14-G-D connector. See the discussion in section 5.20 on page 23.

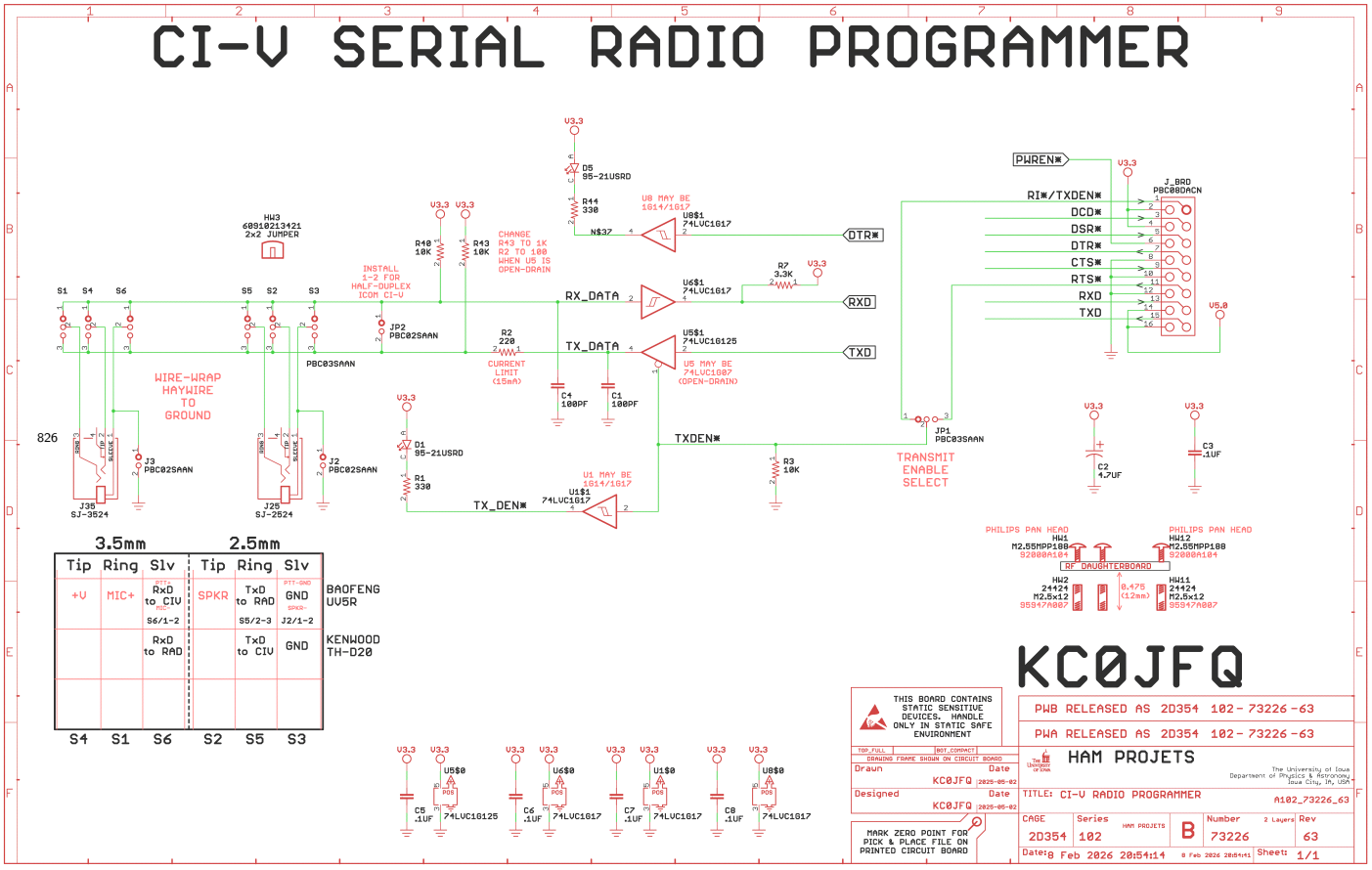
5.18 Sch: 102-73226-62 ICOM CI-V

Basic ICOM CI-V interface with full duplex capability.



5.19 Sch: 102-73226-63 CI-V Radio Programmer

3.5mm/2.5mm CI-V interface with full duplex capability.



The jumper array, positions S1..S6, connect **TxD** and **RxD** to any pins on the 3.5mm and 2.5mm jack. J3 and J2 provide a jumper from ground to the sleeve on the respective connectors.

JP1 is used to connect either the **RI/TXDEN***net of the **RTS*** net to the enable pin on U5. Leavig JP1 open allows R3 to enable U5.
There is not quite enough room on the card, without resorting to a 4 layer board, to provide a presence detect.

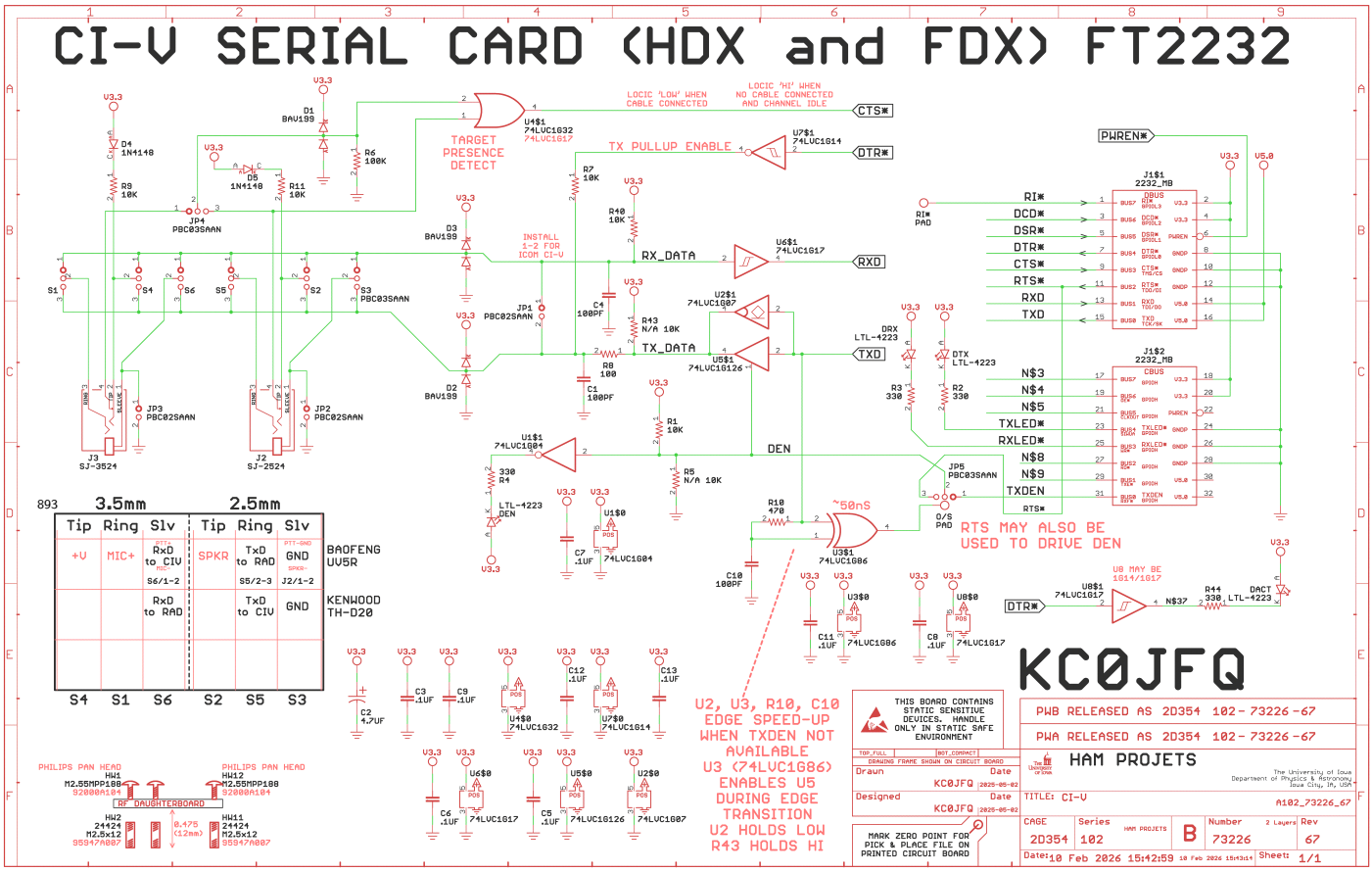
As on the 102_73226_42 board, the CTS* line reports when the target is attached. Pin 1 of the programming connector on the target should be connected to 3.3V.

The LED, **DEN**, indicates when the host system is driving data to the target (during programming or dumping). It will be bright during programming and dim during a dump.

The LED, **DACT**, indicates when the host system is accessing the channel. The host software is expected to assert the **DTR*** net when using the channel and to de-assert the **DTR*** net when done using the channel.

5.20 Sch: 102-73226-67 CI-V Serial Card (FT2232)

3.5mm/2.5mm CI-V interface with full duplex capability.



Similar to the 102-73226-63 card, but laid out on the dual-with card for use with the FT2232 device.

Mechanically and electrically, this board may be used when the motherboard is populated with an FT4232. This configurtion would result in the loss of one of serial ports unless J1$2 has a pass-thru connector installed, such as a Samtec ESQ-108-14-G-D, to pass the 2nd. channel up to another daughtercard.

This board follows the convention of using the **DTR*** net as a channel active indicator to illuminate the *IN USE* LED indicator. The **DTR*** net may also be used to drive a pull-up on the **TxD** line to the target system. R7/R43 deal with this capability, install R7 to use the **DTR*** net to drive the pull-up when the bchannel is selected. Install R43 for a basic pull-up that is continuously on whenever the motherboard is powered.

