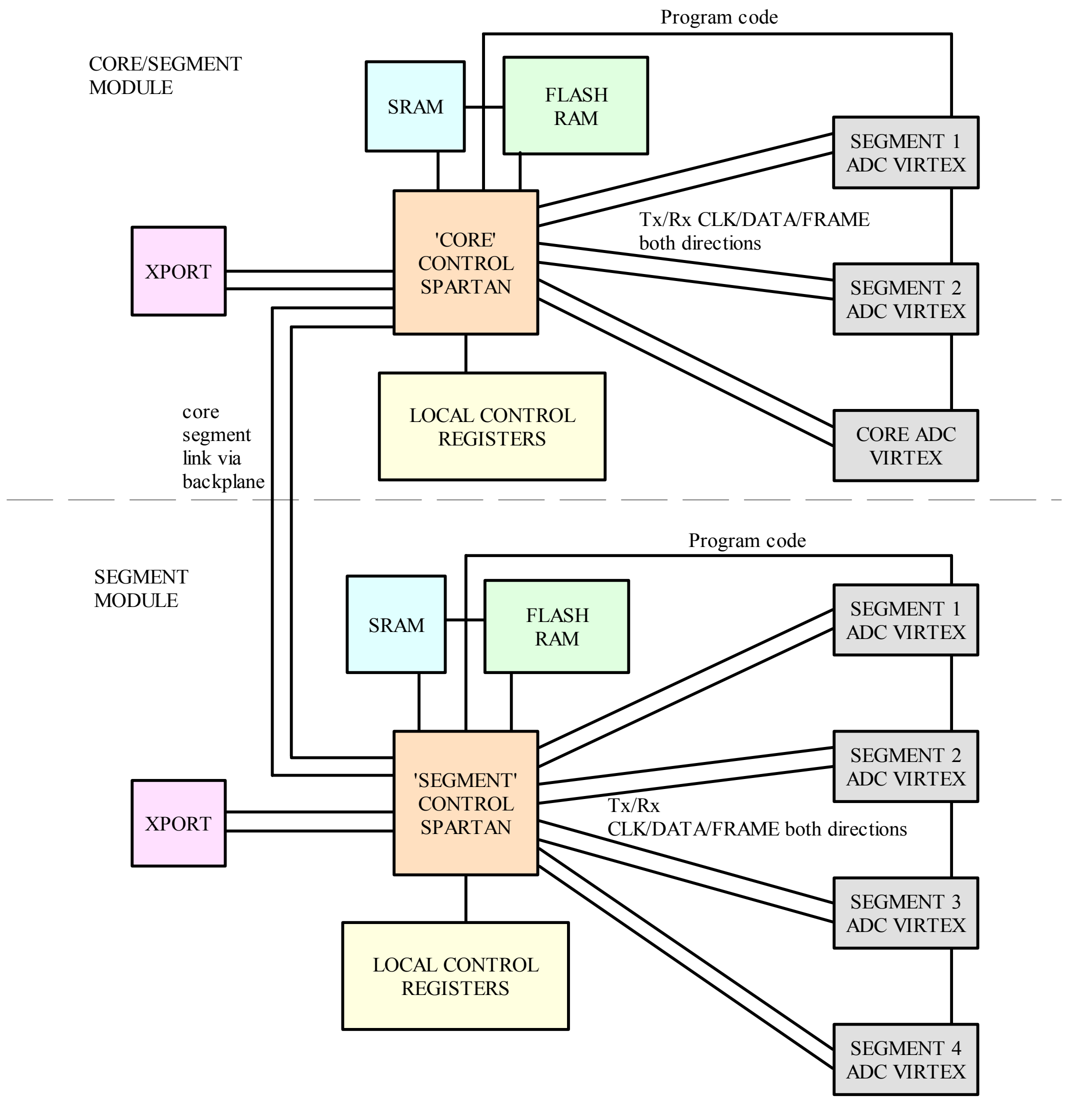
AGATA Digitiser control bus Overview.



The diagram to the left shows an overview of the communications within a single AGATA Digitiser box. The box consists of two Modules as follows:-

- Core/Segment Module which contains the Core electronics and the 2 cards with 7 segment channels per card (6+1 spare each card). This gives the Core and 12 usable Segment channels in this module.
- Segment Module which contains 4 PCBs each with 7 channels of segment ADC (6+1 spare).This gives 24 usable Segment ADC channels.

The communication between all the devices internal to each module and also between modules is serial with 5 wires in each direction. 2 differential CLOCK, 2 differential DATA and a single FRAME. Details of these are discussed later. The FRAME signal has two functions. 1) To show the start and end of a data transfer. Only data and clocks within a FRAME will be accepted. 2) To act as a handshake line from Rx to Tx during data transfers to prevent buffer overruns at the receiving end.
These features are discussed further in this document. All communication is done through the input XPORT device from the 10/100baseT Ethernet link. Neither module can initiate an Ethernet communication, they can only respond to an incoming request.

AGATA Digitiser control and status data format.

Basic command types

There are 3 allowed command types. These are:-

- *Simple Write* commands
- *Long Write* commands
- *Simple Read* commands

All commands operate on one module (either Core/segment module (module 1) or Segment only module (module 2)) at a time. i.e. Many simple commands can be sent as one command stream but they all must be for one of the 2 modules.

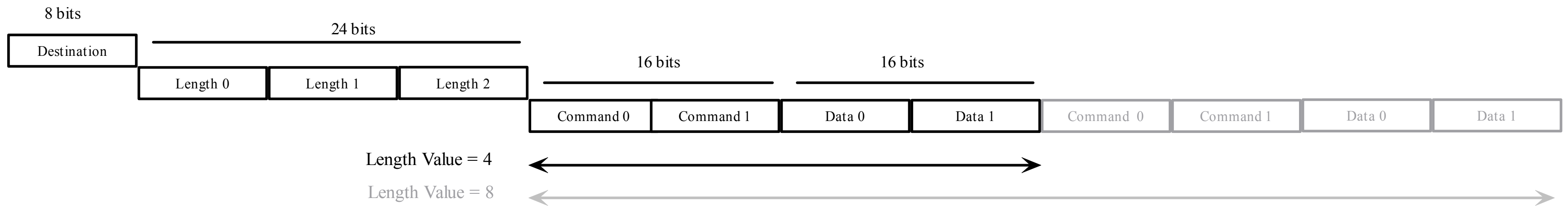
The Long Write command is used for loading the virtex EEPROM device or may be used for similar future applications, e.g. lookup tables etc. within a virtex device. By virtue of the Long Write command these cannot be concatenated.

