FPGA Firmware and Hardware Guide

for

NOvA TDU

Greg Deuerling, Richard Kwarciany, Neal Wilcer

Fermi National Accelerator Laboratory

October 14, 2014

DRAFT

Table of Contents

Revision History Master

Name	Date	Reason For Changes	Version
		Initial Release	01.00
	10/01/10	accelerator signal firmware, new registers	01.01
	10/07/10	1. TOF/2 code, 2. added TCLK event $1D as requested by Andrew Norman, 3. LED pulse stretching so that the LEDs on the TDU can be seen by the naked eye	01.02
	10/29/10	Flip the timing link clock 180 degrees to compensate for reversed clock pins on the RJ45 connectors	01.03
	10/29/10	Change the WRD_CNTR signal from an integer to a STD_LOGIC_VECTOR(3 downto 0) because the integer would occasionally overrun its limit of 4 and the counter would end up in deep space.	01.04
	11/5/10	Fix for multiple register writes when using the SPI bus.	01.05
	11/30/10	1. Presetting and starting of NOvA counter for ALL units is implemented. Done by setting the "Time Synchronization" bit in the control register. 2. Add checksum to command packets 3. Added "command busy" bit in status register to let user know that the command packet registers are being used to send commands on the timing bus, and that the user should not attempt command register access while this bit is high. 4. Modified the NOvA counter initialization to make it more reliable. In previous versions, registers would occasionally get mangled because of high speed 128Mhz clock running arithmetic functions on 56-bit registers. 5. Make sure that when ARM command $17* is issued, no reload of the NOvA counter occurs. That command is just used as a sanity check for the GPS TOW and Week. 6. Created a signal that blocks a SYNC from being sent on the timing link if the value of the lower 32-bits of the NOvA counter is greater than xFFFFFEFF or less than x0000040.	01.06

		NOTE: This new revision of firmware eliminates NOvA counter initialization after a reset. NOvA counter initialization occurs only when the "Time Synchronization" bit is set in the control register.	
	1/2/11	1. Changing of NOvA time zero. 15 seconds added to original NOvA time zero value so NOvA time will be sync'd with UTC time. Any future leap seconds will be handled by NOvA software. 2. All 72Mhz clocks were eliminated to avoid crossing of clock domain boundaries. This should also help with glitching that we've seen on some of the command transfers on the timing link.	01.07
		Modified the 1/2 second timer used to delay an interrupt request to the ARM asking for the latest GPS time. The new timer issues its request at 1/4 second. The hope is that this will allow the ARM plenty of time to process the interrupt, and get the current POLYT message from the GPS unit. This modification was made in response to a problem where timestamps for event slices were off by ~1 second. This could be accounted for if the GPS POLYT message was being decoded after a new 1 pulse per second occurred before the firmware loaded the new NOvA counter times into the downstream TDUs and DCMs.	01.08 (Never released)
	03/09/11	1. Changed the fiber optic TX inputs. TX1_D(0) is the 32Mhz link clock, TX1_D(1) is CMD, and TX1_D(2) is SYNC signal. This requires a wire modification to the TDU. 2. Modified the 1PPS checking circuitry. It now constantly checks the 1PPS intervals. Results that exceed the max allowed variance are saved and sent to the PPS verification registers 3. Added LED definitions. The "Link" LED is lit when we have an error free optic link. The "GPS Lock" lets us know that the GPS unit has satellite lock. The "Error" LED lets us know that there was a mismatch in the GPS 1PPs counts. 4. Modified the NOvA counter initialization. Changed from raw GPS_1PPS signal to a registered version of the 1PPS. This seems to make edge detection more stable. 5. Added a GPS reset bit to the control register (Bit 14). Set the bit to '1' to reset the GPS unit. The firmware will clear this bit after the reset is done.	01.09

		6. A "Command Busy" bit (bit 2) is available in the status register. When this bit is set high, the TDU is busy with some sort of internal register access. The user should always check that this bit is low before accessing registers. 7. An "OK to SYNC" bit is available in the Status register. The user should check that this bit (bit 3) is set high before doing a timing re-synchronization. This bit is high after the GPS unit has 3D satellite lock, and the ARM is ready.	
	3/22/11	1. Modified the logic for the optical link. Previously, signal levels were set in a register. That may result in instability if those registers were accidentally accessed. We don't need register for those signals. Levels can be set DC on power up and left alone. 2. Added a command history fifo and command history registers. Any time a timing link command is written, a copy of the data, address, and header are pushed onto a fifo. The fifo is 32 words deep, and 48 bits wide (16 Data, 16 Address, 16 Header). A read of all three command history registers will pop the oldest command data, address, and header off the fifo and load them into their command history registers. If the fifo fills up, the oldest commands will be discarded, and the newest command will be pushed into the top of the fifo. Be sure to read the Data first, Address second, and Header last. The reading of the Header register loads the next oldest set of commands into the command history registers. The addresses are listed in the document, but as of this writing are: Command History Data: 0x0030 Command History Address: 0x0031 Command History Header: 0x0032 Command History Word Count: 0x0033 (see item 5) 3. Modified the ARM access to registers from signal level to edge detection. 4. Redefined bit 6 of the control register. That bit goes to the control register to reset the Command History Fifo. 5. Added a word count register that reflects the number of words in the Command History Fifo.	01.10

	5/25/11	1.Completely re-did the TOF/2 firmware. NOTE: be sure to add a loopback on the last timing link output of the DCM and TDU. 2. Bit 15 of the control register is "reset the DCM delay fifos". 3. Disabled the error mask register and added an error disable register at address 0x0019. The Error disable register works like the error mask register, but in the opposite direction. The user must write a '1' to the bit location to disable error checking. By default, all errors are enabled. 4. Re-instituted the NOvA counter initialization option. Writing a '1' to bit 7 of the control register tells the FPGA to interrupt the ARM and request the GPS time. That time value will be loaded into the Preset Time registers. 5. Fixed the portion of code that blocks SYNC pulses when we're close to 32-bit rollover for the NOvA counter. Previous version didn't actually work. Also, the blocking only occurs during a NOvA timing synchronization. 6. Remove the ability to modify the TDU_TYPE register from the SPI bus (processor). The user should never need to write this register, and should only be accessed by the ARM on power up or reset. 7. Added check for MIBS signal. This flag is "Or'ed with the check for TCLK signal. If either or both of these signals disappear, a bit is set in the error register.	V01.11
	6/28/11	1. Institute the early Sync pulse option. This will be the natural state of operation for the TDU. The early Sync pulse is needed to compensate for firmware and cabling delays in the timing system. As the pulse travels through the timing system, the early Sync will be delayed by the TDUs and DCMs so that the end result will produce a Sync pulse that is processed by all of the DCMs at almost the exact same time, and on the true 1 second boundary. However, this will only be true if the user has done the delay calculation for the timing system. 2. Add command history fifos. Whenever any timing link command block is issued, a copy of that command, it's address, and it's header, along	V01.12

		with a timestamp for that command block, will be saved to the command history fifo. The fifo will feed the command history registers. The user MUST read out the history registers in the proper order because the last read pops the next command history block off the fifo and into the command history registers. If the user reads the history registers in the wrong order, the history block data and time will be misaligned. 3. Add a warning flag in the status register that the time for a SYNC pulse to be generated will occur in 1msec, so the user may want to wait to send a command on the timing link until after the SYNC pulse has been sent. o 4. Convert the Sync pulse on the timing link from a pulse to a 16Mhz clock that changes to an 8Mhz clock period whenever a Sync pulse is needed. This change is needed due to the addition of the AC coupled timing link inputs and outputs that were implemented to solve a common mode noise problem. The timing of the creation of the encoded Sync pulse was modified so it aligns properly with the GPS 1 Pulse per Second. 5. Changed the clock initialization so that the 1 Pulse per second signal from the GPS unit will always be in phase with internal clocks. 6. Set the default value of the Delay Offset Register to 0x0200 128Mhz clock ticks. This value should account for all cable and logic delays in the DCMs and TDUs.	
	7/18/11	1. Added the following TCLK accelerator events as requested by Andrew Norman. $8F 1Hz generated by GPS RCVR in mc computer room $00 Super Cycle and Master Clock Reset $A4 NuMI Cycle Sample Trigger $A5 NuMI Reset for Beam	V100.10 (forked version of V1.10)
	7/20/11	1. Added occurrence counters for GPS lock, GPS antenna fault, and GPS Holdover. Counter values will be loaded into registers.	V101.10 (forked version of V100.10)
	7/22/11	1. Added another Error register (Error2) and Error2 Mask register. This new error register can also be enabled to generate interrupts. The Error2 Mask register performs in the same	V102.10 (forked version of V100.10)