This boad also incorporates a speed-up circuit using U3, R10, and C10. When the **TxD** line changes state, the exclusive-or (U3) generates a short, approximately 50mS, pulse to enable U5 to drive the **TX__DATA** net to a hi state or a lo state. Once the **TxD** line becomes stable (keeping in mind that a bit time will be many microseconds) U3 is disabled and the open-drain driver (U2), and the pull-up (R7 of R43) keep the level stable.

This provides a means of speeding up the **TX__DATA** net without depending on the FT2232/FT4232 to correctly drive then **TXDEN*** net of the host software to correctly deal with the **DTR*/RTS*** nets.

This board, given the larger circuit card, also implements *target presence detect* function using the switched contact on the 3.5mm/2.5mm connectors.

With no cable attached, the pull-up ob the **TxD** and **RxD** line keep the **CTS*** net de-asserted. when both calbes are plugged in, the switch contact is isolated allowing the pull-down (R6) to assert the *target presence detect* signal (**CTS*** low).

Take note of the device specified for the U4 position. When using two cables, one in each audio jack, the 74LVC1G32 combines the status report from each channel. Both cables must be installed for the **CTS*** net to be driven to the active (i.e. low) state. Install JP4/1-2 for this configuration.

If there is a need to better aqccommodate single cable operation, where the combined status would present a problem, the U4 position may be populated with a 74LVC1G17 to, in effect, ignore the signal arriving at U4-1 (the 74LVC1G17 is a buffer and does not use pin-1). Now the JP4 jumper may be used to select between J3 and J2.

In both cases, the detection logic assumes that both the **TX__DATA** and **RX__DATA** nets coming from the target will be driven high when idle. With only one cable connected, if the target drives either net low, the detection logic will behave incorrectly.

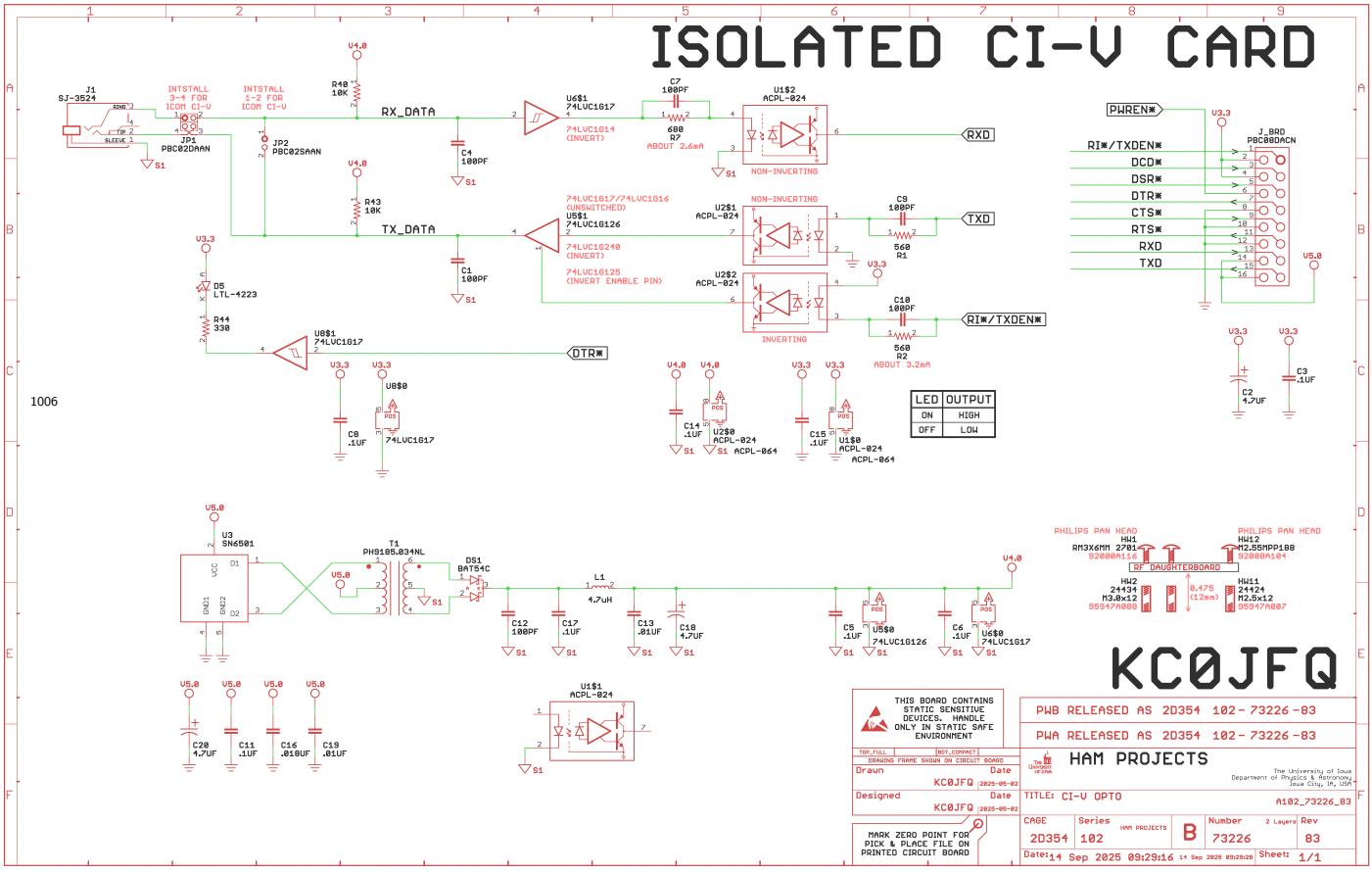
The individual pull-ups (R9 and R11) deal with the case where these conenctions are not used and, therefore, not connected to anything on this boad.

When connecting to a radio that supplies a voltage on the tip, D4 and D5 prevent current from flowing into the daughterboard.

This update also provides a current limit in the **TX__DATA** path in the form of R8. This resistor serves to limit the current that can be supplied by U5 to a maximum of 3.3mA shoiuld the jumper array be incorrectly configured.

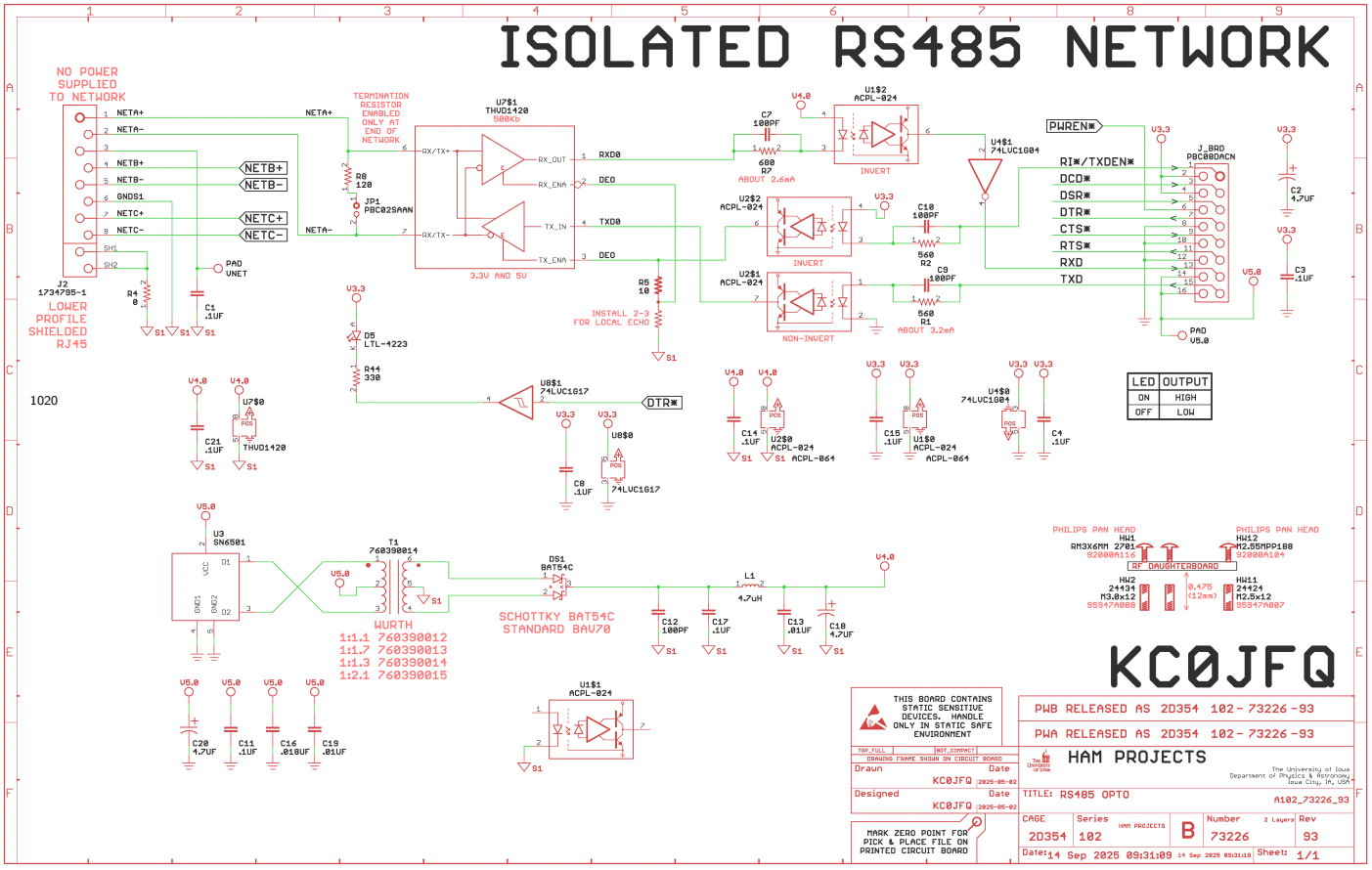
5.21 Sch: 102-73226-81 Isolated ICOM CI-V

Isolated ICOM CI-V interface with full duplex capability.