The Simple Write commands are for setting individual registers for e.g. ADC enable, pulser values, gain range etc. commands can be concatenated within a module.

The Simple Read commands are single commands that request information from one of the modules. This may be temperature values, status or virtex histogram data etc. These commands cannot be concatenated.

The overall structure of the three command types is discussed below.

Command structure



The command structure shown above is a typical Simple Write command of one command (and showing how a second command would concatenate). All boundaries in the system are byte based.

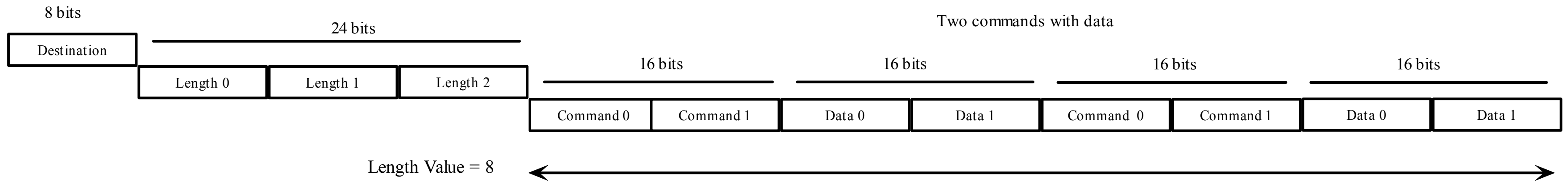
Breaking down the structure above we get:-

- Destination – This defines the Module (1 or 2) that the string is for. Whether it is a Read or Write command and, if it is a Write whether it is a Simple or Long type. The set of bits that define this are shown later.
- Length defines the total number of bytes in the stream that follows the last Length byte. This is the same for all Write and Read commands. The length value must always be a multiple of 2 when a EEPROM data load is requested (internally data is treated as16 bits).
- Command 0 and Command 1 form the 16 bit command. The bits pre-defined in this are shown later. Some bits echo those found in the Destination byte.
- Data 0 and Data 1 form the 16 bit data value that relate to the command bytes. NOTE :- In a Long Write command this will be a continuous byte data stream (Data 0……Data N). For a Long Write the number of bytes of data are defined by the value received in the Length bytes minus 2 for the control bytes. This will be shown later.

In the next section all types of command will be shown.

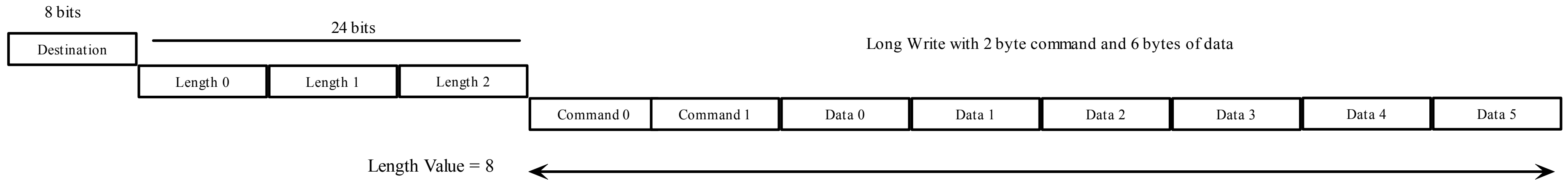
WRITE COMMANDS

Simple Write command



The Simple write command shown above will execute 2 commands. A command and associated data will always be 32 bits (as 4 bytes).

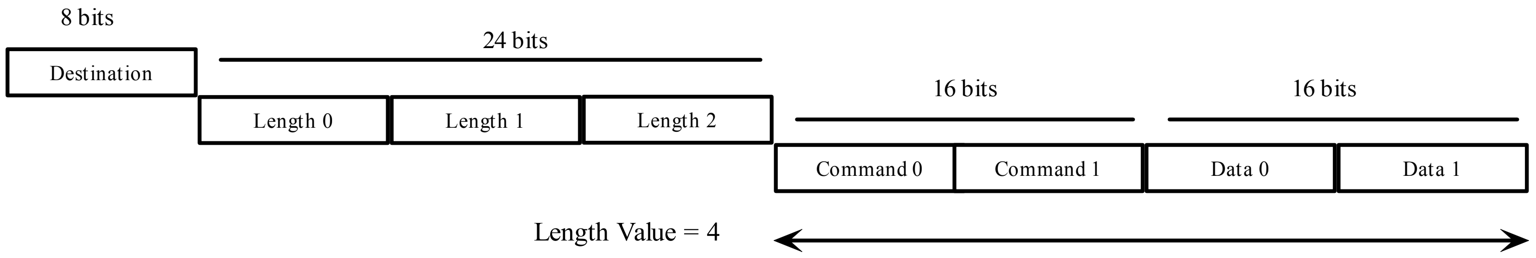
Long Write Command



The Long Write command shown above is an example containing one command and 6 bytes of data. Normally this command would be used for loading the EEPROM Virtex data and would be considerably longer than 8 bytes in length. Although not a current requirement, this type of command may also be used for loading lookup tables etc. into the Virtex devices, Control Spartans or similar.

