Nial Stewart Developments Ltd
FPGA Development
High Speed Digital Design

HSMC General Purpose Interface Board (GPIB)

User Guide

1. Overview

The General Purpose Interface Board (GPIB) has been designed to expand the interface capability of Altera development boards with the HSMC connector, specifically the CIII starter kit.

The board provides the following interfaces….

8 * 10 bit ADC Inputs
8 * 10 bit DAC Outputs
2 * General Purpose 8 bit Digital IO ports
3 * RS232 Interfaces
2 * RS484 Interfaces
3 * One Wire Interfaces (2 of these can be combined for 5V I2C communications, all 3 can be used as jumper selection inputs)
Micro SD card interface & socket (SPI lines connected)
Header for an FTDI Vinculum VDIP1 or pin compatible NSD FT245R USB slave board, this can also be used as a 16 bit general purpose digital IO header.
8 * LEDs
2 * SPI Interfaces

The Vinculum socket and SD card socket provide two options for bulk data storage. The use of our USB slave board which is pin compatible with the Vinculum allows the CIII & FPGA to be driven as a slave USB device.

The SPI headers easily allow simple PCBs to be developed to add further functionality such as….

EEPROM storage
LCD Screen Output
Zigbee Interface
GPS modules
CAN Interface
LIN Interface
Stepper Motor Driver
Inter-board control & status information transfer.
Multiple Digital IO control

Each interface is described with description of the **Physical Interface,** IO connector and the FPGA pins connected.

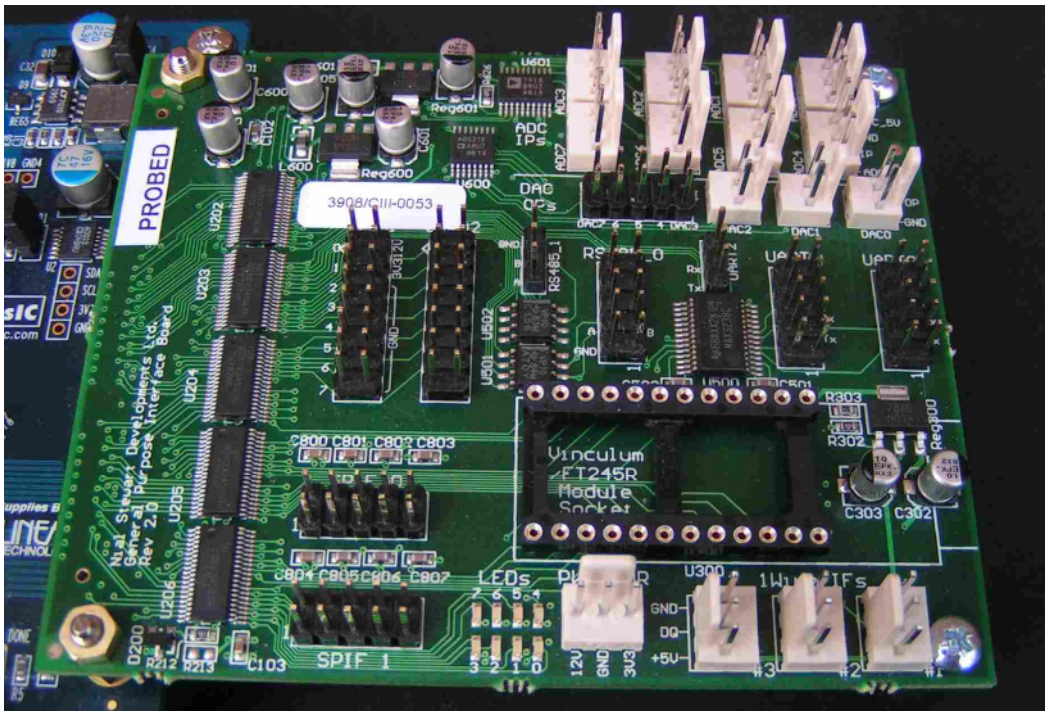
A brief description of the VHDL **Interface Module** in the example FPGA design provided that drives it. The internal register interface to each module is described.

The example FPGA design registers are controlled/monitored by the UART/local bus driver that is connected to RS232 channel 0.

The memory map for the example FPGA design is given in Appendix A.