5.22 Sch: 102-73226-91 Isolated RS485

Isolated RS485 interface with half duplex capability.



5.22.1 zz

1028

5.22.2 zz

1034

Chapter 6

zNEO Programming Hardware and Utility

Source File: FOX_ICARC_zNEO_Hardware.tex

This utility is used to load the software image into the zNEO in the fox transmitter.

This utility is written in plain-old C to run on a Linux box. It makes use of the 102-73220-23 and 102-73220-33 boards to provide the programming connection into the target system.

This utility came about to address an issue with using the ZiLOG ethernet smart cable programmer. Something occurred with an update to Fedora40 that broke the **ZENETSC0100ZACG** device used by the author.

The replacement uses a standard *FTDIchip* USB UART device along with a small interface board to access the debug/programming port on the zNEO.

Help output from programming software:

```
./zNEO_P (V0.0/V2.10)
        ZiLOG ZNEO Programmer

    The following command line switches are recognized:

    -h              help file, abbreviated
    -H              help file, include config file and device file list

    -r              Reset Target (and exit)
    -e              erase only (and exit)
    -p              skip erase
    -f <freq>       define target crystal frequency (default: 20.000 MHz)

    -d              scan target flash to _pre.hex only
    -1              scan target flash to _pre.hex  BEFORE programming
    -2              scan target flash to _post.hex AFTER  programming

    -x <file>       input hex file (filetype, not supplied, must be .hex)
    -X <file.hex>   reformatted hex file (you must supply the .hex suffix!)
    -V              Verify only
    -l <file>       append to log file (include timetag on each line)

    -S <nickname>   Select device by name
                        EZ3PGM    38400
                        EZ5PGM    57600
                        EZ1PGM   115200
                        EZ2PGM   230400
                        EZ4PGM   460800
                        EZ8PGM   500000
```

Typical use:

```
./zNEO_P -SEZ5PGM -xfox_73181
```

The serial device selection is through the **-SEZ5PGM** argument. Note that this particular device operates the 57,600 bit/second rate to stay well within what the zNEO will tolerate. All of the listed bit rates should be within the capability of the zNEO *On Chip Debugger*. In practice, operation above 115,200 may not allow enough time for flash memory programming inside the zNEO. The **ZENETSC0100** programmer that this hardware replaces was operating at around 125,000 bits/second.

The basic processing time for programming a full (128K) device (erase, program, and verify), is a bit over 90 seconds. Page size is limited to 32 bytes during programming and expanded to 256 bytes during verify. Adding the **-1** adds about 30 seconds for the pre-read operation.

The hex file selection is through the **-xfox__73181** argument. The filetype of **.hex** is forced (the .hex must not appear). The device memory image can be saved both before and after programming using the **-1** and **-2** flags.

The hex file decoder in the utility is white-space insensitive. It will accept expanded hex files that have white-space embedded within the record as well as text appended to each line. The output produced when using the **-1** and **-2** flags will be correctly processed.

The pre and post **.hex** files are named based on the **-x** argument. The input filename is simply the supplied argument with the **.hex** appended.

The **-1** will save a file collected before the device is erased. It is named by appending **__before.hex** to the supplied filename.

The **-2** will save a file collected after the device is programmed. It is named by appending **__after.hex** to the supplied filename. Note that this image is collected from the target after programming to verify that the device was successfully programmed.

The **-X** is used to produce an *annotated* hex file. This simply spreads out the hex record to make manual inspection a bit easier. There is also a text dump appended to the hex data.

The **ZiLOG ZNEO Programmer** software will directly deal with this output. It removes white-space as the line of text is read. Extra characters after the checksum field are ignored.

6.1 Single Channel UART

This is the base board that holds the USB UART.

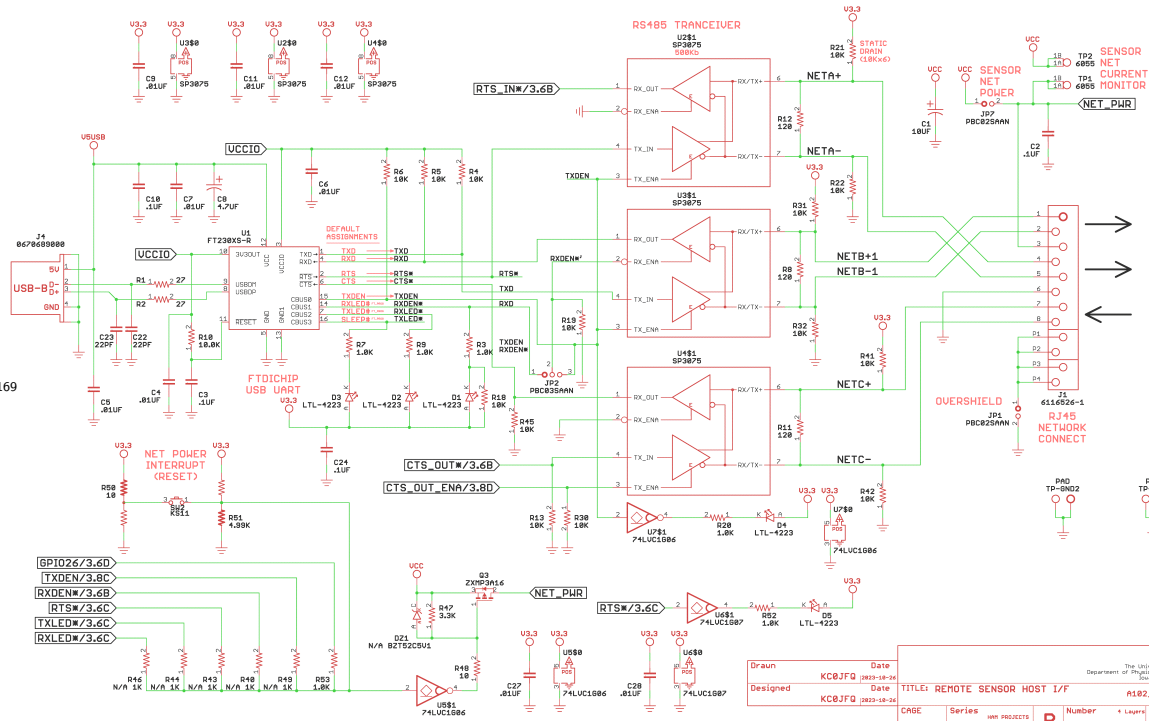


Figure 6.1: Single Channel UART

U1 is the USB UART device that provides a connection to the host system.

For our programming application the RS485 interface U2..U4 is not populated. We use this board as a place for the FT230X to live and a mounting point for the programming board.

The circuit provides visual indicators (LEDs) to allow monitoring activity.

This is the base board connector to the programming board.

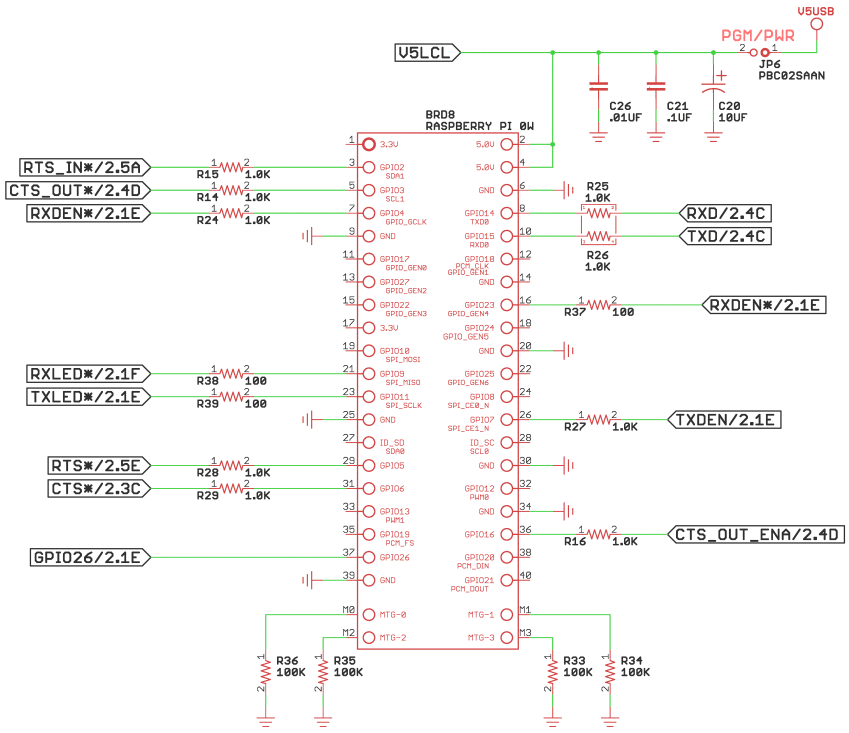


Figure 6.2: base board to programming board

This board was originally designed as a RS485 interface for a sensor network. In addition to the USB interface the board will also accommodate a *Raspberry PI ZERO* in place of the USB UART.

The *Raspberry PI ZERO* connector provides the interface to the programming adapter board. The R25/R26 resistor pair provides for swapping the TxD and RxD lines to the *Raspberry PI* connector should that be necessary.