		manner as the Error Mask register, that is, The associated mask bit must be set to '1' to enable the error. This method was modified in later versions of firmware so that all errors are automatically enabled, and the user must write the Error disable register bit location with a value of '1' to disable error checking for that particular error bit. 2. Added a mailbox register that can only be accessed by the ARM. The mailbox is used by the ARM to let the FPGA know that some type of action needs to be taken on some type of ARM incident. Users should not access this register. 3. Added an error bit in the new Error2 register that lets the user know that the GPS NMEA stream has timed out (stream is locked) and that no current GPS timing information is available. When the user is made aware of this error, the user should issue a GPS reset by setting bit 14 of the control register. The user should wait ~1 minute before trying to access GPS timing information to allow for the GPS unit to stabilize. As it is with the original Error register, the Error2 register must be cleared by a write of 0x0000 to the Error register location.	
	8/19/11	1. Changed the addressing of the history time stamp registers so the readout order matches the readout order of other 64-bit wide registers. 2. Added the following TCLK accelerator events as requested by Andrew Norman. $8F 1Hz generated by GPS RCVR in mc computer room $00 Super Cycle and Master Clock Reset $A4 NuMI Cycle Sample Trigger $A5 NuMI Reset for Beam 3. Added occurrence counters for GPS lock, GPS antenna fault, and GPS Holdover. Counter values will be loaded into registers. 4. Added another Error register (Error2) and Error Disable register. This new error register can also be enabled to generate interrupts.	V01.13

		5. Added a mailbox register that can only be accessed by the ARM. The mailbox is used by the ARM to let the FPGA know that some type of action needs to be taken on some type of ARM incident. Users should not access this register. 6. Added an error bit in the new error register that lets the user know that the GPS NMEA stream has timed out (stream is not locked) and that no current GPS timing information is available. When the user is made aware of this error, the user should issue a GPS reset by setting bit 14 of the control register. The user should wait ~1 minute before trying to access GPS timing information to allow the GPS unit to stabilize. 7. Added Decoded time registers that hold values that the NOvA time initialization process produces. There are 4 registers total that hold the entire decoded NOvA time.	
	Not released	Note: Versions 02.00 (and beyond) have all of the features through 01.13. 1. Added a fifo that contains the number of seconds that a GPS lock signal is inactive. The fifo will hold up to 64 occurrences of GPS loss of lock. 2. The GPS 1PPS verification has been modified to be more accurate 3. New LEDs are defined 4. Missing MIBS or TCLK signal is no longer an error. If the signals are missing, it is now just reported in the status register.	V2.00
	Not released	1. Correction was made to the logic that controls the ACCEL LED. Previously, if an accelerator signal was disconnected and then re-connected, the ACCEL LED would not light up again. Now, the LED will light up again when the accelerator signal is reconnected. 2. Correction was made to the logic that controls the PARITY LED. Previously, it would not turn off if the error register bits that note a parity error were cleared. Now, if the error register is	V2.01

		cleared, the PARIIY LED turns off until another parity error occurs	
	2/21/12	1. GPS 1 pulse per second is now latched on falling edge of 10Mhz GPS clock for better GPS 1PPS stability. 2. The "Aux" connector on the MTDU is an output copy of the latched GPS 1 Pulse Per Second. 3. The logic that creates the "OK2SYNC" bit in the status register has been modified to wait for a stable GPS Lock. After a pushbutton or power on reset, the GPS unit will send out-of-sync GPS 1PPS pulses, which will affect MTDU initialization. Waiting for the stable GPS 1PPS assures correct timing. Users should wait until the "OK2SYNC" bit goes active in the Status before trying to send a Sync bit out on the timing link.	V2.02
	2/29/12	1. Fixed the process that controls how a SYNC is generated. Previous versions would not always generate the correct polarity SYNC signal. 2. Changed the routine that controls the TOF delay calculations. Previous version had 31.25ns resolution. New version has 7.8125ns resolution. 3. Register writes via the SPI bus are no longer permitted during TDU initialization. This modification eliminates the possibility of a user trying execute commands when the TDU is not ready to accept them.	V2.03
	3/1/12	1. Corrected problem in the Delay Calculation firmware.	V2.04
	3/12/12	1. Corrected error in the NO COMMAND WARNING signal in the status register	V2.05
	3/19/12	1. Added signals to the Signal Tap Analyzer.	V2.06
	4/20/12	1. Added a Bit(7) to the Status register that lets the user know that a timing synchronization is in progress. 2. Modified the Status register bits. Bit(15) (MSB) is an OR of Bit(0) (Calculating Delay), Bit(2) (Command Busy), Bit(5) (No Command Warning), an invert of Bit(3) (OK to Sync), and Bit(7) (Calculating NOvA Timing). The user can check only bit(15) now and know if it's okay to perform an action on the timing link. The user can get more detailed information by checking	V2.07

		specific bits in the Status register. 3. Bug fix for sync pulse timing. 4. Added 4 registers that will, when read, show a snapshot of the current NOvA time. These registers are read only. 5. Modified the Delay calculation state machine to eliminate errors that occurred due to change in AC sync pulse timing. 6. Modified the process that controls a "reset" via bit 0 of the control register. The reset is now much more thorough, much more like a front panel pushbutton reset. The TDU type will register is not affected, however. Be aware that recovery time from the register reset is about 20 seconds.	
	07/18/12	1. Added a second control register at addr 0x09. It is referenced as Control2 in the manual. 2. 6 new bits are defined in the new Control2 register. 3 bits are different "scrub" operations, where the FPGA is rebooted (and GPS reset if the unit is a MTDU), and 3 bits are different "soft reset" operations. Scrub operations: a) When the user sets bit "0" high, the GPS unit will be reset and the FPGA on the Master will be reloaded. b) When the user sets bit "1" high, the slaves in the timing chain will have their FPGAs rebooted. c) When the user sets bit "2" high, the master will have its GPS unit reset and its FPGA rebooted. After completion, the Master will then reboot the FPGAs on the slaves that are in the timing chain. Soft Reset operations: a) When the user sets bit "4" high, the Master's registers and variables are reset to power up values. b) When the user sets bit "5" high, the Slave's registers and variables are reset to power up values. c) When the user sets bit "6" high, first the Master and then the Slaves in the timing chain will have their registers and variables reset to power up values.	V2.08