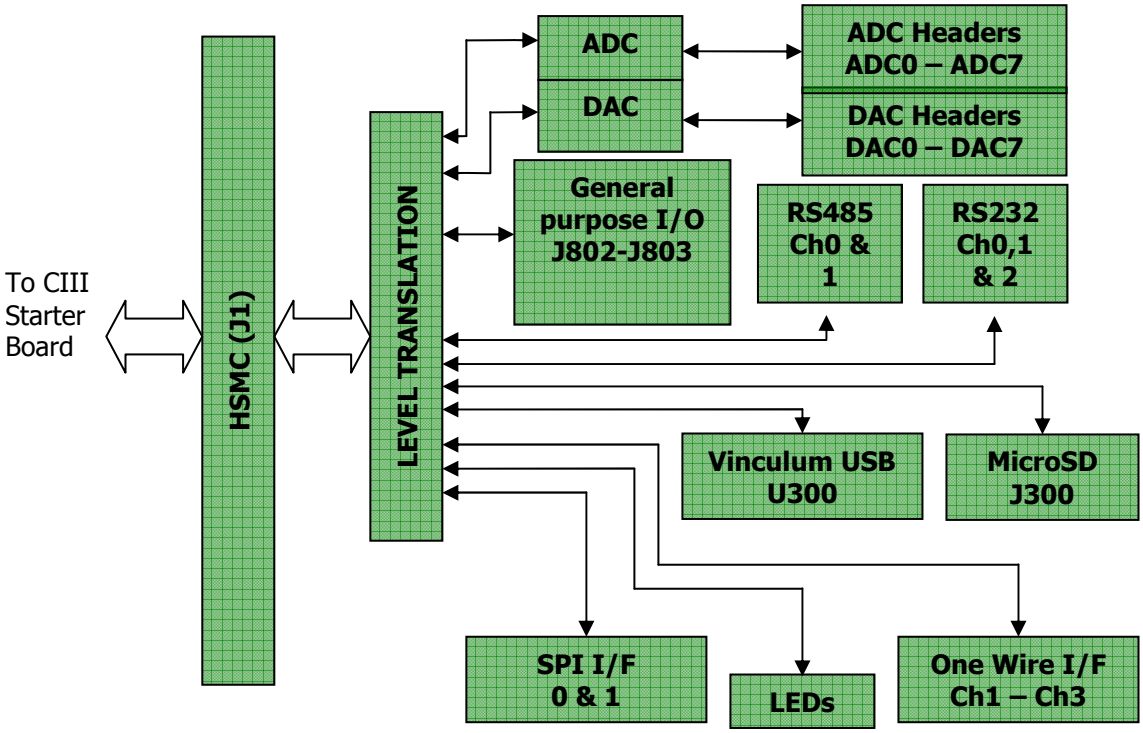
The FPGA – GPIB device connectivity is provided with the included "Rev2_HSMC_Pin_Allocation.xls" spreadsheet.

The GPIB Board



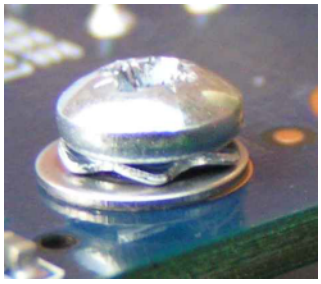
Board Diagram

The following is a top level block diagram of the GPIB.



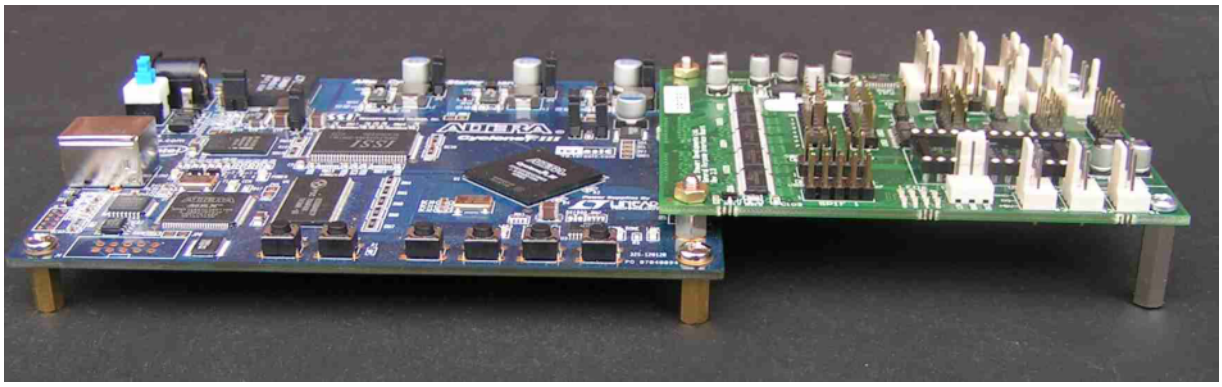
2. Installation

The GPIB board should fit firmly onto the CIII starter kit HSMC connector, the HSMC connector provides a board separation of 5mm. To help maintain a good connection and mechanically support the interface please install the 5mm standoffs that are supplied with the boards. These should be secured with the 6mm M3 screw and both the anti-vibration and plain washers that are supplied as shown below, this stops the M3 screws bottoming out in the 5mm standoffs.