6.2 ZiLOG eZ8 Programming Adapter

This is the full-duplex (FTDIchip device) to half-duplex (zNEO) interface drawing:

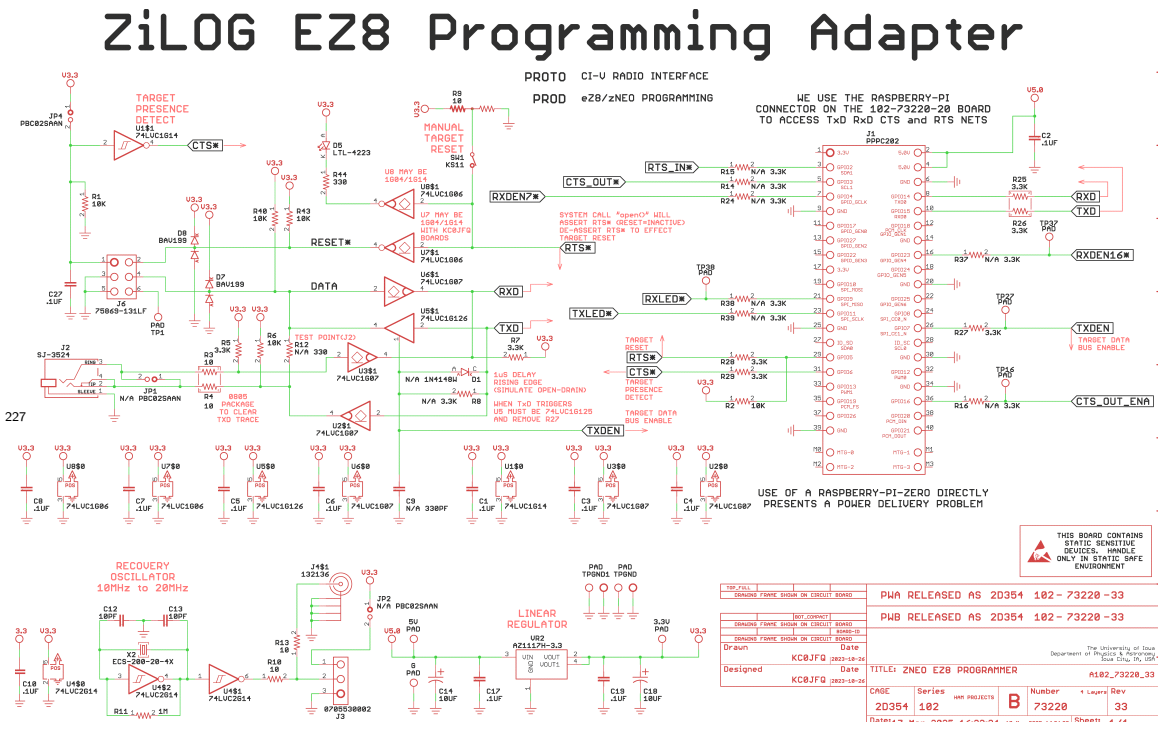


Figure 6.3: eZ8 Adapter

The programming board provides the interface between the full-duplex UART (TxD and RxD data) and the half-duplex interface to the eZ8/zNEO target. The eZ8/zNEO interface is open drain to allow either side to drive the data bus.

To improve speed, data from host(J1) to target(J6) is buffered using a tri-state gate(U5) that is enabled when transmitting. The use of a push-pull gate improves rise time for data being sent to the target.

The 74LVC1G126 device is pin compatible with a 74LVC1G07 open drain driver should this prove useful. One would expect to control the enable pin of U5 using the **TXDEN** net as the FT230X will set the TXDEN bit when it transmits and clear the bit when not transmitting.

The **Recovery Oscillator** shown in the lower left corner is there to provide a clock to a board where the zNEO internal oscillator configuration has been corrupted or improperly programmed. Pin 2 can be used by itself to effect the recovery operation by connecting to the XIN pin on the zNEO (LQFP64 pin 64). Power and ground appear on the connector (J3) should it become necessary for either of these signals to be used.

It should be possible to use a *Raspberry PI ZERO* to drive this board. The R25/R26 resistor pair is also present on this board to allow TxD and RxD to be correctly routes.

The 102-73220-33 board is not mechanically well suited for use with the *Raspberry PI ZERO* as it extends outside the edges of the *Raspberry PI ZERO*. It should, none-the-less be electrically compatible.

Power to the *Raspberry PI ZERO* may present a problem. A regulated 5V supply would need to be connected to either the *Raspberry PI ZERO* has a microUSB connector for providing power which may be used to also power the programming board. There are also a pair of pads on the 5V and GND nets on the programming board that may be used to feed the *Raspberry PI ZERO*.

6.3 Four Channel UART

This is a UART motherboard with 4 identical channels that fits in our Hammond 1599E box. This board is intended to condense the connections required to debug the Fox Transmitter to a single board.

This makes use of the **FTDIchip** *FT4232* quad UART to provide 4 serial channels in a single 64 ping flat-pack.

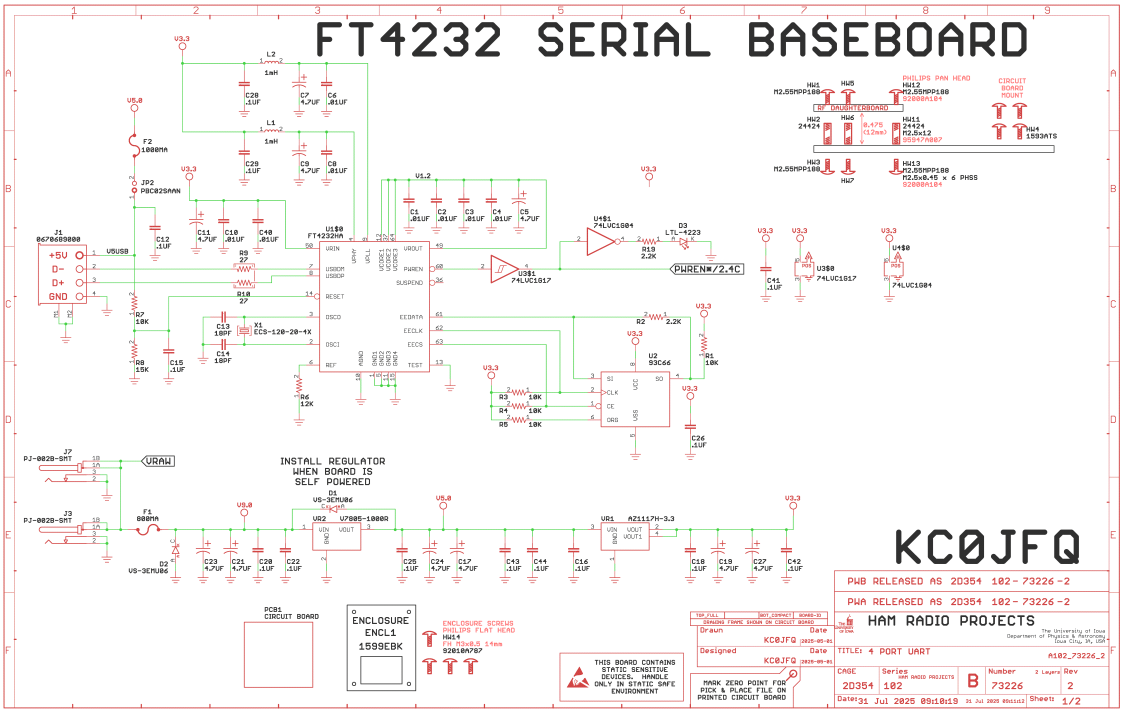


Figure 6.4: FTDIchip FT4232

FT4232 channel-3 is used to program the zNEO, A second channel (channel-2) is used to connect to the zNEO UART-1 through the 3.5mm port (command port). A third channel may be used to connect to the zNEO UART-0 for testing the daughter-board port.

The motherboard has 4 plug-in positions for daughter boards to provide the individual electrical interface.

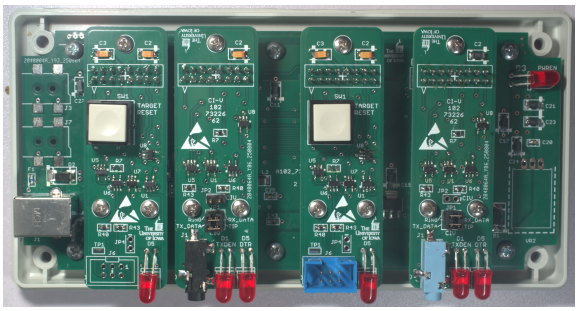


Figure 6.5: 4-Channel prototype

Almost any combination of draughtboard may be installed on the motherboard. The DE9 draughtboard are restricted to channel-0 and channel-1 due to their greater width.

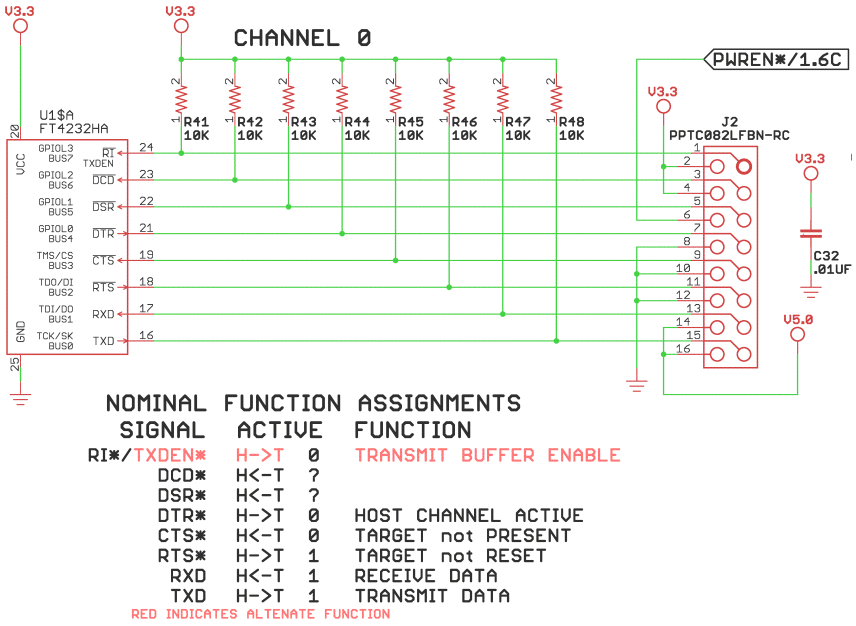


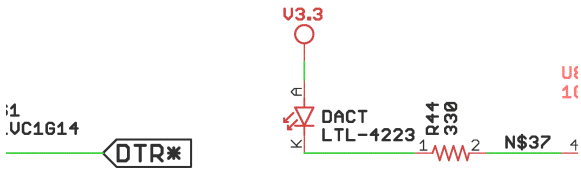
Figure 6.6: FTDIchip FT4232 channel

Each of the other three channels are wired identical to the first.