READ COMMAND

Simple Read

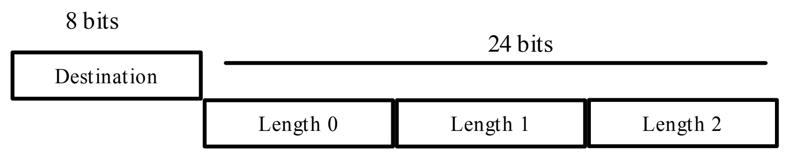


Read commands are only done singularly and are as shown above. The Command is again 16 bits and defines the read destination, e.g. read Histogram when in test mode. The Data is normally zero but could be used as a means of transferring qualifier bits to the command. For example bits could be set to indicate which temperature sensors are to be read , which of two histograms should be read out or if a ADC scope trace was collected the length to be read. In normal experimental use the data bits are zero but they may be useful in the test modes.

RESPONSE TO COMMANDS

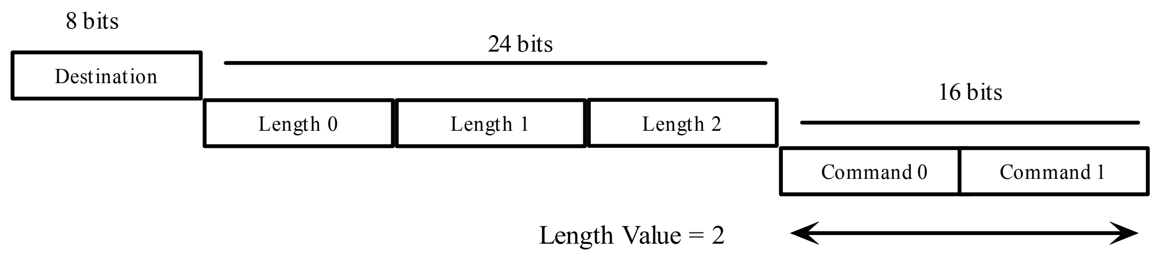
For every command, be it read or write, an acknowledge (ACK) will be generated and sent to the external controller. The ACK will indicate a successful Write, a failed Write and a Failed Read by the form it takes as follows.

Successful Write ACK



The Destination byte is the initial header sent from the controller echoed back unchanged. The Length value will be zero for a good Write sequence with no failures.

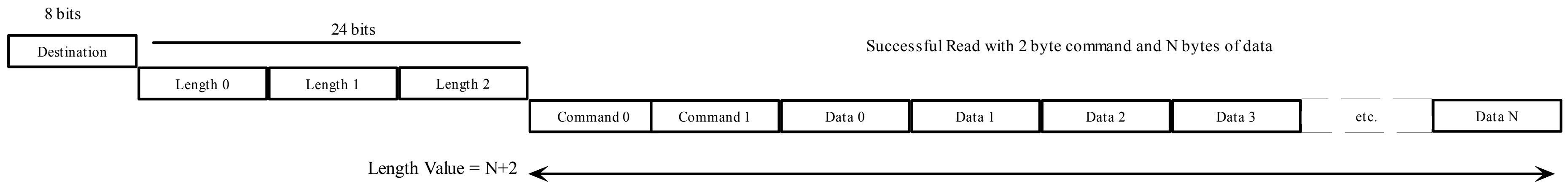
Failed Write ACK



The Destination byte is the initial header sent from the controller echoed back unchanged. The Length value will be 2 for a failed Write with the Command on which the failure occurred in the Command 0 and Command 1 bytes. This will allow easy debugging of controller software.

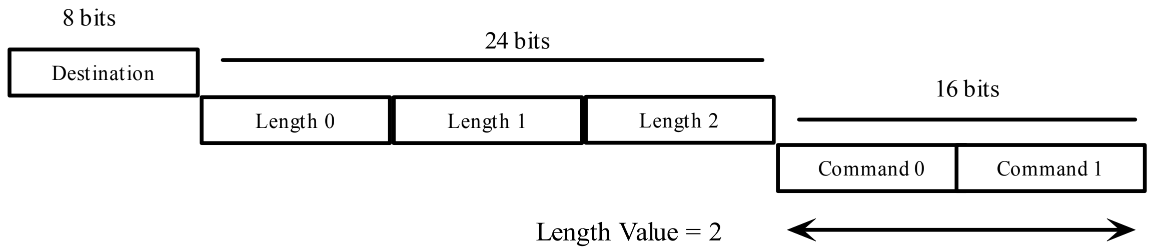
Successful Read ACK

When a successful read is done the controlling machine will receive the expected data stream it has requested in the following format.



Failed Read ACK

When a read fails the ACK sent back to the controlling machine is shown below.



In this case the received stream is the same as that for a failed Write command. The Command bytes will show what the command was that was received by the Digitiser unit and will then have no data following.

NOTE:- It may be possible that data will be sent but due to an unexpected error occurring within the digitiser, the data is truncated before the full length is reached. This will force a timeout within the Digitiser unit and the unit will be ready for further commands approximately 30 seconds or less (to be determined) after the failure was detected. The controlling machine must be able to cope with this eventuality.

FORMAT OF BYTES WITHIN DATA STREAM

To make the following easier to understand it is important to visualise the physical structure of the Digitiser unit. The digitiser unit is split into two modules, these are :-

- MODULE 1 = CORE channels with 12(+2) SEGMENT channels. (This is designated as the *CORE MODULE*).
- MODULE 2 = 24(+4) SEGMENT ONLY channels. (This is designated as the *SEGMENT MODULE*).

Destination Byte

MSB							LSB
CORE /SEG	RD /WR	NW /LW	0	0	0	0	0

Bit 7 (MSB)	selects the *physical Digitiser module* either the Core (=0) or Segment (=1) module as the destination for the command(s).
Bit 6	indicates if the command stream is a Read or Write. (Read =1, Write =0)
Bit 5	indicates that if a Write stream if a Simple Write or a Long Write. (Normal Write = 0, Long Write =1)
Bits 4 to 0	These are set to 0 (reserved for future use).

Length Bytes

See change note at end of document.

Length 0	MSByte
Length 1	---
Length 2	LSByte

These contain a 24 bit binary number which indicates the length of the following data stream.