To support the other end of the GPIB please install the 17mm standoffs provided with the other two M6 screws provided. The length of these is chosen to match the standoffs provided with the CIII starter board so these must be fitted too.

A picture of the two boards secured together is shown here.



3. Interfaces

3.1 3 * RS232 Interfaces

Physical Interface

A MAX3238 buffer is used to drive the three RS232 ports, this provides the level shifting required for the RS232 interface.

Note: Testing shows that the MAX3238 produces line voltages of ± 6V (as spec'd), typical of laptop/more recent device outputs. This might not be sufficient with **very** old legacy devices.

The pin allocation on the 10 pin channel 0 and 1 connectors allow an easy connection to be made with an IDC 10 way socket and an IDC 9 way D type connector with 'standard' RS232 signal assignment as shown here.

Connectors

Connector	Pin	Function
UART0	3	Ch0 Tx
	5	Ch0 Rx
	9	Gnd
UART1	3	Ch1 Tx
	5	Ch1 Rx
	9	Gnd
UART2	3	Ch2 Tx
	2	Ch2 Rx
	1	Gnd

FPGA Pin allocations

FPGA Connection	FPGA Pin
Ch0 Tx	M3
Ch0 Rx	F3
Ch1 Tx	L6
Ch1 Rx	H6
Ch2 Tx	D3
Ch2 Rx	N11

Example Driver Module

A combined UART and local bus controller module, nsd_uart.vhd is provided. This has three main processes, low level 'uart' Tx and Rx processes XXXX check these two, tx fifo? for receiving and transmitting characters and a higher level Master Process. The Master process interprets the received characters as instructions and data and drives a 'local bus' interface to a set of registers.

The first character received defines the address and whether it's a read or write (bit 7 = '1' for a read), the next two are write data for a write transaction, these are then driven on the local bus. If a read is being requested the local bus is read at the defined address and the results returned as two characters.

3.2 ADC Inputs

Physical Interface

Eight 10 bit analogue inputs are provided with a separate 3 pin header per input. An AD7918 is used, a 2.5V reference is provided so the input range is selectable between 0->2.5V or 0->5V. This device is configurable to convert a single channel or sequence of channels automatically.

Each header provides a 5V rail (Pin 3) and GND (Pin 1) with the ADC input on Pin2 for easy connection of potentiometers etc, the pins are clearly labelled on the PCB silk screen.

A series 330Ω resistor and clamping diodes to +5V and GND are used on each input to provide some protection on the input but care should be taken not to subject the device to extreme voltages.

With a clock rate of 10MHz the AD7918 can convert all eight channels at approximately 60K conversions per second.

Connectors ADC0 – ADC7

Pin	Function
1	Gnd
2	Input
3	ADC +5V Rail

These are labelled on the silk screen.

FPGA Pin allocations

FPGA Connection	FPGA Pin
ADC1_CS#	A1
ADC1_DIN	E1
ADC1_DOUT	N10
ADC1_SCLK	J13

Driver Module

The example application configures the inputs for 0->5V operation with a continuous conversion sequence of all eight channels.

This module consists of two main processes, a Master and Slave. The Master controls what's being sent or received by the Slave process, the Slave process blindly transfers data to and from the ADC on a 'do_transfer' tick from the Master process and generates a 'rx_tx_fin' flag to show data is available.

Out of reset the Master sends to sets of '1's to the ADC to allow it to come out of reset, then the control sequence to set up a sequence is sent. The input range is set by module input, **ip_range**, the last channel to be converted in the sequence is a module intput, **seq_end**, in the example app this is set to "111" so the sequence runs through all intput channels.

After this the control bit is set to zero and every transfer receives the latest conversion result, the Master process assigns this to the correct output port. Once every set of conversions the 'outputs_valid' flag is asserted to show a set of conversions has been performed.

With a system clock of 80MHz the ADC clock is driven at 10MHz giving a conversion period of about 2.1uS, or a set of conversions every 16.75 ms.

3.3 DAC Outputs

Physical Interface

An AD5318 is used to provide 8 * 10 bit analogue outputs, three on separate 2 pin headers labelled DAC0 -> DAC2 , 5 on a headers labelled DAC 7 – DAC 3. Each output has an associated ground pin, 470Ω series resistors are provided on each channel to provide some protection against shorted outputs. Again care should be exercised to ensure the outputs aren't overly loaded.