Shown here is the connections from the *FT4232* to the daughter boards along with the nominal functional assignments for the modem control lines.

The DTR net

Most of the daughterboards for the 4-port UART board use the **DTR** *output* net to drive a channel-active indicator LED.



373

Figure 6.7: DTR Indicator

You will note in the daughtercard schematics that the net is buffered to drive the LED. The **DTR** LED is illuminated when the host software establishes the connection to the USB-UART device. When the host software is done using the USB-UART device, it sets this net to an inactive state (LED off). Most of these software units also setup an exit handler to deactivate the DTR net when an error or other abort condition occurs.

Both of the DE9 channel cards route all of the modem signals to the 9-pin connector. This net, then, performs it normal function for these cases.

The CTS net

The **CTS** *input* net is used for a target detect function on the zNEO programming card.

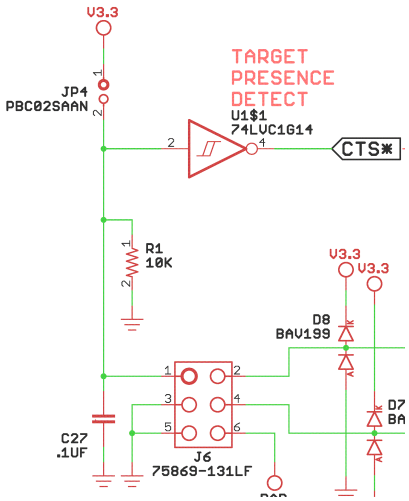


Figure 6.8: CTS Indicator

The target system is expected to tie pin-1 of the programming header to the local 3.3V supply. A pull-down resistor (R1) on the programming card and a Schmitt trigger inverting gate (U1) complete the detection circuit. The **CTS** net will be LO (i.e. active) when the target is plugged in.

Both of the DE9 channel cards route all of the modem signals to the 9-pin connector. This net, then, performs it normal function for these cases.

The RTS net

The **RTS** *output* net is used for a reset function on the zNEO programming card.

Before the USB-UART device is first used, the RTS net will be **inactive** (i.e. HI) which will holt the target zNEO in a reset state. Once the zNEO programming utility is run, the RTS net will be set to an active state to allow the zNEO to run. The RTS net is set to an inactive state for a few milliseconds when it is necessary to reset the target system. Further, the host software makes no effort to leave the RTS net in an active state (to allow the target to run) although this is generally what occurs. A USB disconnect/reset seems to put the RTS net into an inactive state which then holds the target in a reset state.

Both of the DE9 channel cards route all of the modem signals to the 9-pin connector. This net, then, performs it normal function for these cases.

halo_term V1.11

The *halo_term* utility has been updated to manage the DTR net for the 4-port UART boards. By convention we activate the DTR net (active **low**) when accessing the channel. Added in the V1.11 release is an exit handler to release the DTR net when the *halo_term* utility exits.

zNEO_P V1.3

The *zNEO_P* utility has also been updated to manage the DTR net for the 4-port UART boards. Same activation convention when accessing the channel. Same exit handler to release the DTR net when the *zNEO_P* utility exits.

6.3.1 Two Channel UART

The UART board may be built as a 2-channel variant by replacing U1, the FT4232HL with a FT2232HL. The pinouts on these devices match allowing a direct replacement.

Channel A (or channel 0) and C (or channel 2) carry the UART signals. Channel B (or channel 1) carries the TXDEN signal for the first channel (moved from the RI/TXDEN net). Channel D (or channel 3) has the same complement of signals for the second channel.

There are two FT2232 channel cards, one for programming the zNEO, and one for use with the CI-V port on the target Fox Transmitter. The CI-V channel card must be placed in the first position (i.e. on the right side of the mainboard). The zNEO programmer card must be placed in the second position (i.e. on the left side of the mainboard).

Two Channel UART connector change

The FT2232 device provides 16 signal pins for each channel running out through the same pins as on the FT4232.

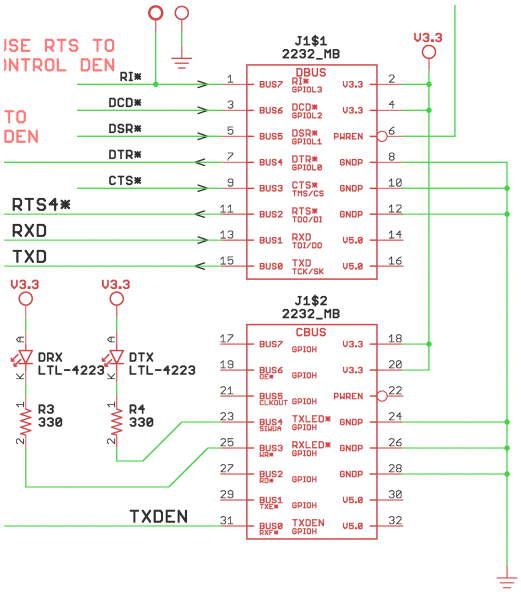


Figure 6.9: Two Channel UART connector change

As the channel cards are spaced unevenly, the FT2232 channel cards can only be placed on one pair of connectors.

You can easily match the connector pins on the *DBUS* connector to those used in figure 6.6 on page 6.6. The *CBUS* connector carries the **TXDEN** signal to drive the output buffer along with a couple of LED drivers. The *CBUS* connector also appears in the part to lock the connector positions.

These channel cards both incorporate a speed-up circuit that improves performance during transmit edge transitions. This feature ie enabled using a jumper on the card.

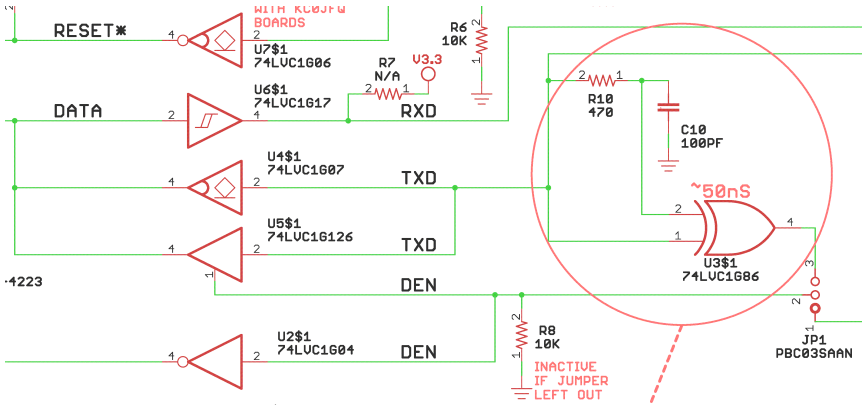


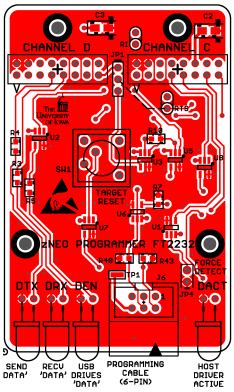
Figure 6.10: Two Channel UART connector CI-V

U3, **R10**, and **C10** form an edge detector. U3 emits a hi going pulse of approximately 500nS when the **TXD** net changes state. The output of **U3** can be then be used to drive the enable pin on **U5**. **U4** holds the **DATA** net low while a pull-up resistor holds the **DATA** net hi. This causes **U5** to drive the **DATA** net for less than a bit time during transitions.

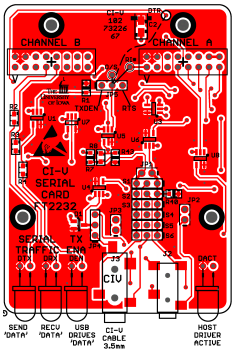
The feature jumper, **JP1** in figure 6.10, is installed in the 1-2 position to use the **TXDEN** signal from the FT2232, in the 2-3 position to use the local speed-up feature. Finally, the jumper may be left out to use the board in a *default* configuration.

For the zNEO card, this uses **only** the open-drain gate to drive the DATA net. For the CI-V card, the 74LVC1G126 is enabled for use as a full duplex channel.

6.3.2 Two Channel board images



(a) Two Channel zNEO



(b) Two Channel 3.5mm

Note the dual (8x2) connectors with different center-to-center spacing. These are implemented using the same logic as the boards used with the FT4232. The connector spacing forces them to be correctly installed.

The center spacers, needless to say, must not be installed on the mainboard when using these channel cards.

6.3.3 UART 3.5mm Channel Card

This is the logic-level UART interface used by the Fox Transmitter.

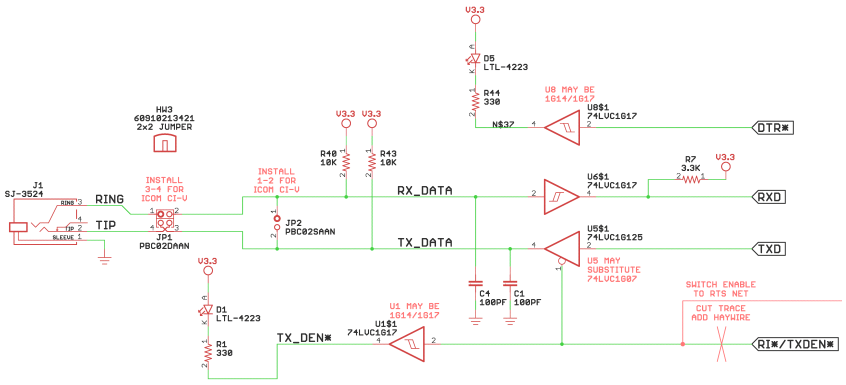


Figure 6.12: 3.5mm serial channel

This is used to connect to the primary 3.5mm connector on the Fox Transmitter to load FRAM and FLASH. It may also used to connect to the other serial channel (on the main board) through the 102-73181-24 daughter-board.