Command bytes

Command Byte 0

MSB LSB

CORE /SEG	RD /WR	NW /LW	SM2	SM1	SM0	0	0

There are 6 bits used within this command byte. The top 3 bits echo exactly those of the Destination byte at the start of the data stream. The next 3 bits labelled SM0, SM1 and SM2 are address bits to indicate the destination within the module of the following command and data. These bits are designated as follows:-

CORE

SM2	SM1	SM0	Destination Item
0	0	0	Virtex for Segment ADC card 1
0	0	1	Virtex for Segment ADC card 2
0	1	0	Virtex for Core ADCs
0	1	1	Main board
1	0	0	reserved
1	0	1	reserved
1	1	0	reserved
1	1	1	reserved

SEGMENT

SM2	SM1	SM0	Destination Item
0	0	0	Virtex for segment ADC card 1
0	0	1	Virtex for Segment ADC card 2
0	1	0	Virtex for Segment ADC card 3
0	1	1	Virtex for Segment ADC card 4
1	0	0	Main Board
1	0	1	reserved
1	1	0	reserved
1	1	1	reserved

Core is selected when MSB=0 Segment is selected when MSB=1. True for both read and write operations.

The final 2 bits in the Command 0 byte are reserved and set to zero.

Command Byte 1

This byte contains the address within the destination item indicated by Command byte 0. These addresses (or commands) can be anything that the designer of the destination item requires. The full 8 bits are available giving 256 addresses (or commands) for any item.

Examples of commands are shown below.

1)

Command 0								Command 1							
1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1

This is a Read Command to act on the Segment module. The final destination is the *Virtex for ADC Card 3*. The command to be done within this Virtex device is hex 5.

2)

Command 0								Command 1							
0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1

This is a Long Write to the Core module. The final destination is the *Virtex for Segment ADC Card 2*. The address or command to be executed within the Virtex device is hex 3.

Time out on failures.

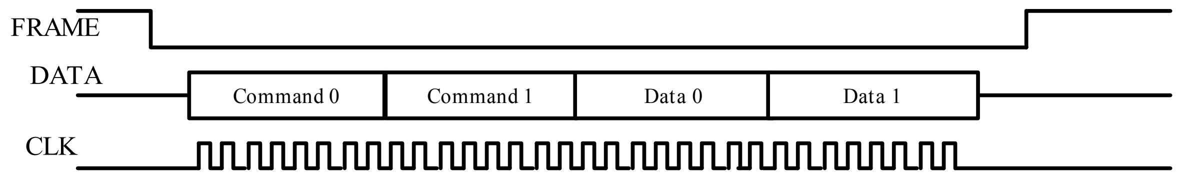
If an error occurs within any of the state machines within a Control Spartan or Virtex device a watchdog timer will force a Reset. This will take approximately 30 seconds (*subject to review*) to be detected to cover the longest possible internal response time. External controllers must also have a timeout in order to detect any failure within them. The Unit will be ready for a new command after the 30 second period and be fully reset. (Please section later on '*Timeout Reset'*).

Data interface between Control Spartan and the VIRTEX devices.

The Control Spartan device within the Digitiser module will deal with commands on a one by one basis. This is to simplify the code within the Spartan and make the structure more sensible. Each command will be sent to the appropriate Virtex device together with its Data.

In order to appreciate the commands and command structure, some basic concepts about the workings of the serial link between the Spartan and Virtex are required.

The diagram below shows how the FRAME, DATA and CLOCK signals are used for a Simple Write or Read command from the Spartan to the Virtex. The FRAME signal determines the completion of the command transfer. (In the case of a Long Write this may span many hundreds of Data Bytes.) (Note FRAME is active Low)



The FRAME signal is monitored to detect the start and end of the command being issued. Only single commands will be received by the Virtex. All Virtex command strings will start with two bytes of Command followed by two bytes of Data, (a total of 32 bits). This is expanded on for the Long Write command and is shown below

If the Virtex fails to recognise the Read or Write command it must issue an error message. This is shown in the ACK section below.

Write Commands.

A total of four bytes (32 bits) will be received for a normal Single Write Command thus:-

Command 0	Command 1	Data 0	Data 1

If a Long Write is to be done to a Virtex device then this will take the form:-

Command 0	Command 1	Data 0	Data 1	Data 2	Data 3	etc.	Data N

It should be noted that a handshake mechanism is in place so that the data stream can be suspended at any byte boundary to allow for processing of the data in the Virtex. (This is true in both directions, Read and Write). It may be necessary to use this to halt data transmission after the first 2 data bytes are received (to produce an initial format similar to Simple Write above) in order to set up the correct data path within the Virtex for the Long Write command. This is similar to RTS/CTS handshaking in RS232.The operation of this handshake is described in the *Electrical specifications and timing* section.

Read Command

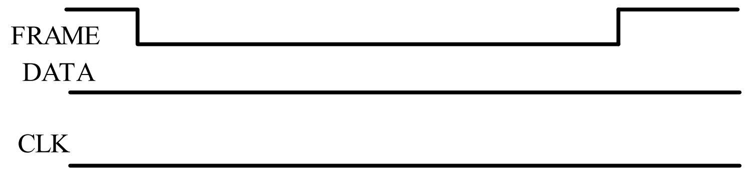
Data read from a Virtex device will take the form:-

Command 0	Command 1	Data 0	Data 1	Data 2	Data 3	etc.	Data N

Note that the read command is echoed at the start of the data stream.

ACK signals from the Virtex to Spartan.