Note: The AD5318 was chosen for its combination of output resolution/speed/channel number/ price. However the 'A' version fitted (ARUZ) has a ± 4 bit INL, ie the output can be ± 15 output values from it's theoretical ideal for a given output value. Please see the AD5318 data sheet for more details.

Connectors DAC0 – DAC2

Pin	Function
1	Output
2	Gnd

Connector DAC3 – DAC7

Pin	Function
1	DAC 3
2	Gnd
3	DAC 4
4	Gnd
5	DAC 5
6	Gnd
7	DAC 6
8	Gnd
9	DAC 7
10	Gnd

These are labelled on the silk screen.

FPGA Pin allocations

FPGA Connection	FPGA Pin
DAC1_DIN	T1
DAC1_LDAC#	N8
DAC1_SCLK	M5
DAC1_SYNCH#	N7

Driver Module – AD5318_if.vhd

This is a simple module that drives the serial interface to the DAC when the input data word, **dac_wr_data**, is written to indicated by the **dac_wr** flag. These are driven from the UART interface.

dac_wr_data takes the following format…
(15) - '0' = Data write, '1' = Control word write
(14..12) - Ch address
(11..2) - 10 bit output value.

This allows complete control of the configuration and output of the DAC by writing to **dac_wr_data**

3.4 General Purpose Digital IO

Physical Interface

Two 8 bit general purpose digital IO channels are provided. These are driven straight from the FPGA with just the protection of the IDT Quickswitches on the inputs.

Connectors GPIO1 & GPIO2

Pin	Function
1	GPIOX_0
2	+12V
3	GPIOX_1
4	+3.3V
5	GPIOX_2
6	Gnd
7	GPIOX_3
8	Gnd
9	GPIOX_4
10	Gnd
11	GPIOX_5
12	Gnd
13	GPIOX_6
14	Gnd
15	GPIOX_7
16	Gnd

These are labelled on the silk screen

FPGA Pin allocations

FPGA Connection	FPGA Pin	FPGA Connection	FPGA Pin
GPIO1(0)	K1	GPIO2(0)	H15
GPIO1(1)	H16	GPIO2(1)	N16
GPIO1(2)	N15	GPIO2(2)	R16
GPIO1(3)	T16	GPIO2(3)	C2
GPIO1(4)	C1	GPIO2(4)	H2
GPIO1(5)	H1	GPIO2(5)	K5
GPIO1(6)	L5	GPIO2(6)	L3
GPIO1(7)	L4	GPIO2(7)	L2

Driver Module

The two GPIO ports each have a set of output, input and direction registers. Each bit in the direction register that is set to '1' sets the associated pin to an output controlled by the relevant bit in the output register. The example design defaults the pins to inputs.

The input reads the current state of the pins.

No driver module is used as such, the output settings and direction control is implemented at the top level of the design.

3.5 2 * RS484 Interfaces

Physical Interface
Two Ti SNHVD10 transceivers are used to drive the RS485 interfaces. These are capable of driving 30mbps however the quickswitches & pull ups used to provide the level shifting and protection as described in Appendix XXXX may limit the data rates. Testing has shown that transmission and reception between the two devices over a 100m CAT5 twisted pair is reliable at 10Mbps, faster reliable transmission is obtainable, but untested.
As with the RS232 connections, the 10 pin header allows an easy connection to the RS485 signals via an IDC socket.

Connector RS485_0

Pin	Function
1	
2	Gnd
3	IO_N
4	IO_P
5	
6	
7	
8	Gnd
9	
10	

Connector RS485_1

Pin	Function
1	IO_P
2	IO_N
3	Gnd

FPGA Pin allocations

FPGA Connection	FPGA Pin
RS485_0_RX	B1
RS485_0_RXEN#	G2
RS485_0_TX	K2
RS485_0_TXEN	G1
RS485_1_RX	B2
RS485_1_RXEN#	P11
RS485_1_TX	T2
RS485_1_TXEN	K17

Driver Module

The example driver module drives both RS485 ports. The control inputs to the module are direction and enable, when enable is asserted the test is started. A byte is sent on one on the Tx channel (direction = '0' for Ch0 transmission '1' for Ch1) then the module waits until a byte is received, this is compared to the transmitted byte. The transmitted byte is then incremented and the process repeats. If the received byte does not match the transmitted byte a fail flag is set. This can be reset by setting the enable intput to '0' then back on to '1' again.

The test continuously runs when enable is asserted, the transmission speed is 1mbps.

3.6 3 * One Wire Interfaces

Two of these can be combined for 5V I2C communications.
These can also be used as selection jumper inputs.