As populated in the above schematic, this board may be jumpered to operate as an ICOM CI-V bus node. Install JP1/3-4 (labeled "TIP/TX_DATA") and JP2/1-2 (labeled "CIV") for operating on the CI-V bus which uses only tip and sleeve on 3.5mm connector.

This card can also be populated to emulate the FTDIchip *TTL-232R-3V3-AJ* cable by installing all ICs as 74LVG1G17. U5 and U6 are then Schmitt trigger non-inverting buffers.

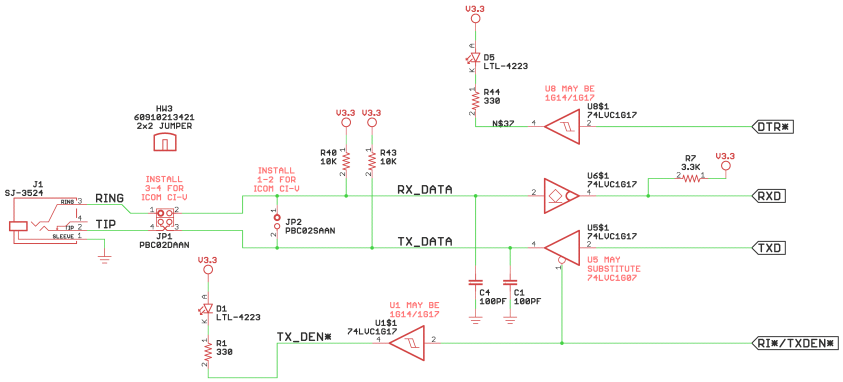


Figure 6.13: FTDIchip TTL-232R-3V3-AJ emulation

Note the changes in the part values shown on this schematic.

This, in effect, eliminates the RS485-like feature of the **TX_DATA** net by populating all ICs with 74LVC1G17. Therefore the **TX_DATA** net is always driven, no longer depending on the RI*/TX_DEN signal from the FT4232 USB UART on the main board.

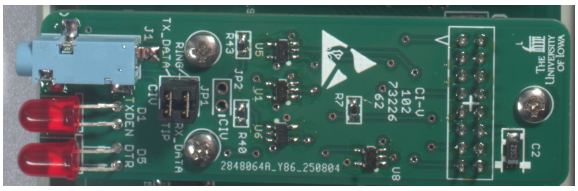


Figure 6.14: 3.5mm CIV card

This is the logic-level UART interface for programming radios using 3.5mm/2.5mm audio cables.

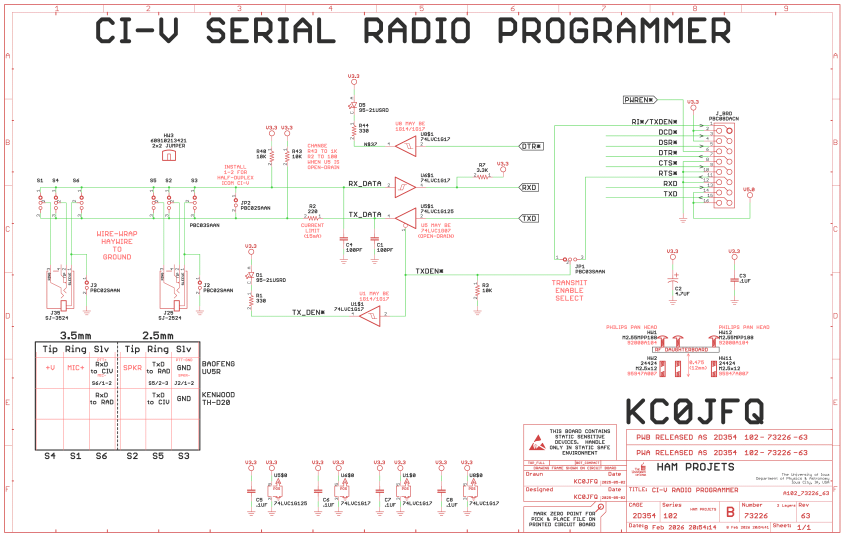


Figure 6.15: 3.5mm/2.5mm serial channel

The connector pinout is set using the jumper array S1..S6 and JP2/JP3

This card may be configured as a CI-V interface in much the same manner as the 102-72336-62 card.

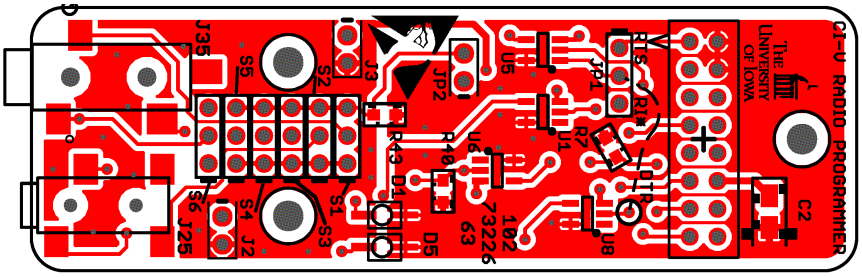


Figure 6.16: Radio Programmer

6.3.4 ZiLOG zNEO Programming Channel Card

This simply duplicates the programming channel from the eZ8 Programming Adapter in section 6.2 on page 31.

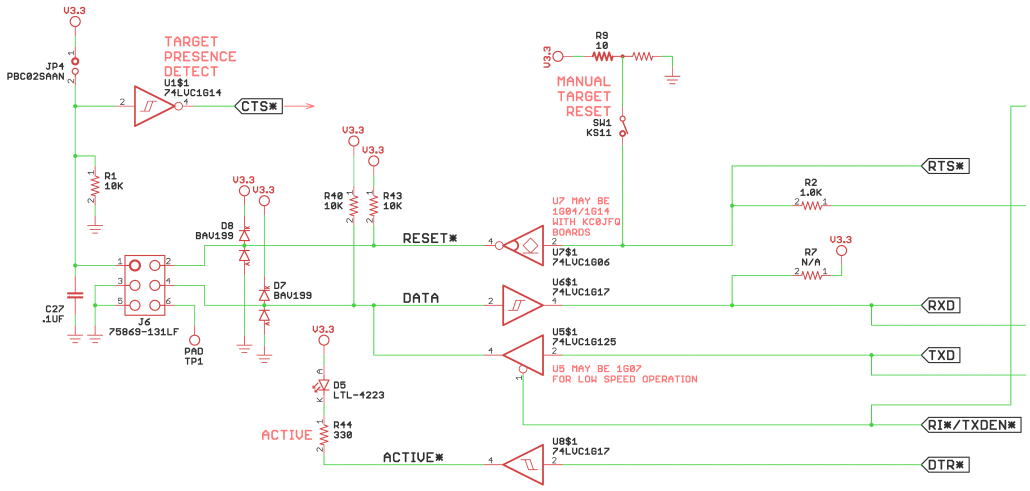


Figure 6.17: eZ8/zNEO Programming schematic

The *recovery oscillator* (seen in section 6.2) is removed owing to space constraints.

This interface should behave identically to the 102-73220-32/102-73220-33 eZ8 Programmer above.

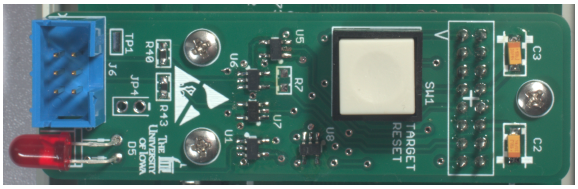


Figure 6.18: eZ8/zNEO Programming card

6.3.5 Additional Channel Cards

There exist designs for additional cards that are not used by the Fox Transmitter system.

An isolated CI-V network card.

An isolated RS485 half-duplex network card.

A pair of RS232 cards with DE9 connectors. These cards may be placed in positions 3 and 4 due to their slightly larger width. (one male and one female).

UART 3.5mm Isolated Channel Card

This logic-level UART interface is also available with isolation.

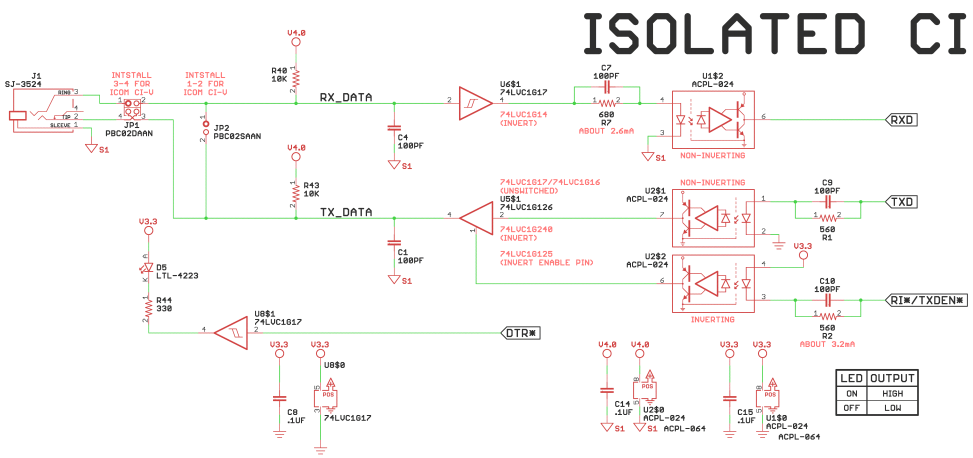


Figure 6.19: 3.5mm serial channel, isolated

This provides electrical isolation between the host computer (through the USB cable) and the Fox Transmitter.