		3. The interrupt register and interrupt mask register have a new bit defined. Bit 15 of the interrupt and interrupt mask register are defined to interrupt the ARM processor so that the ARM processor will issue a "scrub" reset the appropriate devices. The user should not set this bit. This bit is used by FPGA firmware. If the user wishes to issue a "scrub" reset, it should be done by setting the proper bit in the Control2 register. 4. The Reset bit (0) in the control register is no longer used. It has been replaced by the multiple types of resets in the Control2 register. 5. Added a "scrubbing system" bit (bit 8) to the Status register that lets the user know that the slave TDUs in the timing chain are being scrubbed, or a soft reset is occurring. This bit is "Or'ed with the other bits to create bit 15 the "timing link busy" bit. The user should always monitor bit 15 of the Status register before performing any action on the timing link.	
	7/30/12	1. Fixed bug in "scrub reset" process. Previously, the slave wouldn't scrub its FPGA when the "scrub all" option was selected.	V2.09
	8/15/12	1. The process that creates time stamps for Accelerator Events now operates at 128Mhz(7.8125ns period) as opposed to the pseudo 64Mhz clock that is the NOvA time counter. This means that the timestamp for Events is now a 57 bit counter as opposed to the 56-bit counter used for the NOvA time. 2. Fixed a bug in the code that controls readout of the Accelerator Event timestamp fifo. Previously, the fifo wasn't being implemented correctly and events were being read out of the fifo and dumped to the Event timestamp registers as soon as they were written to the fifo. The fifo now operates correctly in that the only time the fifo dumps a new event timestamp to the timestamp registers is after a read of the timestamp registers. Note that the fifo advances another word to the timestamp registers after a read of the Event Number Register, so the proper method for reading out the event timestamp and the event number register is for the user to first read out all four of the timestamp registers, and finish by reading the event number register. Failure to do so could lead to skewed event information. As per Ron Rechenmacher's	V2.10

		request, the fifo is set so that if the event fifo fills up, newer events are discarded and the older events remain in the fifo until read out. 3. The Event timestamp fifo overflow flag is no longer set in the error register. It is now a bit in the Status register. 4. A new "Event Fifo Overflow Count" register has been added. If the user doesn't read out the Event timestamp registers in a timely manner, the Event fifo will overflow. When this occurs, the "Event Fifo Overflow Count" register will be incremented each time an event is missed. The max value of this register is xFFFF. 5. The Control2 register has a new bit definition. Bit 8 is now the "Clear Event Fifo" bit. When this bit is set, the Event fifo is flushed and the Event Fifo Overflow counter is zeroed out. 6. Modified the timing of the GPS 1PPS. Since the 1PPS used throughout the FPGA firmware was created from the falling edge of the 10Mhz GPS clock that immediately follows the rising edge of the GPS PPS, we end up with a skewed FPGA 1PPS that is late by ~ 50ns. For accuracy, this needed to be corrected. The new firmware recreates a 1PPS for the FPGA that is aligned with the real GPS 1PPS.	
		1. Added an option to change the width of the 1PPS that is output on the AUX front panel connector. The width value is written to the PPS_WIDTH register by the user, and each bit value is worth 7.63usecs. The full value (xFFFF) will give a little over 1/2 second 1PPS pulse width on the AUX output.	V2.11 (not released)
		1. Modified the width change option on the GPS 1PPS that comes out of the AUX output on the TDU front panel so that a value of x"0000" in the EXT_PPS_WIDTH register results in NO 1PPS coming out of the AUX front panel connector. This change was requested by Dr. Norman. Default value on power up is x"0000", meaning no 1PPS will appear at the AUX output.	V2.12
		1. Due to the fact that the phase relationship between the GPS 1 pulse per second and the 10Mhz clock that come out of the GPS unit is not stable even though a GPS Lock is achieved, we needed to extend the delay of the flag saying	V2.13

		that the TDU Master is ready for operation until the phase relationship stabilizes. The time between when the GPS Lock signal goes active and the phase relationship stabilizes can be ~ 2 minutes. This version of the firmware has the modification that extends that delay. We will be safe and actually delay the flag saying the unit is ready for operation for 3 minutes after the GPS Lock is received. This new function will activate on a press of the front panel reset button, when a "scrub" command is issued to the MTDU, or when a MTDU power cycle occurs. The user should monitor bit 3 (OK to SYNC) and/or bit 15 (Timing Link Busy) of the status register. When bit 3 goes high and bit 15 goes low, the unit is ready for operation.	
		1. A problem was found in the extended reset that was created in version 2.13. It doesn't affect operation, but it flags false errors in the error register. This version fixes that. 2. A problem was found with the "scrub" devices routine, in that the routine would sometimes fail to execute. This problem has been fixed.	V2.14
		1. Fixed the "TCLK and MIBS accelerator signals present" bit 6 in the status register. This bit is supposed to be set high when both the TCLK and MIBS signal are present. Instead, it would go active if either or both of the signals was present. 2. Added two bits to the status register. Bit 10 is the "TCLK present" signal, and bit 11 is the "MIBS present" signal. If the accelerator signal is present, the flag for the appropriate signal will be set high. This modification will allow the user to independently monitor the accelerator signals. 3. Added Event List Mask2 register. The address is 0x29.This register was added to allow the monitoring of more events on the accelerator signals. 4. Added TCLK event $21 to the new Event List Mask2 register.	V2.15
		1.Add a dynamic bit set of the GPS loss of lock to error register 2. If the GPS has lost lock the bit will be high. If the GPS has lock the bit will be low. This is different than the loss of lock bit in error register 1 in that error register 1 latches a loss of lock and keeps the loss of lock bit high	V2.16

		until the user resets the bit. 2. Remove the extra “wait for 1PPS”s in the delay calculation logic. Right now, it takes about 9 seconds to complete. We should be able to get this down to ~4 seconds. 3. Change the function of the "OK to Sync" flag in the status register. If a GPS loss of lock is occurring, reset the bit and hold it low until 4 minutes after GPS lock returns. This will give the 1PPS time to stabilize.	
		1. Changed the logic that senses the BNB/MDAT signal. Evan Niner noticed that the atomic clock signal fed into the MDAT input was occasionally missed or double time stamped. This might be because of the fact that the signal being fed into the MDAT input is asynchronous or because of bounce on the signal. Added some flops to synchronize the MDAT input. Also, I will add some debounce logic that will ignore glitches for a millisecond after a rising edge is seen. 2. An issue was found with the way the firmware was handling timestamps for the accelerator events. Originally, the timestamps were fed to a common fifo using an "if/else" subroutine that could, on rare occasions, miss a timestamp. The firmware was modified to add another layer of fifos between the fifo that feeds the event timestamp registers and the event timestamps. Now each of the event inputs has their own fifo that will buffer the timestamps and will, when time allows, load the data from those individual fifos into the common fifo using a "round robin" method. This should eliminate any possibility of missed event timestamps. That is, unless the user never reads the timestamp registers.	V2.17
		1. Changed the code that increments the nova counter. Previous version incremented at 128Mhz. That may be too fast for a 60 bit counter. Added a "wait" state for one 128 MHz cycle. The counter still uses the 128Mhz clock, but now increments by 2 every other clock tick, which gives us a true 64Mhz counter. 2. Modified the NOvA time synchronization routine by eliminating an unneeded section of code that would, on very rare instances, stop the time synchronization from executing.	V2.18

		3. Added an error bit to the error2 register. Bit 4 lets the user know that a hang occurred during a NOvA time synchronization (Control register value = 0x0020). This hang can occur on rare occasions if a power pc process modifies the control register while the TDU's FPGA is in the process of a time synchronization. This problem can be eliminated if the user checks the Status register and verifies that the FPGA isn't running the time sync process before accessing the control register.	
		NOvA management requested a modification to the code that will allow an automatic reload of delay values to the master and slave TDUs after power up or a scrub. Additionally, any time a new delay value is written to any of the delay registers, that value will be saved and will be used on the next power up or scrub cycle. Since there is no non-volatile ram available on the TDU, the values will be saved in the FPGA's eeprom. A modified version of the ARM code is needed also. Modifications need to account for the following conditions: 1. Load the saved delay values after a power up 2. load the saved delay values after a scrub. 3. Save the values to eeprom after a delay learn. 4. Save the values to eeprom after a manual re-write of the delay registers. The following options are now available: Scrub MTDU, don’t load saved delay values Scrub MTDU, load saved delay values Scrub STDU, don’t load saved delay values Scrub STDU, load saved delay values Scrub MTDU and STDU, don’t load saved delay values Scrub MTDU and STDU, load saved delay values	V2.19