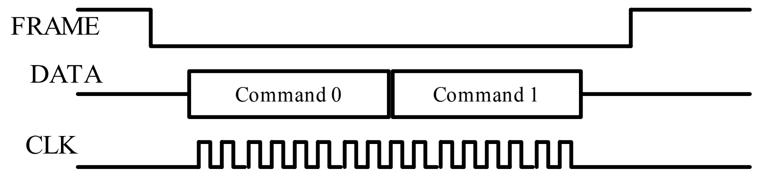
By using the ACK signal it is possible to acknowledge a good Write command without sending any Data or Clocks. The ACK command reply MUST follow the end of the FRAME received from the Spartan and not before. Please see *Electrical specifications and timing* section. The ACK should take the form shown here:-



Good Write ACK

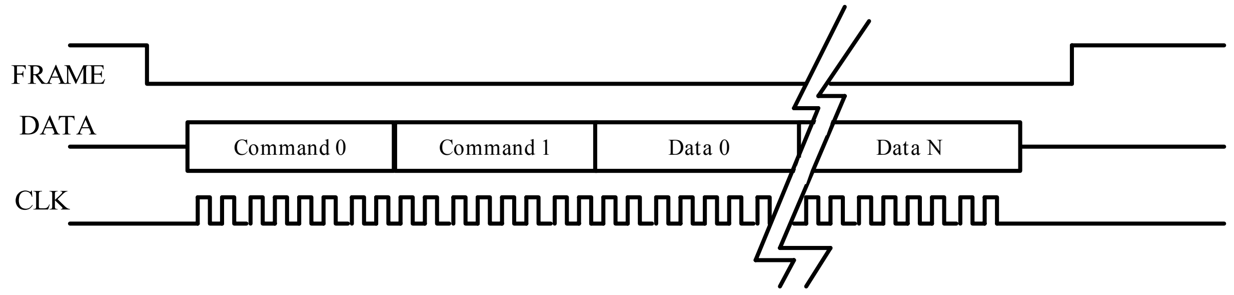
Timing is discussed in the *Electrical specifications and timing* section.

If a Read or Write command FAILS for any reason within the Virtex (e.g. not understood, not available etc.) then the following is used as the ACK signal from the Virtex. The Command is echoed back as a debug for the system and will be returned to the original Controlling source PC.



FAILED Read or Write ACK

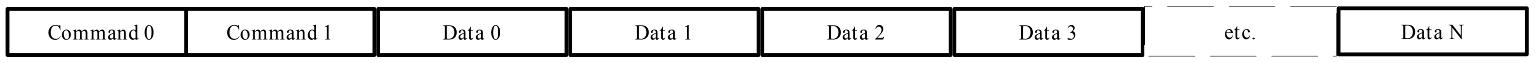
A good Read response from the Virtex to the Spartan is as shown here:-



GOOD data read from Virtex implicitly indicates good Read ACK.

The ACK in this case is the correct data being transmitted as expected.

The Data stream is in the following form, and is the same format as would be received in a Long Write.



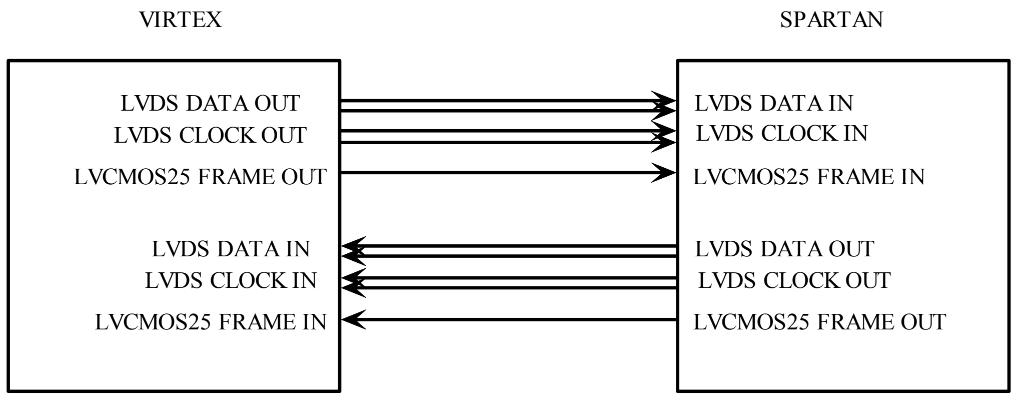
Electrical Specifications and Timing.

This electrical specification is specifically intended to be used between the Spartan and Virtex devices. It equally applies between the Core and Segment Spartans.

There are 5 wires connected in each direction between each Virtex and Spartan. These wires are as follows:-

- DATA high – LVDS
- DATA low – LVDS
- CLOCK high – LVDS
- CLOCK low – LVDS
- FRAME – LVCMOS25 (Active Low)

The CLOCK and DATA lines are complimentary Differential (LVDS) , whereas the FRAME signal is Single ended 2.5V CMOS (LVCMOS25). Only the FRAME is active Low.