Physical Interface
Three one wire interface headers are provided, these provide gnd, DQ and 5V for one wire compatible sensors/inputs. The DQ lines can be driven by the FPGA and have 2.2K pull-up resistors to 5V. One wire signalling is achieved by driving the line low to signal a '0' and tri-stating the output to signal a '1'.

All one wire devices (that we are aware of) are capable of accepting a 5V interface.

Connectors 1 Wire Ifs #1,#2 & #3

Pin	Function
1	+5V
2	DQ
3	Gnd

FPGA Pin allocations

FPGA Connection	FPGA Pin
OneWire_1_DQ	R2
OneWire_2_DQ	R1
OneWire_3_DQ	P2

Driver Module
This drives the interface to three DS1821 temperature sensors via the one wire interface/protocol. A Master module generates all the timing and control signals to drive the interfaces. Three simple slave devices drive the IO and sample inputs when flagged to by the master process. This keeps the logic 'footprint' for each device small to allow a lot of devices to be driven with a small amount of logic.
The logic of the master module has been reduced to the minimum needed to drive the DS1821, it comprises a master and low level process. The low level process drives an 'initialisation' (that must be performed before each data transfer), byte write or byte read under the control of the master process. The master process contains the sequence required to start a conversion and read the results. If more features of the DS1821 were to be used only the master process need be modified.

Alterative use.
I2C Interface
Two of these outputs could be combined to provide a 5V I2C interface. With 1.7m of 3 wire ribbon cable to a DS1821 the rise time of the signal is ~ 400ns. With no load the rise time is ~ 100ns. This is sufficiently fast to drive the fastest 400Khz I2C devices at full speed.

Jumper Selection Inputs
As the centre DQ line is pulled high to 5V these can be used as a jumper selection inputs by placing a jumper between DQ and the GND pin.

3.7 Micro SD card interface (SPI lines connected)

Physical Interface
The FPGA pins are routed direct to a Micro SD card socket on the bottom of the board. Only sufficient pins are routed to allow SPI mode control of the card.

Connector/FPGA Pin allocations

Connector Pin	Connection	FPGA Pin
5	SD_CLK	C14
3	SD_CMD	D14
7	SD_DAT0	M2
2	SD_DAT3	L1
4	+3.3V	
6	Gnd	

Driver Module
When triggered the driver module simply initialises the SD card to SPI mode, this checks that the SD card is present, behaving as expected and that the physical interface to and from the SD card is verified. A flag is set at the output to indicate that the card has initialised OK.

Writing to and reading from SD cards is not a trivial exercise, especially if the FAT32 file structure is adhered to.

We may develop and provide the IP to do this if the demand exists but this is beyond the scope of this example application.

3.8 2 * SPI Interfaces

Physical Interface
The SPI interfaces have been designed to allow fast serial connection down ribbon cable (of reasonable length) and to allow extra functionality to be added to the GPIB.

SN74LVC1G17 3.3V schmitt trigger input drivers are used to drive the interface outputs. These have 75 Ω source termination resistors positioned between the drivers and the output connector to provide a source termination for transmission lines of ~100 Ω. The data and clock signals are routed to the header with alternating ground pins, this gives a line impedance of ~100 Ω down 'standard' 0.1" pitch ribbon cable. This allows coherent transmission of signals to remote boards, this has been tested with ribbon cable lengths of 7m at 1Mbps.

The 'input' signal is simply routed straight to the FPGA via the quickswitches. This should be source terminated (to provide ~100 Ω) on any remote device that is connected.

Two chip selects are provided to allow two remote devices to be addressed per interface. These can also be used as generic digital IO.

These headers allow a great expansion in functionality, boards with the following could be connected…

EEPROM
LCD Screen
Zigbee Interface
GPS Interface
CAN Interface
LIN Interface
Stepper Motor Driver
Serial Control/Status interfaces with other hardware
Bulk Digital IO control (driven off a small CPLD)

Connectors SPIF_0 & SPIF1 & FPGA Pin allocations

Pin	Function	FPGA Pins IF0	FPGA Pins IF1
1	+3.3V		
2	SPI_CS1	K18	R17
3	Gnd		
4	SPI_Serial_Data_Out	G17	L15
5	Gnd		
6	SPI_Serial_Data_In	E17	R18
7	Gnd		
8	SPI_Clk	G18	L13
9	Gnd		
10	SPI_CS2	E18	M14

Note: Although these nets are labelled 'CS1', 'CLK' etc they are all identical outputs and can be used for any purpose, Serial Data In is the only input.