Revision History Slave

Name	Date	Reason For Changes	Version
		Initial Release	01.00
		Minor changes	01.01 – 01.03
	01/19/11	1. Correct problems with the copper timing link to improve reliability. 2. Added fiber timing link option. 3. Flip timing link clock 180 degrees.	01.04
	02/08/11	1. Re-wired and re-wrote the firmware for the fiber timing link. Stopped using the receive clock from the fiber timing link and instead used one of the data bits on the timing link to create a 32Mhz timing link clock. This requires a wire mod	01.05
	03/09/11	1. Created the firmware to decode the incoming 8b/10b command stream. 2. Define the "GPS Lock", "Error", and "Link" LEDs. 3. A "command busy" bit (bit 2) is available in the status register. When this bit is set high, the TDU is busy with some sort of internal register access. The user should always check that this bit is low before accessing registers. 4. Removed the Fiber Register 5. Removed the "initiate 1PPS verification" option in the control register. 6. The error register bit 3 is checksum error. The TDU slave verifies the checksum that accompanies the data from the upstream TDU.	01.06
	04/11/11	Modified the Signal Tap analyzer portion of the firmware to allow better debugging Test stand Slave TDU	01.07
	04/13/11	1. Disabled access to the TDU type register from the timing link. Access to the TDU type register is allowed only through the ARM	01.08
	04/25/11	1. Incorporate the TOF/2 (delay calculation) firmware. NOTE: be sure to add a loopback on the last timing link output of the DCM and TDU. 2. Added "Delay Offset" register that contains the Offset value used during TOF/2 calculations. 3. Fixed the clock recovery error that occurred after a Master reset when the Slave TDU is using the fiber link. 4. Removed TDU_TYPE from the soft reset list.	01.09

		TDU would get reset and think it was a copper slave even when it was a fiber slave. 5. Bit 15 of the control register is "reset the DCM delay fifos". Setting this bit will set the delay timing of the command, clock, and sync pulses from the TDU to the DCMs to zero. 6. Disabled the error mask register and added an error enable register at address 0x0019. The Error disable register works like the error mask register, but in the opposite direction. The user must write a '1' to the bit location to disable error checking. By default, all errors are enabled. 7. Disable output links if fiber loses lock via Serdes sync error or loss of power. Wait until certain number of comma characters have been seen before re-establishing link outputs to downstream TDUs and DCMs.	
	6/29/11	1. Sync pulse was modified from a single pulse on a DC signal to an elongated pulse embedded in a 16Mhz clock signal. 2. LEDs on RJ45 connectors and front panel are defined.	01.10
		Note: Firmware versions starting with "02" are for the new XTDUs. 1. Finalized the LED firmware. 2. Added a second Error register so it matches the MTDU.	V02.00
	4/20/12	1. Modified the code that controls the timing delay calculations. 2. Changed the method for decoding 8b/10b data so that false data is not decoded after a loss of timing clock lock and a re-lock.	V2.01
	7/17/12	1. Fixed bug wherein register contents were getting overwritten when slave was a SERDES (Fiber) Slave. 2. Fixed Error register and LEDs so they don't give a false "link clock missing" error. This was done by delaying error logging until the system has settled, which is about 20 seconds. 3. Added a second Control register, Control2	V2.02

		register. 4. Control2 register has 2 new bit definitions. They are the "Scrub" reset option and the Soft Reset option. The "Scrub" option reboots the entire FPGA. The "Soft Reset" option resets all variables and registers to their power up state. This is to be used in the event of a TDU hang, or system initialization. 5. Change the logic that checks for a return Sync so that it uses the onboard 128Mhz clock instead of the timing link clock. This was done to avoid any possible problems with the starting and stopping of the Sync, clock, and data signals during a Delay Learn command.	
	7/30/12	1. Fixed a bug in the "Scrub Reset" utility.	V2.03
		1. Added a framing error bit to error 2 register. When a command packet is sent down the timing link and received by a slave, the slave decodes the packet. The packet should consist of six 8-bit words – a header, two address words, two data words, and a checksum. If the slave detects command packets that are not complete, this bit will be set 2. Blocked false timing link and SERDES errors during an MTDU "scrub" command	V2.04
		1. Modify code to save delay learn values to the FPGA's eeprom. 2. A SCRUB can now be performed one of two ways. A) Do not load saved delay values after a scrub, or B) load the saved delay values after a scrub.	V2.05

*** NOTE TO USER: Due to modifications to the timing link specification, version compatibility issues exist between the DCM, MTDU, and STDU. Minimum version requirements for running AC coupled timing links are listed below:

DCM: Major version 9 or later

MTDU: V1.12 or later

STDU: V1.10 or later

*** NOTE TO USER: Both Master and Slave TDU firmware versions 2.00 and above MUST be used with the version 2 Master and Slave TDU units.

1. Introduction

1.1 Purpose

This specification describes the hardware, processes, and functions that are available to the user of the Master and Slave Timing Distribution Units (MTDU, STDU).

1.2 Intended Audience and Reading Suggestions

This document is intended for anyone involved with the TDU who finds it necessary to understand the operation of the TDU registers and commands. This specification, along with the VHDL source code and schematics, will aid the user in understanding TDU operation or diagnosing problems.

1.3 References

The following documents were used as reference of guidelines for development of the firmware. They are available on the NOvA document website http://NOvA-docdb.fnal.gov/

4342-v3 Electronics and DAQ Overview for external review

3989-v3 NOvA Data Concentrator Module Hardware User Guide

3664-v6 NOvA Data Concentrator Module Functional Requirements

4354-v2 Electronics Review - DAQ Timing Distribution System

2. Overall Description

The firmware described in this document will control I/O to and from the TDU's FPGA. There will be two firmware versions, one for the Master TDU and one for the Slave TDU. The processes in the FPGA firmware will handle TDU initialization, resets, timing synchronization, delay calculations, and communication between Master and Slave TDUs, and Slave TDUs and DCMs.

The Master TDU at the near site will monitor Accelerator signals and timestamp accelerator events that are of interest to the NOvA experiment.

The Near Master TDU will be located ~200 meters from the Slave TDUs and will require a special function to operate the SERDES fiber link that is required to communicate at that extended distance. The Far Master will use wire as its communication medium. Note: The

selection of fiber or copper Master to Slave TDU connection is still being debated and may change.

The Master will handle data transmission to Slave TDUs. The only communication from Slave TDUs to the Master TDU is a return (echo) of the SYNC signal. The TDU's Ethernet communication will be through an ARM processor if the TDU is a Slave or with a Power PC processor if it is a Master.

Signals from the GPS module on board the MTDU will be decoded, modified, and transmitted to the TDU slaves as necessary by the both the near and far Masters.

3. External Interface Requirements

3.1 Communications Interface Connections

Both the Slave and Master TDU will have 10/100Base-T Ethernet ports available. MTDUs will have two Ethernet ports, one to communicate with the ARM, and the other to communicate with the Power PC. These ports will be connected to the Detector Control System Network, or DCSNet. DCSNet commands meant for the FPGA are passed through the Power PC to the FPGA on the Master TDU and through the ARM on the Slave TDU.

Two USB ports are available on the TDU. They can be described as upper and lower, as viewed from the front panel. The upper USB port communicates with the Power PC. The lower USB port communicates with the ARM

3.2 Timing Link Connections

An LVDS based set of signals is available for TDU to TDU communication and TDU to DCM communication. The signals consist of data, clock, and sync. Master TDUs will utilize the "Timing Chain Out" RJ-45 connection as the timing link to the first downstream STDU. STDUs have 4 RJ-45 connectors. "Timing Chain In" receives information from either the MTDU or and upstream STDU, "Timing Chain Out" passes the timing link information to downstream STDUs. "Timing Out Top" and "Timing Out Side" pass timing link information to top and side DCMs respectively. The data passed on these daisy-chained set of signals will be in the 8b/10b data format. The clock is 32 MHz. The SYNC will correspond to a GPS 1 Pulse per Second signal that is generated on the MTDU and is embedded in a 16 MHz clock.

Users should verify that loopback connectors are placed on any unused output timing link ports to assure that echo Syncs are properly transmitted back upstream through the timing link daisy chain.

A SERDES connection for fiber communication between the Master TDU and the first Slave TDU is available as an alternative to the RJ-45 copper link. Both Master and Slave TDUs will automatically sense whether the mode of communication is fiber or copper and initialize themselves accordingly.

3.3 Other Connections

Besides the above listed connections, the Master TDU has an input for a GPS antenna that is used to synchronize both the Near and Far MTDUs. Older versions of the MTDU use an SMA connector. Production versions will use a TNC connector.