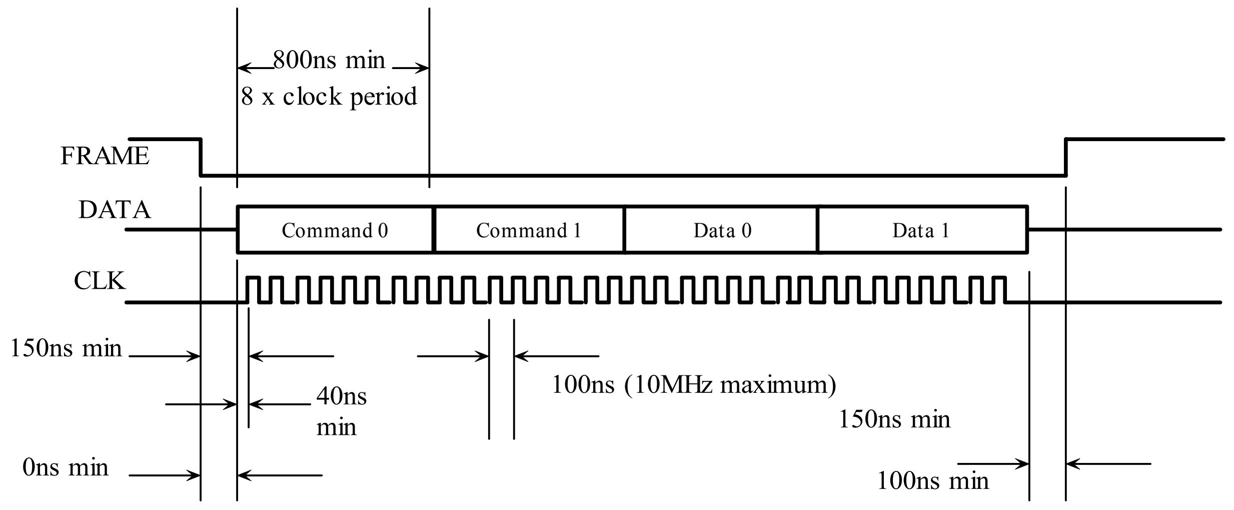


The FRAME line controls the data transfer and has two functions depending on where it occurs within a transfer cycle. Primarily the FRAME line is used to instigate a Data transfer. It goes true to start the transfer, remains true during the transfer and goes false at the termination of the transfer. This indicates to the receiving device that it has received the entire data stream. The data transmission is complete and the receiving device can now process the data etc.. Data and clock should not be presented or accepted if the FRAME signal is not true.

In this instance TRUE is active LOW and FALSE is active HIGH with a pull-up on the FRAME input line for any non connected signals.

In order to allow for the eventuality that the receiving end is unable to accept further data, the FRAME signal in the *return direction* acts in a similar fashion to an RS232 RTS/CTS status line. This, importantly, occurs during the transfer time only (when the forward direction FRAME signal is true). This means that any transfers can be temporally suspended until the receiving device can again accept them. The timing for this is shown later.

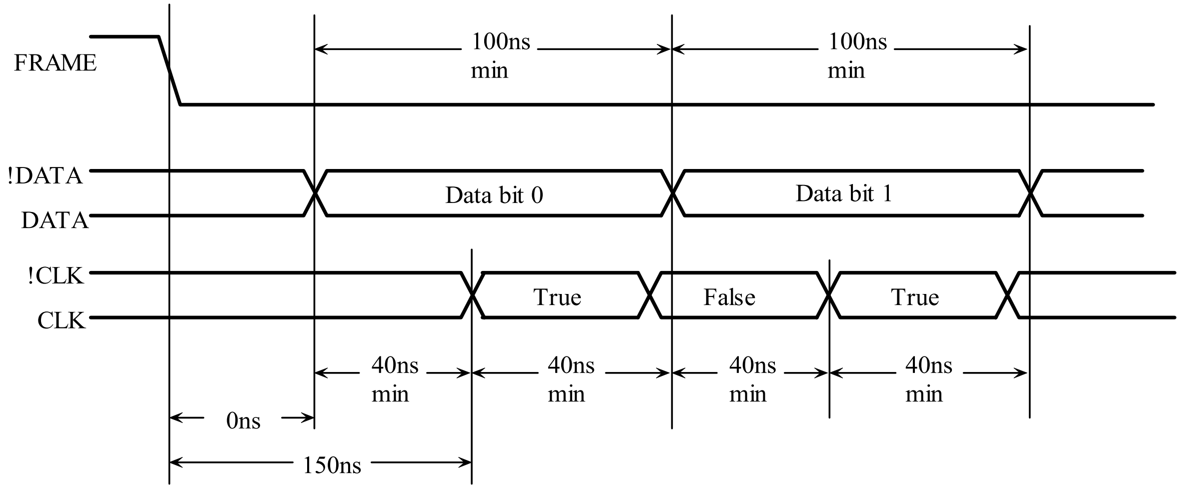
Basic Frame Data Transfer Timing.



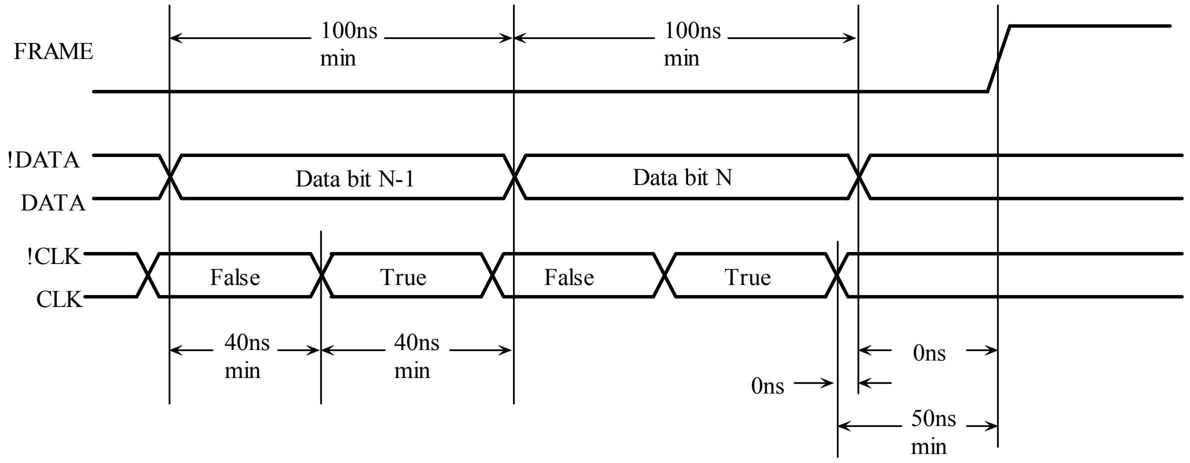
The above diagram shows the typical timing between the Virtex and Spartan in either direction. The Maximum permissible Clock rate is 10MHz and the FRAME, DATA and CLOCK edges should be programmed to have slow slew. The times shown above are assuming a maximum clock speed. Please note that it is possible to have gaps between bytes sent from the transmitting end, however it is anticipated that data will normally be continuous.

The clock rate is determined by the source of the transfer up to a maximum of 10MHz. All DATA and CLOCKs are contained within the FRAME.