Optical isolators are inserted between the motherboard UART and the level shifters. The indicated opto-isolator device has very low drive requirements to reduce the current load presented by the LED in U1. The ACPL-024 is a high speed isolator that should have no difficulty running at the 115,200 b/S rate used for fast loading the FRAM/FLASH memory.

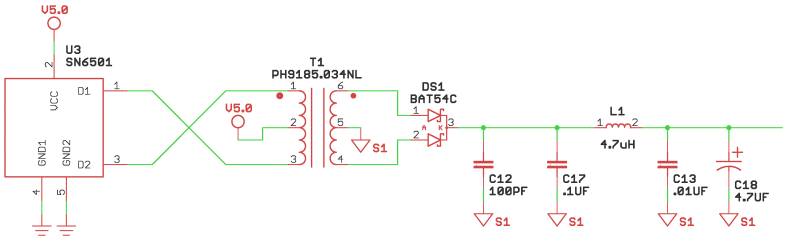


Figure 6.20: Isolated 5V supply

The isolated power is supplied using a SN6501 driver and a Wurth isolation transformer. This power converter runs off of the 5V rail (the 5V rail has a higher current capability than the 3.3V rail).

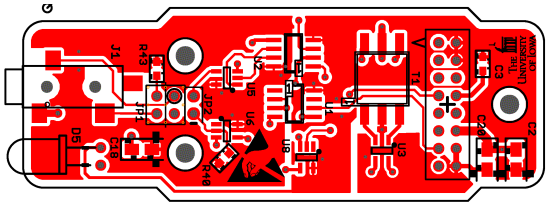


Figure 6.21: isolated 3.5mm artwork

RS485 Opto Channel Card

Basic RS485 half-duplex network card (isolated). Same function as 102-73220-23 and 102-73220-20. Software driver compatible with only the data pair and ground connected (no power).

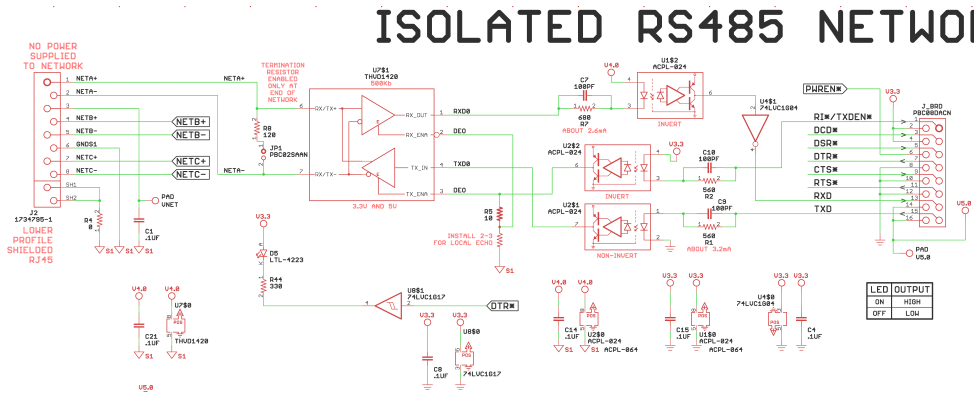


Figure 6.22: RS485 Opto Channel Schematic

When the network is idle or disconnected, U1 idles with the LED off. This requires that the output be inverted (input is inverted).

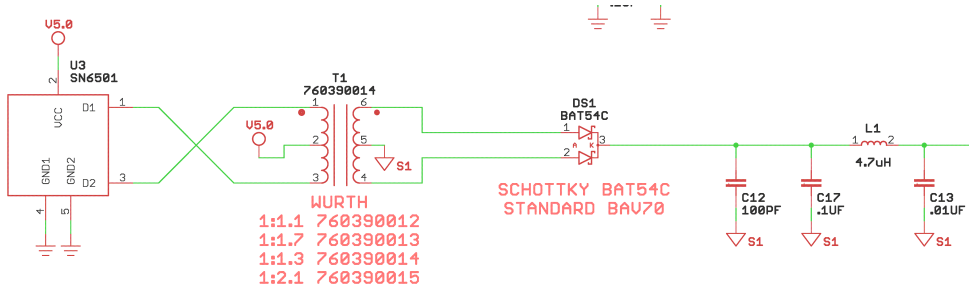


Figure 6.23: RS485 Opto 5V power

We use the same secondary side power design here.

Part value on the transformer reflects suggested parts from the SN6501 datasheet.

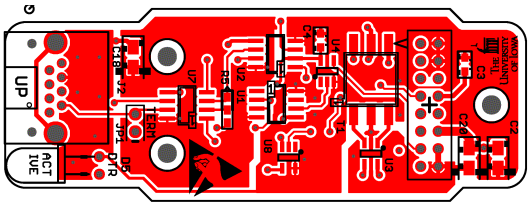


Figure 6.24: RS485 Opto Channel Card artwork

DE9F Channel Card

Standard D connector (E shell size). Pinout for female connector.

Jumper array allow reassigning the connector pinout.

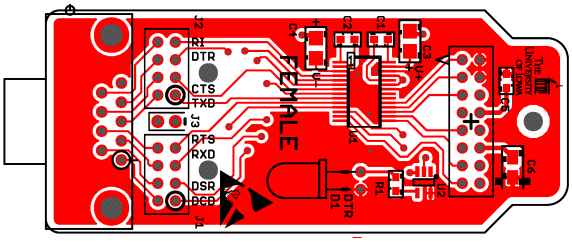


Figure 6.25: DE9F Channel Card artwork

DE9M Channel Card

Standard D connector (E shell size). Pinout for male connector.

Jumper array allow reassigning the connector pinout.

When jumpered straight through, this matches the standard PC serial port connector.

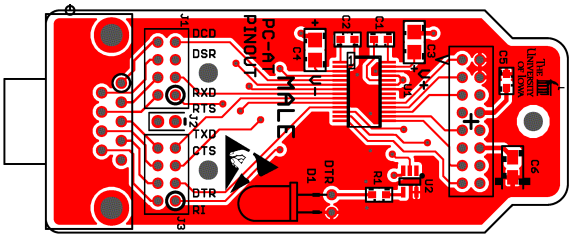


Figure 6.26: DE9M Channel Card artwork

6.4 FTDIchip EEPROM programming

All of the sub-projects presented here make use of FTDIchip USB-UARTS. The EEPROM typically will need to be loaded with configuration information specified to the application where the chip is used.

We will first use the *ftdi_eeprom* utility under Linux to extract the current data in the EEPROM, saving it to a *.conf* file. This *.conf* file can then be modified to suit your needs.

We can then use the *ftdi_eeprom* utility under Linux to program the EEPROM in the FTDI UART with new data.

Programming using *ftdi_sio* utility

The **ftdi_sio** module conflicts with the EEPROM programming utility and must be removed.

Kill any programs that are currently using an FTDIchip device. Remove (i.e. unplug) all FTDIchip devices to release any locks on the **ftdi_sio** module. Plug the target device into the USB bus (it will be the only device present) and verify:

```
lsusb | grep -i "Future Technology Devices International"
lsmod | grep sio
```

With the new board plugged ito the USB bus we can remove the **ftdi_sio** module and verify success.

```
sudo rmmod ftdi_sio
lsmod | grep sio
```

The system is now prepared for prograqmming the new device. Our example reference the *EZ8PGM.conf* configuration file on page 53.

First verify we can access the device by reading the current EEPROM contents:

```
sudo \
    ftdi_eeprom \
    --verbose \
    --device i:0x0403:0x6015 \
    --read-eeprom \
    EZ8PGM.conf
```

Assuming we can successfully read the EEPROM, go ahead and program the new setup:

```
sudo \
    ftdi_eeprom \
    --verbose \
    --device i:0x0403:0x6015 \
    --flash-eeprom \
    EZ8PGM.conf
```

If all went well, you can disconnect the new device from the USB bus and replug it and inspect the results:

```
sudo dmesg
```

The **dmesg** log should show the device with the newly assigned manufacturer, product, and serial fields.

We can also look at the detail of the device using this command:

```
lsusb -v -d0403:
```

As we just loaded the EEPROM, our newly programmed device should be only FTDI device present, so we tell *lsusb* to dump (-v is the verbose flag) all FTDI devices (the -d0403: flag).

Vendor_ID and **Product_ID**

Always use the values provided in the device before it is programmed. We use the standard serial drivers so there is no good reason to change these values.

Serial Numbers

Serial numbers **must** be unique. There is a numbering convention used by the author to serialize all FTDI devices. In the examples below, the last 4 characters of the serial number are sequential across all devices produced by the author. The initial characters serve to identify the type of device and are not detailed here.

Max Power

All of these device should fit comfortably within the power constraints imposed by the USB standard.

You may consult the data-sheets to estimate the maximum power that should be required by the various projects.

6.4.1 ftdi_eeprom keytable

Configurable string values.

Decode Macro	Key Name	Description
CFG_STR	manufacturer	String we provide
CFG_STR	product	String we provide
CFG_STR	serial	String we provide (unique!), *see use_serial*
CFG_STR	user_data_file	
CFG_STR	filename	EEPROM image dump file

Configurable integer values.

Decode Macro	Key Name	Description
CFG_INT	vendor_id	**DO NOT ALTER**
CFG_INT	product_id	**DO NOT ALTER**
CFG_INT	default_pid	**DO NOT ALTER**
CFG_INT	max_power	calculate this value
CFG_INT	eeprom_type	0x66(93C66) 0x56(93C56) 0x46(93C46)
CFG_INT	release_number	
CFG_INT	usb_version	**DO NOT ALTER**
CFG_INT	user_data_addr	for large eeproms

Most of the devices supplied by FTDIchip have pins that can be configured by the EEPROM to route useful signals to these pins.

Consult the datasheet to determine which of the *cbus* groups are supported by the device. You can then configure them as needed. Reading the existing EEPROM usually will give you clues as to which *cbus* group to choose from.