There are 4 SMB connectors on the front panel of the MTDU. Two of the connectors are used for TCLK and MIBS Accelerator signals, another is used for the BNB wall monitor signal, and the last connector is undefined, but available for future use.

4. Registers

TDU masters control timing and synchronization to slave TDUs, DCMs, and FEBs through the use of multiple registers. Slave TDUs have a subset of the Master TDU's registers. All TDU registers are 16-bit.

Warning: Due to the fact that the SPI bus does not do any handshaking with the process that handles register reads and writes, care should be taken that simultaneous SPI and ARM register accesses do not occur.

Additionally, since register access is shared between the SPI bus, the ARM bus, and processes inside the FPGA, polling of the registers should be kept to a minimum. Failure to do so may result in failure of the FPGA processes to update their register information. The user should rely on processor interrupts instead.

TDU Register Descriptions	
0x0000	Control
0x0001	Status
0x0002	TDU Delay Value
0x0003	Side DCM Delay Value (Slave Only)
0x0004	Top DCM Delay Value (Slave Only)
0x0005	GPS Time Of Week Lo Word (Sec) (Master Only)
0x0006	GPS Time Of Week Hi Word (Sec) (Master Only)
0x0007	GPS Week (Master Only)
0x0008	Delay Offset
0x0009	Control2
0x000A	1PPS Verification Counter Lo Word (Master Only)
0x000B	1PPS Verification Counter Hi Word (Master Only)

0x000C	Interrupt
0x000D	Early SYNC (Master Only)
0x000E	Error2
0x000F	Error2 Disable
0x0010	Preset Time, Bytes 1,0 (Master Only)
0x0011	Preset Time, Bytes 3,2 (Master Only)
0x0012	Preset Time, Bytes 5,4(Master Only)
0x0013	Preset Time, Bytes 7,6(Master Only)
0x0014	Error
0x0015	FPGA Firmware ID
0x0016	TDU Type
0x0017	Event Fifo Overflow Count
0x0018	~~Error Register Mask~~ (Discontinued)
0x0019	Error Disable
0x001A	PPS_Width
0x001B	Event Timestamp Byte 1,0 (Master Only)
0x001C	Event Timestamp Byte 3,2 (Master Only)
0x001D	Event Timestamp Byte 5,4 (Master Only)
0x001E	Event Timestamp Byte 7,6 (Master Only)
0x001F	Event Number (Master Only)
0x0020	Command Data (Master Only)
0x0021	Command Address (Master Only)
0x0022	Command Header (Master Only)
0x0023	Event List Mask (Master Only)
0x0024	Interrupt Mask (Master Only)
0x0025	GPS Lock Loss Counter
0x0026	GPS Holdover Counter
0x0027	GPS Antenna Fault Counter
0x0028	GPS Lock Loss Timer
0x0029	Event List Mask2 (Master Only)
0x0030	Command History Time, Byte 1,0 (Master Only)
0x0031	Command History Time, Byte 3,2 (Master Only)
0x0032	Command History Time, Byte 5,4 (Master Only)
0x0033	Command History Time, Byte 7,6 (Master Only)
0x0034	Command History Data (Master Only)
0x0035	Command History Address (Master Only)
0x0036	Command History Header (Master Only)
0x0037	Command History Word Count (Master Only)
0x0040	Decoded Time, Bytes 1,0 (Master Only)

e0x0041	Decoded Time, Bytes 3,2 (Master Only)
0x0042	Decoded Time, Bytes 5,4(Master Only)
0x0043	Decoded Time, Bytes 7,6(Master Only)
0X0050	Time Snapshot, Bytes 7,6 (Master Only) (Read Only)
0X0051	Time Snapshot, Bytes 5,4 (Master Only) (Read Only)
0X0052	Time Snapshot, Bytes 3,2 (Master Only) (Read Only)
0X0053	Time Snapshot, Bytes 1,0 (Master Only) (Read Only)
0x00FE	ARM Mailbox (Internal use only. Do not use!)
0x00FF	Internal Register for Firmware Initialization. Do Not Use!

4.1 Control Register – 0x00

This register contains bits which will be written by the ARM/PPC that will command the FPGA to initiate a particular function.

Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RF	GR	MM	ES	CE	EE	SS		IC	CH	TS	LE	VE	PE	CE	~~RS~~

~~Bit 0 – RS: Reset. When set high, the FPGA is forced into a reset state. The Reset bit is automatically cleared on the next 72 MHz clock tick. *Slave and Master.*~~

Bit 1 – CE: Count Enable. When set high, the Timestamp Counter will begin incrementing upon receipt of the next Sync pulse. When set low, the Timestamp Counter will stop incrementing upon receipt of the next Sync pulse. The Count Enable bit is persistent, and will remain in the state it is set to until it is changed. *Slave*

Bit 2 – PE: Preset Enable. When set high, the Timestamp Counter will be preset to the value in the Timestamp Counter Preset Register upon receipt of the next Sync pulse. The PE bit is also cleared by the Sync pulse. *Slave*

Bit 3 – VE: Verify Enable. When set high, the Timestamp Counter will be compared with the Timestamp Counter Preset Register upon receipt of the next Sync pulse. Errors will generate a status packet with the Timestamp Counter Error bit set. The VE bit is also cleared by the Sync pulse. *Slave*

Bit 4 – LE: Timing System Learn Enable Command. All timing modules and DCMs in the system enter learn mode. Upon the next receipt of SYNC, each module will start a timer that will run until the SYNC_ECHO signal is received. The SYNC_ECHO pulse is the SYNC pulse that is returned from the last module in a given chain. Each module will then preset its internal delay register with a value of ½ of the time measured during

learn mode. This allows cable delays in the timing distribution system to be compensated for. *Slave and Master.*

Bit 5 – TS: Time Synchronization. Setting this bit tells the Master TDU to initialize the NOvA counter for the entire system. The Master TDU will preset Slave TDUs, DCMs, and FEBs with the current time, and then send a SYNC that tells all of the units to start their counters. This bit will stay active until the synchronization initialization is completed. *Master*

The TDU Master's time synchronization state machine waits for the TS bit to be set in the TDU Master control register. When this occurs, the state machine waits for a rising edge of the 1 Pulse per Second from the GPS unit. The state machine waits for a half second after the GPS 1Pulse per Second rising edge so that the GPS unit's POLYT string that contains timing information has the correct time. The ARM is then interrupted with a request to get the new time from the GPS unit. The ARM responds by writing the new time values into the Master TDU's GPS registers. When this write occurs, it sets into motion the process that calculates the new NOvA Time.

Once the new NOvA Time is calculated, it is sent over the Timing Link in 4 words (1 command packet) to the downstream TDUs, the DCMs, and the FEBs. The downstream unit's control registers are written with a value that tells these units to use and increment the new NOvA Time values when the next SYNC signal is sent from the Master TDU. Finally, the state machine sends the SYNC pulse down the timing link, starting all counters in the NOvA system simultaneously.

Bit 6 – CH: Clear Command History Fifo. Setting this bit will clear the Command History Fifo. This bit will clear itself after the Command History Fifo has been cleared. *Master*

Bit 7 - IC: Re-Initialize the NOvA Counter. The TDU Master FPGA will interrupt the ARM processor and request that the current UTC TOW (Time of week, in seconds) and the current UTC week be reloaded into their appropriate registers. Once this occurs, the NOvA Time will be recalculated and re-started. *Master*

Bit 9 – SS: Send SYNC
The Master will send out a SYNC pulse aligned with the 32Mhz timing clock at the next 1 pulse per second from the GPS. This bit clears on a return SYNC. Active high. The red LED on the RJ45 timing link connector will flash when a SYNC is sent. *Master*

Bit 10 – EE: Enable Events
Setting this bit high enables Accelerator events to be monitored and recorded. *Master*

~~Bit 11 – CE: Clear Error Register~~
~~Writing a one to this location will clear the error register~~

~~Bit 12 – ES: Early SYNC Enable~~
~~Set this bit high if the user wants to use the Early SYNC option. See Early SYNC register for more details. Because of the timing requirements to implement an early SYNC, the user should be aware that the actual SYNC may be delayed up to ~ 2 seconds.~~