Timing diagram of start of data transfer. (Transmit end)

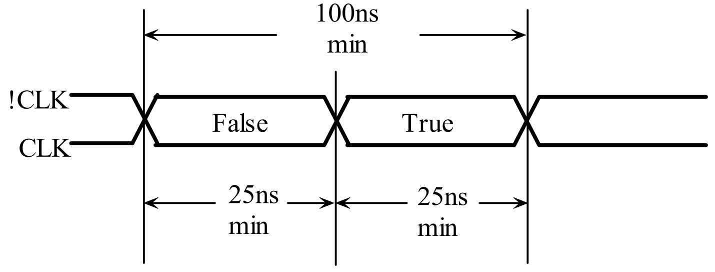


Timing diagram of end of data transfer. (Transmit end)



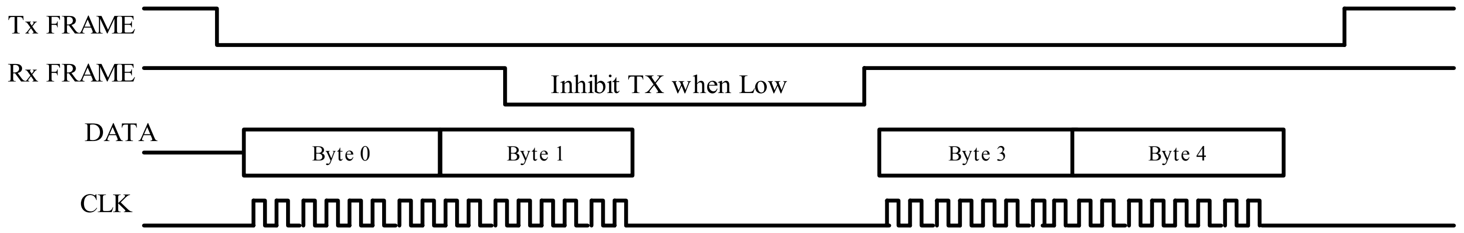
Clock Timing.

The Maximum permissible clock speed is 10MHz . There is no lower limit but a rate of 100kHz would seem reasonable to maintain transfer rate times. The Clock does not need to be a 50% mark space ratio as long as it complies with the timing below:-



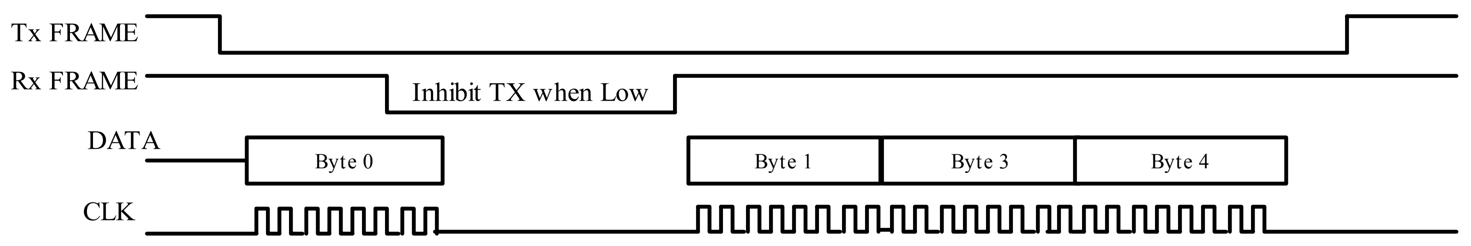
Handshaking

During long data transfers it may be necessary to interrupt the reception of data until space is available in a buffer at the receiver end for example. To facilitate this a method of handshaking using the receiver FRAME output signal has been included.