Driver Module

The SPI_IF module simply actions a transfer of data in both directions when data is written to the Tx port or read from the Rx port. Data is transmitted on the XXXX edge, read on the XXXX edge, this can be easily changed in the module code.

3.9 Header for an FTDI Vinculum VDIP1 or pin compatable NSD USB slave board.

Physical Interface

This interface is basically a routing of 16 generic FPGA IO lines to the 0.1" 24 pin header.

+5V is provided on pin1, GND on pins 7 and 18, this allows the connection of the FTDI VDIP1 Vinculum VNC1L Development Module. This is a USB host controller….

http://www.vinculum.com/prd_vdip1.html

…which allows USB flash drives to be easily used for data storage or retrieval.

We have designed a slave USB interface around the FT245R with the same footprint as the VDPI1 to allow the GPIB/CIII to be driven as a USB slave device. This is available as an add-on for the GPIB and allows the local bus to be driven via USB rather than the serial interface.

Connector/FPGA Pin allocations

Connector Pin	Connection	FPGA Pin
1	5V	
2	N/C	
3	N/C	
4	N/C	
5	N/C	
6	USB1_AD0	M17
7	Gnd	
8	USB1_AD1	L16
9	USB1_AD2	L18
10	USB1_AD3	R3
11	USB1_AD4	T3
12	USB1_AD5	P1
13	USB1_AD6	M6
14	USB1_AD7	N6
15	USB1_AC0	M13
16	USB1_AC1	N13
17	USB1_AC2	P17
18	Gnd	
19	USB1_AC3	L14
20	USB1_AC4	M18
21	USB1_AC5	L17
22	USB1_RS#	H18
23	USB1_PC#	H17
24	N/C	

Slave Driver Module

The slave USB interface module provides the similar functionality as the RS232 uart, this allows internal registers to be driven via a USB link. An example Excel application can be downloaded to demonstrate how this interface is easily driven.

Like the UART interface a control and address byte is received first, then write data if a write is being driven. If a read is actioned the returned data is written back upstream to the FT245R. The local bus provides address, data_out, data_in write and read flags.

3.10 8 * LEDs

The LED outputs are 8 FPGA lines routed to the cathode of 8 leds pulled to 3.3V via 110R resistors.

FPGA Pin allocations

LED	FPGA Pin
LED(0)	V18
LED(1)	U18
LED(2)	T18
LED(3)	T17
LED(4)	M1
LED(5)	P18
LED(6)	R5
LED(7)	R4

4. APPENDIX 1 – Memory Map

The memory map of the example FPGA application is given here.

Address	Description	Read / Write	Bit Definitions
0x00	Dac Data	Wr	(15) – '0' = Data . '1' = Control Word (14 .. 12) = Ch Address (11 .. 2) = 10 bit data value
0x01	ADC Ch0 Value	Rd	(9 ..0) = Channel Value
0x02	ADC Ch1 Value	Rd	(9 ..0) = Channel Value
0x03	ADC Ch2 Value	Rd	(9 ..0) = Channel Value
0x04	ADC Ch3 Value	Rd	(9 ..0) = Channel Value
0x05	ADC Ch4 Value	Rd	(9 ..0) = Channel Value
0x06	ADC Ch5 Value	Rd	(9 ..0) = Channel Value
0x07	ADC Ch6 Value	Rd	(9 ..0) = Channel Value
0x08	ADC Ch7 Value	Rd	(9 ..0) = Channel Value
0x0A	RS485 Control	Wr	(1) = Direction (0) = Enable
		Rd	(0) – '1' = Fail, '0' = Running without error
0x10	Temp Sensor 1	Rd	(7 .. 0) = Temp
0x11	Temp Sensor 2	Rd	(7 .. 0) = Temp
0x12	Temp Sensor 3	Rd	(7 .. 0) = Temp
0x14	SD Interface	Wr	Any value write = trigger SD card initialisation
		Rd	(15 .. 8) = Number of times the initialisation has been sucessful (0) - '1' = Initialisation finished
0x16	GPIO 1 Data	Wr	(7 .. 0) – Data to the GPIO port
		Rd	(7 .. 0) – GPIO Data in
0x17	GPIO 1 Direction	Wr	(7 .. 0) – GPIO port directions. '1' = Output
0x18	GPIO 2 Data	Wr	(7 .. 0) – Data to the GPIO port
		Rd	(7 .. 0) – GPIO Data in
0x19	GPIO 2 Direction	Wr	(7 .. 0) – GPIO port directions. '1' = Output
0x20	Led Out	Wr	(7..0) – LED output
0x21	Led Control	Wr	(1..0) – "10" – LED Output Value "01" – All alternating flash "00" – Incrementing Count (Default)
	SPI IF 1		Not defined Yet
	SPI IF 2		Not defined Yet
0x7f	Test Reg	Wr/Rd	(15..0) – Test register
Others	Default Reg	Wr/Rd	(15 ..0) – Default register if access is to no defined address above.