Bit 13 – MM: Manually Modify DCM Delay
Set this bit high to modify the Timing Link Delay to the Top and Side DCMs from this particular Slave TDU. When this bit is set, the delay value in the Top and Side DCM Delay register will be used as the new delay value for the fifos that control Command, Clock, and Syncs to downstream DCMs. This bit will clear itself after the delay has been modified. *Slave and Master*

Bit 14 – GR: GPS Reset
Set this bit high to Reset the GPS unit. This bit will clear itself once the reset is complete. *Master*

Bit 15 – RF: Reset the DCM delay fifos in the Slave TDUs. This bit can be used by the user, but is used by firmware when doing a Delay calculation for the timing system. The DCM delay fifos need to be reset when a delay calculation is requested so that the firmware doesn't get an erroneous delay value because of a previous fifo delay setting. *Slave*

4.2 Status Register – 0x01

Status Register – 0x0001

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TB				MP	TP	EF	SS	CT	AP	NC	NS	OS	CB	DC	CD

Bit 0 – CD: Calculating Delay
When a Master or Slave TDU is determining what their timing delay is, this bit will be set "1" by the FPGA. When the delay calculation is completed, this bit will be set to "0". *Master and Slave*

Bit 1 – DC: Delay Calculated
A TDU must have received a command to calculate delay, seen the SYNC and the 1SYNC Echo pulse, calculated the delay, and saved the delay value to the TDU delay register to set this bit to "1". After a reset or if a command to (re)calculate delay is received, this bit will be set to "0". *Master and Slave*

Bit 2 – CB: Command Busy
When the FPGA is in the process of sending out a command block, this bit will be high. The user should not attempt to write a command to the command registers while this bit is high, as it will result in corrupted commands. The user can monitor this bit as a flag to know when it's safe to send a new command. *Master and Slave*

Bit 3 – OS: OK to SYNC
Master only. This bit will go high when we have a GPS Lock and the ARM has done its initialization. The user should wait until this bit goes high to do a NOvA Time Synchronization. If a GPS loss of lock occurs, this bit will be reset until the GPS lock returns *and* a 4 minute period has elapsed allowing the GPS 1PPS to settle. *Master*

Bit 4 – NS: No Return Sync Pulses
This bit will go high when the Slave TDU has no downstream TDUs or top or side DCMs attached to a timing link. This is not necessarily an error since all timing link chains may not have a DCM or downstream TDU attached. *Master and Slave*

Bit 5 – NC: No Command Warning
Master only. This bit will go high when the Master TDU is within 200 usecs of issuing a Sync pulse. The user should not send a command block on the timing link when this bit is high, as an unexpected Sync pulse could occur during command block transfer. *Master*

Bit 6 – AP: TCLK and MIBS accelerator signals present
Master only. This bit will go high if both the TCLK and MIBS accelerator signal are present. This bit has been moved from the error register.

Bit 7 – CT: Calculating NOvA Timing
Master only. The firmware is running the state machine process that does NOvA timing initialization. The user should not access any timing link command registers or perform any timing link activity while this bit is set.

Bit 8 – SS: "Scrubbing" System
Master only. The firmware is running a process that is scrubbing or resetting the MTDU or STDU. The bit is or'ed with other bits to create bit 15 of the Status register.

Bit 9 – EF: Event Fifo Full
Master only. There is a 128 word deep fifo that holds the Event data and timestamps. If this fifo becomes full as a result of the fifo not being read out, this bit will be set high. Once the fifo is no longer full, this bit will go low.

Bit 10 – TP: TCLK present
Master only. This bit will go high if a TCLK accelerator signal is present.

Bit 11 – MP: MIBS present
Master only. This bit will go high if a MIBs accelerator signal is present.

Bit 15 – TB: Timing Link Busy
Master only. Bit(15) (MSB) is an OR of Bit(0) (Calculating Delay), Bit(2) (Command Busy), Bit(5) (No Command Warning), an invert of Bit(3) (OK to Sync), Bit(7) (Calculating NOvA Timing), and Bit(8) (Scrubbing System). The user can check only bit(15) now and know if it's okay to perform an action on the timing link. The user can get more detailed information by checking specific bits in the Status register. The user should ALWAYS check bit 15 of the status register before performing any action on the timing chain.

4.3 TDU Delay Value Register – 0x02

Contains the TDU chain Time of Flight value /2 that is calculated using a 128Mhz counter that starts on a SYNC pulse from an upstream TDU and stops on the reception of a SYNC Echo return pulse from a downstream TDU. That count is divided by 2 to determine the delay value for the specific TDU. The rightmost bit is the LSB and the leftmost bit is the MSB. The register is cleared on a reset, a "Calculate Delay" command, or can be manually written. The maximum delay value for 16 bits @ 128Mhz is 512usecs.

4.4 DCM Delay Value (Top & Side) Registers – Side 0x03, Top 0x04

Slave Only. Contains the DCM chain Time of Flight value /2 that is calculated using a 128Mhz counter that starts on a SYNC pulse from the TDU and stops on the reception of a SYNC Echo return pulse from a downstream DCM. There are two of these registers, one for the Top DCM chain, and one for the Side DCM chain. This delay value is used delay commands, clocks, and sync pulses to the DCMs that are attached to a particular TDU to DCM timing link. The registers are cleared on a reset, a "Calculate Delay" command, or can be manually written. The maximum delay value for 16 bits @ 128Mhz is 512usecs.

4.5 GPS Time of Week (Low & High) Registers – Low 0x05, High 0x06

Master Only. The GPS unit generates a serial stream of data that is decoded by the ARM processor on the Master TDU. One of the pieces of information contained in the stream is the GPS Time of Week (TOW). The TOW data contains the number of seconds that have passed since Sunday morning 00:00.00, GPS time (not UTC). There are 604,800 seconds in a week, which translates to x 93A80 in hex. The lower 16 bits are held in the GPS TOW Low register, and the upper nibble is held in the GPS TOW High register. The upper 12 bits of the GPS High register are padded with zeroes. These register values, along with the GPS Week number, are used to calculate NOvA time.

4.6 GPS Week Number Register – 0x07

Master Only. The GPS unit generates a serial stream of data that is decoded by the ARM processor on the Master TDU. One of the pieces of information contained in the stream is the GPS Week Number. This value is the number of weeks that have passed since the GPS zero hour which start on 0h, January 6, 1980. This value, along with the GPS TOW, is used to calculate NOvA time.

4.7 Delay Offset Register – 0x08

Master and Slave. This register is written by the user and contains a value that is greater than the longest delay in the timing link chain, from Master TDU to the farthest DCM. The value is in multiples of 128Mhz clock ticks (7.8125ns). This offset is used during timing link delay calculations. If the user doesn't write a value to this register, a default value 0x200 will be written to this register. This value is equal to x200 x 7.8125ns = 4usecs.

4.8 Control2 Register – 0x09

Master and Slave. This register contains bits which will be written by the ARM/PPC that will command the FPGA to initiate a particular function.

Note: The user should monitor bit 3 of the status register after writing any of the reset bits in this register. The user should not attempt any actions on the TDU as the MTDU and/or the STDU are in an unknown state of reset while bit 3 is low.

Control2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SAN	SSN	SLN	LD	RD	IE	CF		RA	RS	RL		SA	SS	SL

Bit 0 – SL: "Scrub" Local Unit, Load Saved Delay Values. When set high, the FPGA issues an interrupt to the ARM processor, telling the ARM to do a local TDU hard reset, which includes rebooting the FPGA, and in the when the unit is a Master, the GPS unit. This type of reset is very similar to a front panel pushbutton reset, except it does not reset the ARM processor.
Slave and Master.