6.4.2 ftdi_eeprom CBUS items

In the way of example, the FT230**X** device lists only 4 cbus pins, so pick from the *cbus**x*** group. The FT232R has 5 *cbus* pins so here we use the *cbus* group. And finally, the FT232H, with 10 *cbus* pins, will use the *cbus**h*** group.

Alternate features for legacy FT232R devices (USB port on the Fox Transmitter).

Decode Macro	Key Name	Description
CFG_INT_CB	cbus0	
CFG_INT_CB	cbus1	
CFG_INT_CB	cbus2	
CFG_INT_CB	cbus3	
CFG_INT_CB	cbus4	

CBUS Options	Description
TXDEN	Output Driver Enable (RS485)
PWREN	USB configured, USB suspend: high
TXLED	TX activity status LED
RXLED	RX activity status LED
TXRXLED	TX & RX activity status LED
SLEEP	USB suspend
CLK48	48 MHz clock
CLK24	24 MHz clock
CLK12	12 MHz clock
CLK6	6 MHz clock
IOMODE	IO port for CBUS bit bang mode
BB_WR	Synchronous Bit Bang Write strobe (FT232R and FT-X only)
BB_RD	Synchronous Bit Bang Read strobe (FT232R and FT-X only)

Alternate features for FT232H/FT2232H/FT4232H devices (used on the 102-73226 boards).

Decode Macro	Key Name	Description
CFG_INT_CB	cbush0	
CFG_INT_CB	cbush1	
CFG_INT_CB	cbush2	
CFG_INT_CB	cbush3	
CFG_INT_CB	cbush4	
CFG_INT_CB	cbush5	
CFG_INT_CB	cbush6	
CFG_INT_CB	cbush7	
CFG_INT_CB	cbush8	
CFG_INT_CB	cbush9	

CBUS-H Options	Description
TRISTATE	IO Pad is tri-stated
TXLED	TX activity status LED
RXLED	RX activity status LED
TXRXLED	TX & RX activity status LED
PWREN	USB configured, USB suspend: high
SLEEP	USB suspend
DRIVE_0	Drive constant 0 (FT232H and FT-X only)
DRIVE1	Drive constant 1 (FT232H and FT-X only)
IOMODE	IO port for CBUS bit bang mode
TXDEN	Output Driver Enable (RS485)
CLK30	30 MHz clock
CLK15	15 MHz clock
CLK7_5	7.5 MHz clock

Alternate features for FT230X devices (used on the 102-73220-20 and 102-73220-23 boards).

Decode Macro	Key Name	Description
CFG_INT_CB	cbusx0	
CFG_INT_CB	cbusx1	
CFG_INT_CB	cbusx2	
CFG_INT_CB	cbusx3	

CBUS-X Options	Description
TRISTATE	IO Pad is tri-stated
TXLED	TX activity status LED
RXLED	RX activity status LED
TXRXLED	TX & RX activity status LED
PWREN	USB configured, USB suspend: high
SLEEP	USB suspend
DRIVE_0	Drive constant 0 (FT232H and FT-X only)
DRIVE1	Drive constant 1 (FT232H and FT-X only)
IOMODE	IO port for CBUS bit bang mode
TXDEN	Output Driver Enable (RS485)
CLK24	24 MHz clock
CLK12	12 MHz clock
CLK6	6 MHz clock
BAT_DETECT	Battery Charger Detect (FT-X only)
BAT_DETECT_NEG	Inverse signal of BAT_DETECT (FT-X only)
I2C_TXE	Transmit buffer empty (FT-X only)
I2C_RXF	Receive buffer full (FT-X only)
VBUS_SENSE	Detect when VBUS is present via the appropriate AC IO pad (FT-X only)
BB_WR	Synchronous Bit Bang Write strobe (FT232R and FT-X only)
BB_RD	Synchronous Bit Bang Read strobe (FT232R and FT-X only)
TIME_STAMP	Toggle signal each time a USB SOF is received (FT-X only)
AWAKE	Do not suspend when unplugged/disconnect/suspend (FT-X only)

6.4.3 ftdi_eeprom channel type

Configure channel type.

Decode Macro	Key Name	Description
CFG_INT_CB	cha_type	
CFG_INT_CB	chb_type	

Channel Type	Description
UART	
FIFO	
OPTO	
CPU	
FT1284	

6.4.4 ftdi_eeprom pin drive

Configure drive strength.

Decode Macro	Key Name	Description
CFG_INT_CB	group0_drive	

Group-0 Drive	Description
4MA	
8MA	
12MA	
16MA	

6.4.5 ftdi_eeprom feature flags

Configurable Feature Enable Flags.

Decode Macro	Key Name	Description
CFG_BOOL	change_usb_version	
CFG_BOOL	flash_raw	
CFG_BOOL	high_current	
CFG_BOOL	in_is_isochronous	
CFG_BOOL	out_is_isochronous	
CFG_BOOL	remote_wakeup	
CFG_BOOL	self_powered	*true* powered external to USB *false* powered by USB
CFG_BOOL	suspend_pull_downs	
CFG_BOOL	use_serial	*true* to use the serial number string

Configurable Feature Enable Flags.

Decode Macro	Key Name	Description
CFG_BOOL	cha_rs485	**TRUE** for TXDEN*
CFG_BOOL	chb_rs485	**TRUE** for TXDEN*
CFG_BOOL	chc_rs485	**TRUE** for TXDEN*
CFG_BOOL	chd_rs485	**TRUE** for TXDEN*
CFG_BOOL	cha_vcp	**TRUE** for *virtual com-port driver*
CFG_BOOL	chb_vcp	**TRUE** for *virtual com-port driver*
CFG_BOOL	chc_vcp	**TRUE** for *virtual com-port driver*
CFG_BOOL	chd_vcp	**TRUE** for *virtual com-port driver*

Configurable pin polarity flags.

Decode Macro	Key Name	Description
CFG_BOOL	invert_cts	
CFG_BOOL	invert_dcd	
CFG_BOOL	invert_dsr	
CFG_BOOL	invert_dtr	
CFG_BOOL	invert_ri	
CFG_BOOL	invert_rts	
CFG_BOOL	invert_rxd	
CFG_BOOL	invert_txd	

6.4.6 EEPROM, FOX17.conf

Configuration for the FOX17 Fox Transmitter. This is 102-73181-0 hardware where the only command path is through the on-board FT232 UART.

```
vendor_id=0x403
product_id=0x6001
max_power=24
manufacturer="KC0JFQ"
product="KC0JFQ_FOX_V3"
serial="2078-0-0114"
filename="FOX17.bin"
self_powered=false
use_serial=true
```

The *self_powered=false* line indicates that the device is accepting 5V power from the USB bus. The USB-UART on the Fox Transmitter is powered through the USB bus so it remains active when the Fox Transmitter is not powered. This keeps the USB devices from dropping off the bus when power is removed.

6.4.7 EEPROM, EZ8PGM.conf

Configuration for the zNEO programmer.

```
vendor_id=0x403
product_id=0x6015
max_power=400
manufacturer="KC0JFQ"
product="EZ8PGM"
serial="42B3-0-0118"
filename="EZ8PGM.bin"
self_powered=false
use_serial=true
#
cbusx0="TXDEN"
cbusx1="IOMODE"
cbusx2="RXLED"
cbusx3="TXLED"
```

The zNEO programmer is intended to obtain power from the USB bus as the power requirements for the programmer are minimal.

We do advertise a higher power level than we actually use.

6.4.8 EEPROM, prototype

Configuration for the some other prototype.

```
vendor_id=0x403
product_id=??
max_power=??
manufacturer="KC0JFQ"
product="produce name"
serial="serial number"
filename="prototype.bin"
self_powered=false
use_serial=true
```

Set the *self_powered=* line as needed.

Set the *serial=* line to a unique value so it can be cleanly discovered and not conflict.

6.4.9 EEPROM, radio 25.conf

Configuration for the KC0JFQ radio interface. This project uses the FT4232 device.

```
filename=radio_25.bin
vendor_id=0x0403
product_id=0x6011
eeprom_type=0x66
manufacturer="Perf Proc"
product="KC0JFQ Radio I/F"
serial="2078-0-0025"
use_serial=true
max_power=0
self_powered=true
remote_wakeup=false
cha_type=UART
chb_type=UART
cha_rs485=false
chb_rs485=false
chc_rs485=false
chd_rs485=false
cha_vcp=true
chb_vcp=true
chc_vcp=true
chd_vcp=true
```

The USB-UART is powered externally (not by the USB bus) so the *serial=* reflects this use.

6.4.10 EEPROM, 4-port UART

Configuration for the 4-port UART is a bit more complicated in that the 4 channels all use daughter-boards to hold the physical interface.

Because the daughter-board configuration is somewhat dynamic all that will be provided here is a base configuration for one zNEO programming channel and one 3.5mm channel.

Channel 0 This channel is not discussed here.

Currently, it is available for other projects.

Channel 1 This channel is not discussed here.

Currently, it is available for other projects.

Channel 2, Fox Transmitter Configuration (3.5mm)

The 102-73226-62 card is installed in this position.

The daughter card is configured with JP1 having both jumpers installed to allow communications with the Fox Transmitter. JP2 is left open (we are operating in a full-duplex mode).

Channel 3, zNEO Programmer

The 102-73226-42 card is installed in this position.

We set **cha_rs485=true** to allow the FT4232 to drive the **TXDEN*** net when transmitting.

1653

```
vendor_id=0x0403
product_id=0x6011
manufacturer="KC0JFQ"
product="PGM4UART"
serial="2170-0-0118"
use_serial=true
eeprom_type=0x66
filename="PGM4.bin"
max_power=100
self_powered=false
remote_wakeup=false
cha_type=UART
chb_type=UART
cha_rs485=false
chb_rs485=false
chc_rs485=false
chd_rs485=false
cha_vcp=true
chb_vcp=true
chc_vcp=true
chd_vcp=true
```

6.4.11 EEPROM,

Configuration for the .

```
ver
```

testing