5. Appendix 2 – Pin allocations

A complete list of FPGA pin allocations is given here.

FPGA/GPIB Net	EP3C25 Pin No
ADC1_CS#	A1
ADC1_DIN	E1
ADC1_DOUT	N10
ADC1_SCLK	J13
DAC1_DIN	T1
DAC1_LDAC#	N8
DAC1_SCLK	M5
DAC1_SYNCH#	N7
GPIO1(0)	K1
GPIO1(1)	H16
GPIO1(2)	N15
GPIO1(3)	T16
GPIO1(4)	C1
GPIO1(5)	H1
GPIO1(6)	L5
GPIO1(7)	L4
GPIO2(0)	H15
GPIO2(1)	N16
GPIO2(2)	R16
GPIO2(3)	C2
GPIO2(4)	H2
GPIO2(5)	K5
GPIO2(6)	L3
GPIO2(7)	L2
LED(0)	V18
LED(1)	U18
LED(2)	T18
LED(3)	T17
LED(4)	M1
LED(5)	P18
LED(6)	R5
LED(7)	R4
OneWire_1_DQ	R2
OneWire_2_DQ	R1
OneWire_3_DQ	P2
RS232_0_RX	F3
RS232_0_TX	M3
RS232_1_RX	H6
RS232_1_TX	L6
RS232_2_RX	N11
RS232_2_TX	D3

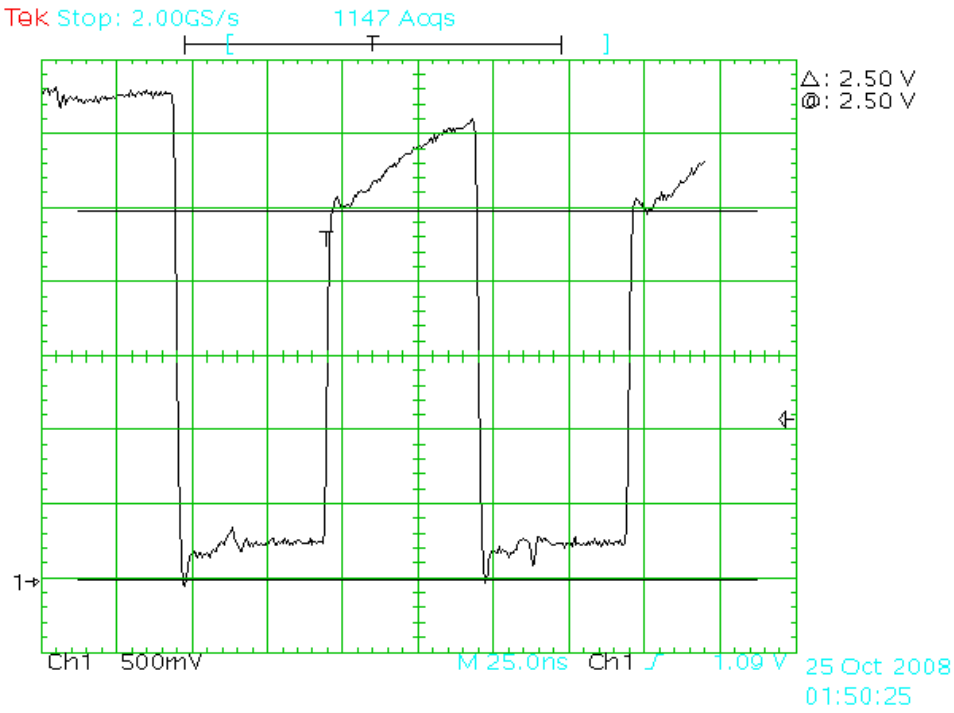
RS485_0_RX	B1
RS485_0_RXEN#	G2
RS485_0_TX	K2
RS485_0_TXEN	G1
RS485_1_RX	B2
RS485_1_RXEN#	P11
RS485_1_TX	T2
RS485_1_TXEN	K17
SD_CLK	C14
SD_CMD	D14
SD_DAT0	M2
SD_DAT3	L1
SPI_IF_0_CS_N_1	K18
SPI_IF_0_CS_N_2	E18
SPI_IF_0_SCLK	G18
SPI_IF_0_SDI	E17
SPI_IF_0_SDO	G17
SPI_IF_1_CS_N_1	R17
SPI_IF_1_CS_N_2	M14
SPI_IF_1_SCLK	L13
SPI_IF_1_SDI	R18
SPI_IF_1_SDO	L15
USB1_AC0	M13
USB1_AC1	N13
USB1_AC2	P17
USB1_AC3	L14
USB1_AC4	M18
USB1_AC5	L17
USB1_AD0	M17
USB1_AD1	L16
USB1_AD2	L18
USB1_AD3	R3
USB1_AD4	T3
USB1_AD5	P1
USB1_AD6	M6
USB1_AD7	N6
USB1_PG#	H17
USB1_RS#	H18