Bit 1 – SS: "Scrub" Slaves in Timing Chain, Load Saved Delay Values. When set high, the MTDU will send a "Scrub Local Unit" command to all of the slaves in the timing chain. *Master.*

Bit 2 – SA: "Scrub" All Units, Load Saved Delay Values. When set high in a Master, the MTDU gets "scrubbed". After it recovers, the MTDU will "scrub" all of the STDUs in the timing chain. *Master.*

Bit 4 – RL: Reset Local Unit. When set high, the FPGA resets all of its registers and variables. *Slave and Master.*

Bit 5 – RS: Reset Slaves in Timing Chain. When set high, the FPGA resets all of the registers and variables in the Slave timing chain. *Master.*

Bit 6 – RS: Reset All Units. When set high, the FPGA resets all of the registers and variables in the Master, and then repeats the process in the Slave timing chain. *Master.*

Bit 8 – CF: Clear Event Fifo. When set high, the Event fifo is flushed, and the Event fifo overflow counter is reset to zero. *Master.*

Bit 9 – IE: Ignore Errors. When set high, the STDU will ignore any timing link errors for approximately 3.5 minutes. ***This bit should not be set by the user.*** It will be set by firmware when a Scrub Local Unit command is issued to a MTDU. When a scrub command is issued to a MTDU, false timing link errors will be flagged in the STDU. Telling the STDU to ignore these errors until the timing link system is stabilized is the solution.

Bit 10 – RD: Reset Delay Registers. When this bit is set high, all of the registers associated with the timing link delays are reset to all zeroes. When this reset occurs, the new values (all zeroes) are not written to the FPGA eeprom as they are during a "delay learn" or manual re-write of delay registers.

Bit 11 – LD: Load Delay Registers. Setting this bit will generate an interrupt to the ARM processor. The ARM will respond by loading all of the registers associated with timing link delay register values from the FPGA's eeprom.

Bit 12 – SLN: "Scrub" Local Unit, No Load. This action is the same as setting bit 0 with the exception that the Delay register values that are saved in FPGA eeprom are not loaded, resulting in the Delay register values being set to zero.
Slave and Master.

Bit 13 – SSN: "Scrub" Slaves in Timing Chain, No Load. This action is the same as setting bit 1 with the exception that the Delay register values that are saved in FPGA eeprom are not loaded, resulting in the Delay register values being set to zero. *Master*

Master.

Bit 14 – SAN: "Scrub" All Units, No Load. This action is the same as setting bit 2 with the exception that the Delay register values that are saved in FPGA eeprom are not loaded, resulting in the Delay register values being set to zero. *Master*
Master.

NOTE: When implementing the reset/scrub options in the Control2 register, the user should write only 1 bit in the Control2 register.

4.9 1PPS Verification – Two 16-bit registers – Low 0x0A, High 0x0B

Master Only. This applies to the Master TDU only. When requested by the ARM/PPC, the FPGA will initiate a counter that starts and stops on the rising edge of the 1PPS signal. The counter value will be saved to these registers. The counter values will be compared to previous counts to verify that the counts match within a certain range.

4.10 Interrupt Register – 0x0C

Master and Slave. When an interrupt is issued by the FPGA, the ARM/PPC will read this register to determine what action took place that initiated the interrupt. Interrupt definitions are as follows:

Bit 0 – 1PPS verification counter ready (Debug firmware only) *Master Only.*

Bit 1 – Beam Event. *Master Only.*

Bit 2 – Error

Bit 3 – Save Delay Register values to FPGA eeprom

Bit 4..8 – Not used

Bit 9 - Load Delay Register values from FPGA eeprom to delay registers

Bit 10..13 – Not used

Bit 14 – "Scrub" Reset, No Load. This bit should not be written by the user. If the user wishes to perform a "Scrub, No Load" reset on the local TDU, the user should set bit "12" high in the Control2 register. To scrub the entire chain, the user should set bit "14" high in the Control2 register of the Master TDU.

Bit 15 – "Scrub" Reset. This bit should not be written by the user. If the user wishes to perform a "Scrub" reset on the local TDU, the user should set bit "0" high in the Control2 register. To scrub the entire chain, the user should set bit "2" high in the Control2 register of the Master TDU.

Remember that the interrupt will only occur if the action is enabled in the Interrupt Mask Register. Once an interrupt occurs, it can only be cleared by setting the "Clear interrupts" bit in the Interrupt Mask/Enable/Test register. If the condition that caused the interrupt to occur still exists, another interrupt will be generated. To stop an interrupt causing event from

creating an interrupt, you must either disable the type of interrupt in the Interrupt Mask/Enable/Test register, or disable the specific signal causing the interrupt in the Error Mask register or the Event register

4.11 Early SYNC Value Register – 0x0D

Master Only. Any SYNCs that are sent out will precede the actual GPS 1 Pulse Per Second by the number of 128Mhz clock ticks contained in this register. For instance, if the value in the Early SYNC Value register is 0x1FD, the outgoing SYNC will precede the GPS 1PPS by 3.976 usecs (7.8125ns x 0x1FD).

4.12 Error2 Register – 0x0E

Error2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										GT	SS	DG	FE	PPS	NT

Master and Slave. The Error2 register is another error register that was added to allow additional error checking capability.

Bit 0: NT: NMEA Timeout
The GPS unit outputs a stream of NMEA data in serial format that contains GPS timing and location information. Occasionally, the NMEA stream will hang for unknown reasons. This error bit will go high when the stream has been inactive for an extended period. *Master*

Bit 1 – PPS: 1 Pulse Per Second Test
The 1 Pulse Per Second accuracy test has detected a value outside the acceptable parameters of accuracy. *Master*

Bit 2 – FE: Framing Error
When a command packet is sent down the timing link and received by a slave, the slave decodes the packet. The packet should consist of six 8-bit words – a header, two address words, two data words, and a checksum. If the slave detects command packets that are not complete, this bit will be set. *Slave*

Bit 3 – DG: Dynamic GPS Loss of Lock Error
If the GPS loss of lock signal is set by the GPS unit, this bit will be high. If the loss of lock signal is low, this signal will be low. This is different than the GPS loss of lock signal in Error register 1 in that the Error register 1 flag is static and will stay set until reset by the user. *Master*

Bit 4 – SS: Time Sync bit in control register is stuck
An error condition can occur if a process in the power pc writes the control register during a time synchronization. The power pc process can modify the control register contents incorrectly resulting in a hang of the time synchronization process. The user should *always* check the status register before accessing registers to make sure that the FPGA on the TDU

isn't in the process of doing a time sync. This bit will be set if the time sync process got into the hung state. The firmware will recover from the hang, but the time sync will not be performed and the user must execute another time sync. *Master*

Bit 5 – GT: GPS Time of Week error
If a new GPS time of week value did not get written to the TOW register when requested during a time sync, this bit will be set.

4.13 Error2 Disable Register – 0x0F

Master and Slave. By default, all error conditions in the error2 register are enabled. By writing a "1" to a bit location in the Error2 Disable register, the user will disable notification of an error that is defined by that bit location. The Error2 Disable bit definitions are exactly the same as the Error2 register.

4.14 Preset Time Register – Bytes 1/0 0x10, Bytes 3/2 0x11, Bytes 5/4 0x12, Bytes 7/6 0x13

Master Only. When the user either requests a NOvA system time initialization or requests a reload of the NOvA time, a request is sent by the FPGA firmware to the GPS unit asking for GPS Time of Week and GPS week number. The FPGA firmware converts these numbers into the NOvA experiment time. This time conversion is then stored in 4 16-bit registers which are accessible via normal register reads. The four registers each contain ¼ of the NOvA time and are broken down into bytes 1/0, 3/2, 5/4, and 7/6. The upper 8 bits of the "Byte 7/6" register is padded with zeroes, resulting in the actual 56-bit NOvA time.

4.15 Error Register – 0x14

Error Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IS	SE	NS	NO	FF	BP		TP	GA	GH	PLL	NPPS	CE	NLC	NSC	NK

Master and Slave. The error conditions shown below will be latched on every 72 MHz ARM clock. A "1" in the bit location indicates that the particular error is present. Once an error occurs, the error bit is latched into the error register and can only be cleared by writing a "0" to that bit location, or by writing a "1" to the "Clear error register" bit in the control register.

Nibble 0 (Bits [0..3]) refers to the Timing Link signals.
Nibble 1 (Bits [4..7]) refers to the GPS signals.
Nibble 2 (Bits [8..11]) refers to Accelerator signals.
Nibble 3 (Bit [12..15]) refers to miscellaneous errors.