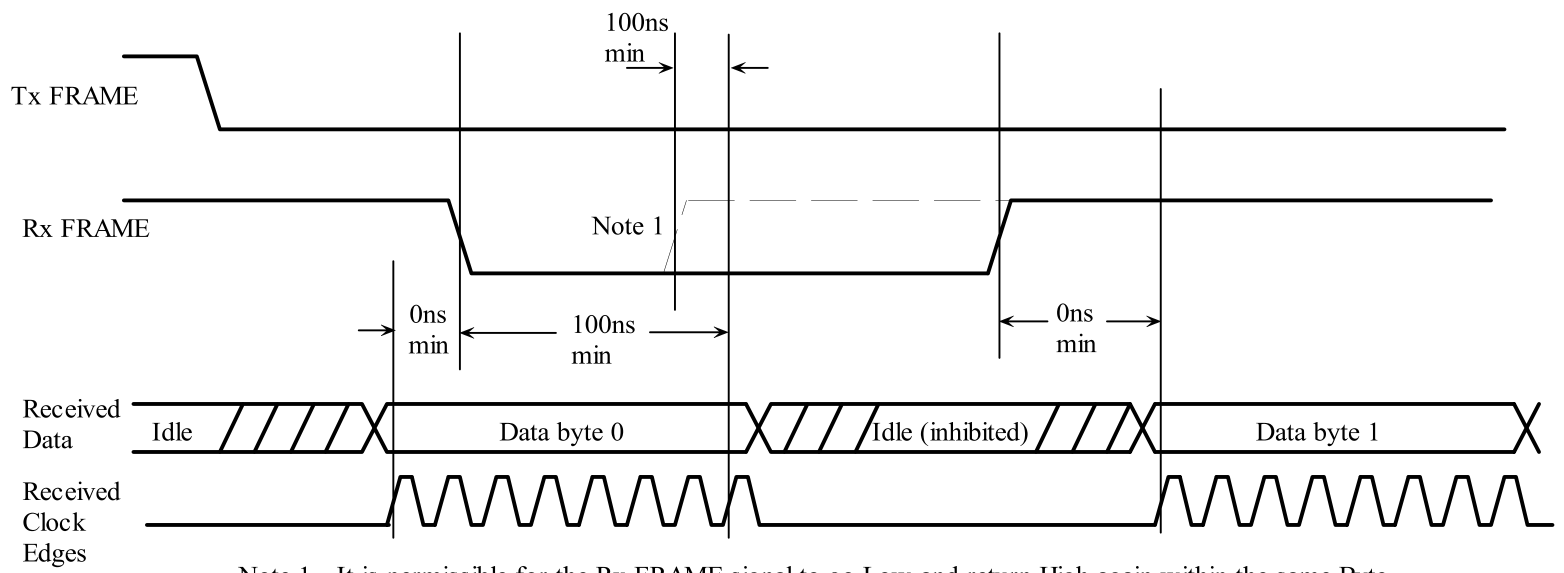


In the above diagram the Tx FRAME signal is generated from the transmitting end together with DATA and CLOCK. When the Rx FRAME signal is detected *LOW* the next DATA word transmission is inhibited until the Rx FRAME returns *HIGH*. The Rx FRAME only inhibits the transfer of complete data bytes and must be asserted during the time of the byte previous to that which is to be inhibited from transmission. It is important to note that the Rx FRAME signal can only operate in this way when the Tx FRAME is asserted *LOW*. Rx FRAME has a different function outside of this time. (See *ACK signals from the Virtex to Spartan* above).

A second example is shown below:-

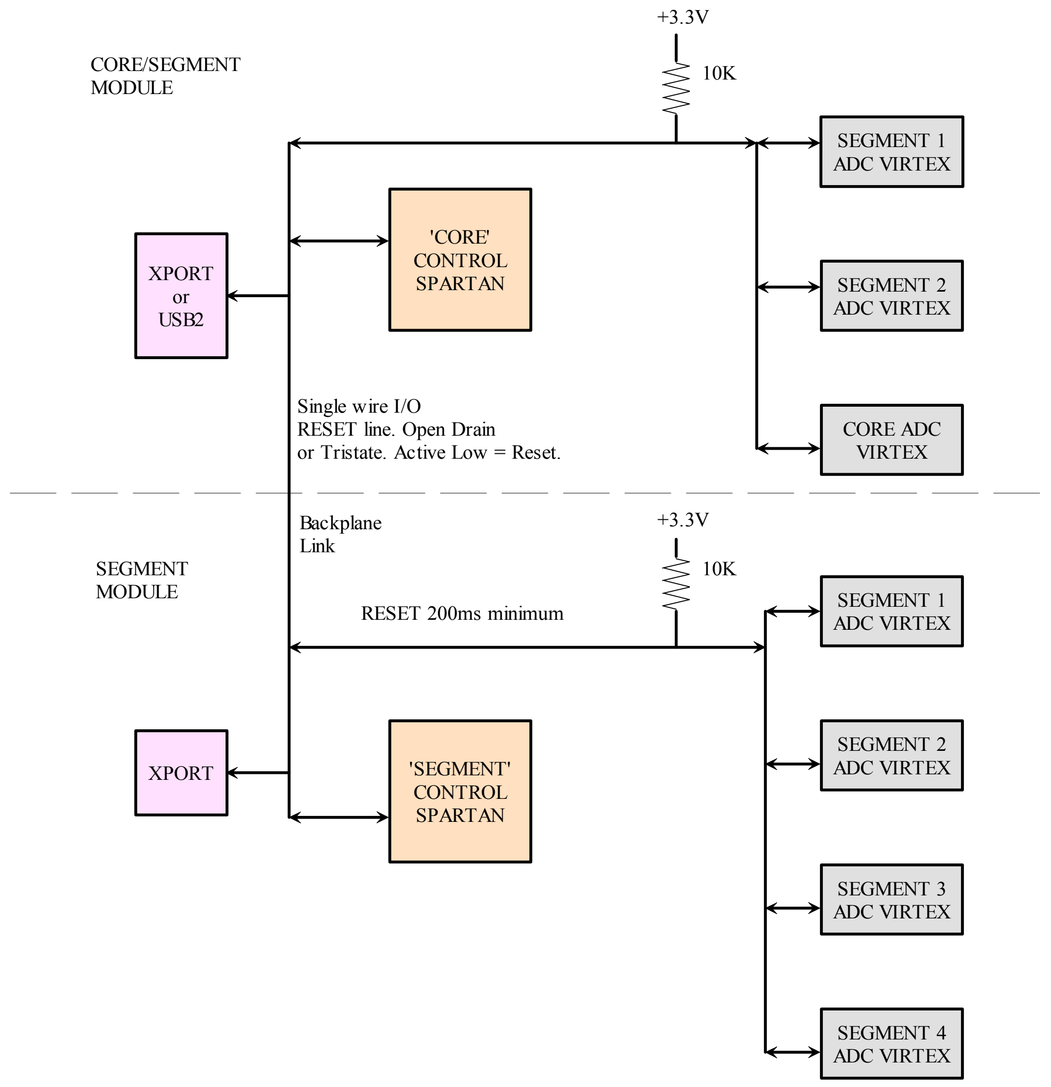


The timing diagram below (based on the diagram above) shows the relationship of the clock edges to the FRAME signals.



Note 1 - It is permissible for the Rx FRAME signal to go Low and return High again within the same Byte.

Timeout Reset



What follows is a new concept within the Digitiser and is required to ensure that the unit can recover from any fault conditions without user intervention.

The diagram on the left shows a common wire that runs between all the Virtex, Spartan and XPORT devices in a Digitiser box. This line is a LVCMOS25 open drain (or tri-state) signal with pull-up resistors in both Segment and Core modules. Any device, be it Virtex or Spartan, can initiate an I/O reset by pulling this line low. In order to reset the XPORT devices the reset must be at least 200ms long. It is envisaged that a timer will be placed on this line in the originating device to stretch the reset pulse time to this minimum value.

The I/O Reset line must be monitored by all the devices that can drive it and they must respond to a reset input by completely resetting all the input/output circuits within themselves. Registers containing downloaded values and settings that have been previously configured will be unchanged. Only the I/O interface is to be reset. This is to ensure that should a device hang up mid operation for any unknown reason, the Digitiser can detect this and reset itself so that normal communication and control can be resumed.

Changes to this document.

Rev 4 from rev 3. Length bytes L0 and L2 change so that L0 is now MSB and L2 is LSB of length . (Page 8) 8 Jan 2006.