6. Appendix 3 - FPGA Protection / Level shifting

The FPGA on the CIII evaluation board is configured to drive the HSMC interface at 2.5V, all of the logic devices on the GPIB operate at 3.3V. In order to provide level shifting and protection for the 2.5V FPGA pins a series of IDT QS32X245 quickswitches are used with a reference voltage of 3.3V. This clamps the signal transferred from one side to the other to ~2.6V (well within the maximum level for 2V5 signalling) with the signals on the GPIB being pulled to 3.3V or 5V as appropriate.

The FPGA can only drive outputs to 2.5V, this 'clamped' by the quickswitch near the top end of this range but line inductance causes slight overshoot. The 3.3V signals on the GPIB rely on pull-ups to bring the level up to 3.3V, this provides a slower rise time to 3.3V.

This can be seen on the scope cature below.



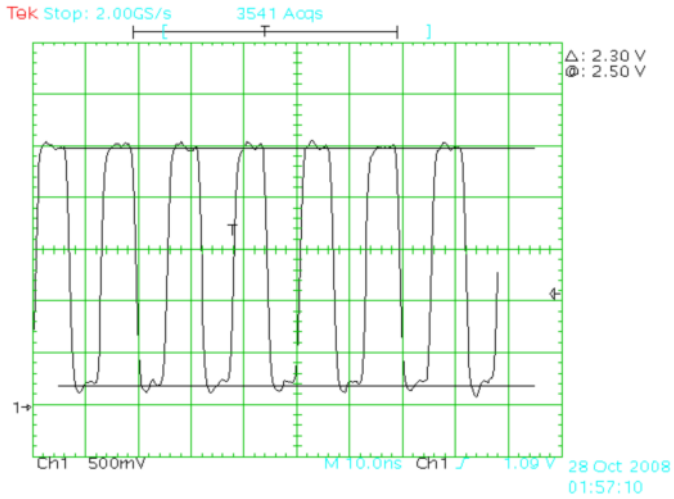
This is a scope capture of the ADC clock which is driven at 10MHz, the cursors are at 0V and 2.5V.

This leads to the possibility of glitches on inputs with logic thresholds at about 2.5V. The interfaces on the GPIB have been checked for this risk (ie the VIhigh for the DAC is 1.7V). Care should be taken with any external logic that is connected to the GPIO or USB interface connectors.

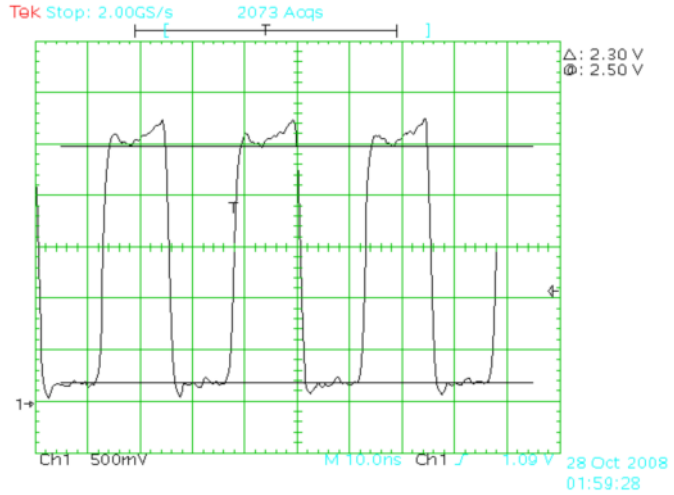
The SPI interfaces have 3.3V driver devices with hysteresis on their inputs so this concern does not apply.

The scope captures overleaf show different output rates and the resultant waveforms, the timebase was 10ns per division for all three.

1) GPIO Pin toggled at 80MHz. Note the cursors are at 200mV and 2.5V



2) GPIO Pin toggled at 40MHz. Again the cursors are at 200mV and 2.5V.



3) SPI DOut toggled at 80MHz. Cursors at 0V and 3.2V. This was taken with no load on the output.