Bit 0 – NK: No comma character in 1024 transfers

Slave TDUs will reset and start a counter each time they receive an 8b/10b comma character. The counter will reset and restart every time it sees a comma character. If the counter value reaches 1024 (0x400), this error bit will be set. *Slave*

Bit 1 – NSC: No SYNC Echo
When the SYNC signal is sent out, a counter will reset and start, and will continue incrementing until the TDU sees the SYNC Echo signal. If a SYNC Echo is not seen before the counter reaches a certain timeout value, this bit will be set. The timeout value has yet to be determined. This bit will only be set in the Slave TDU when NONE of the downstream TDUs and DCMs return a SYNC echo. *Master and Slave*

Bit 2 – NLC: No Timing Link Clock Present
The Timing Link utilizes a 32Mhz clock. The TDU has an onboard oscillator that operates at the same frequency, 32Mhz. The onboard oscillator clocks a 3-bit shift register that will, if not reset by the Timing Link clock, set the final output of the shift register to a value of "1" signifying a missing Timing Link Clock. *Master and Slave*

Bit 3 – CE: Checksum Error
If the checksum received on the timing link at the end of the command doesn't match the checksum that is calculated onboard the Slave TDU, this bit will be set. *Slave*

Bit 4 – NPPS: GPS 1PPS Not Present
The GPS 1PPS signal is an extremely accurate 1 pulse per second clock generated by the GPS unit. A 26-bit counter that utilizes the 32Mhz clock will be set to the value 0x2000000 and decremented. At the rising edge of each 1PPS signal the counter will be re-loaded and restarted. If the counter reaches 0x0000000, a period greater than 1 second has passed, and this bit will be set. *Master*

Bit 5 – PLL: GPS Lost Satellite Lock
This applies to the Master TDU only. This signal originates at the GPS unit and is fed directly into this error register. If the GPS antenna loses lock with the GPS receiver, this signal will go active. *Master*

Bit 6 – GH: GPS Holdover Active
This signal originates at the GPS unit and is fed directly into this error register. The GPS unit will go into Holdover mode and the Holdover signal will go active if the GPS antenna loses lock. Holdover mode uses an internal clock reference that allows continuation of GPS functions. *Master*

Bit 7 – GA: GPS Antennae Fault
This signal originates at the GPS unit and is fed directly into this error register. This antenna fault signal goes active if there is a problem with the GPS antenna voltage. *Master*

Bit 8 – TP: TCLK Parity Error
Master

~~Bit 9 – NR: TCLK and/or MIBS Not Present~~
~~*Master*~~ Moved to Status register.

Bit 10 – BP: MIBS Parity Error
Master

~~Bit 11 – FF: Event Fifo Full~~
~~Valid accelerator events are stored in a fifo so as to eliminate the possibility of missing an event due to readout delay. If Run Control is not servicing interrupts, the fifo will eventually fill up. If this occurs, this bit will be set.~~ *Master.* This bit was removed and is now set in the status register.

Bit 12 – NO: No value in offset register
During a Time of Flight/2 calculation, the Slave TDU didn't see a new Delay Offset value written into the Delay Offset register. A value must be written before a Delay "Learn" is done. This error should not occur since Delay Offsets are handled by firmware, but is flagged just in case. *Master*

Bit 13 – NS: No SYNC
During a Time of Flight/2 calculation, no SYNC pulse was seen from the upstream TDU. *Slave*

Bit 14 – SE: SERDES Error.
Indicates a SERDES frame error for the current word being received. *Master and Slave*

Bit 15 – IA: Illegal Register Access
Someone tried to access the TDU type register on the slave via a timing link command. *Master and Slave*

4.16 Firmware ID Register – 0x15

Master and Slave. This register holds the version of firmware that is currently loaded on the TDU. Since STDUs and MTDUs have different firmware, they will each have different firmware versions.

4.17 TDU Type Register – 0x16

Master and Slave. TDUs can be configured as one of four possible TDU types. There are Masters and Slave, each of which can host either a copper timing link or fiber timing link. They are described in the TDU Type register as one of the following:

0 – Standard Slave (Copper timing link Slave)
1 – Serdes Slave (Fiber timing link Slave)
2 – Far Master (Copper timing link Master)
3 – Near Master (Fiber timing link Master)

4.18 Event Fifo Overflow Count – 0x17

Master. If the user doesn't read out the Event timestamp registers in a timely manner, the Event fifo will overflow. When this occurs, the "Event Fifo Overflow Count" register will be incremented each time an event is missed. The max value of this register is xFFFF.

4.19 Error Disable Register – 0x19

Master and Slave. By default, all error conditions in the error register are enabled. By writing a "1" to a bit location in the Error Disable register, the user will disable notification of an error that is defined by that bit location. The Error Disable bit definitions are exactly the same as the Error register.

4.20 PPS Width Register – 0x1A

Master. This register provides an option to change the width of the 1PPS that is output on the AUX front panel connector. Each bit value in the register is worth 7.63usecs. The full value (xFFFF) will give a little over 1/2 second 1PPS pulse width on the AUX output.

4.21 Event Timestamp Registers – Bytes 1/0 0x1B, Bytes 3/2 0x1C, Bytes 5/4 0x1D, Bytes 7/6 0x1E

Master Only. When an Accelerator event occurs, and the event is of interest to the experiment, the event number is saved to the event number register. Additionally, the time that the event occurred is saved to 4 Event Timestamp Registers. The timestamp is broken down into 4 16-bit registers designated as bytes 1/0, 3/2, 5/4, and 7/6. ~~Since the timestamp is only 56 bits, the value of byte 7 will always be zero.~~ Starting with version 2.10 MTDU firmware, the Event time is 57 bits. This was changed to allow 128Mhz (7.8125ns) Event timing resolution.

4.22 Event Number Register – 0x1F

Event Number Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE	0	0	0	0	BNB	MIBS	TCLK	Accelerator Event Value							

Master Only. When an accelerator event occurs, and the event type is enabled in the event mask register, the event number, event source, and parity information is loaded into the Event Number Register. The lower 8 bits are the event number (value), bit 15 is the parity error bit ('1' = error), and bit 8, 9, or 10 is the event type. TCLK = '001', MIBS = '010', BNB = '100'. Holger Myer requested that the BNB event always be loaded as x"041B". This has been implemented in the firmware.

Note that the fifo advances another timestamp package into the timestamp registers after a read of the Event Number Register. Therefore, the user should follow the proper method for reading out the event timestamp and the event number register. The correct sequence is for the user to first read out all four of the timestamp registers, and finish by reading the event number register. Failure to follow this readout sequence could lead to skewed event information.

4.23 Command Data, Address, Header Registers – Data 0x20, Addr 0x21, Header 0x21

Master Only. These 3 registers are grouped together here because they are always used as a group. The contents of these registers is sent down the timing link in 8b/10b/ format and decoded by the Slave TDUs, DCMs, and FEBs. There is an exact method for writing these registers. The Data register is written first, the Address register is written second, and the Header register is written last. When the Header register is written, the logic in the Master FPGA senses that write and takes the data in the 3 registers, converts it to 8b/10b format and sends it in a serial stream down the timing link. This 8b/10b serial stream transitions on the falling edge of the Timing Link Clock (32Mhz). This allows downstream Slave TDUs, DCMs and FEBs to use the rising edge of the clock to cleanly capture the data in the serial stream.

The user must be aware of the limitations of writing successive sets of commands to these registers. A serial stream of command data, address, and header words requires transferring 60 bits serially at 32Mhz. This transfer time is ~ 2usec (60 x 31.25ns = 1.875usecs). Therefore, the user should wait at least 2usecs between commands to assure that errors do not occur. The user should monitor and verify that the "Command Busy" and the "No Command Warning" bits in the Status Register are not set before issuing commands.

4.24 Event List Mask Register – 0x23

BNB				BS Clk (MIBS)				TCLK							
			$1B				$74	$A5	$A4	$00	$8F	$1D	$1F	$AD	$A9
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Master Only. The user must set the appropriate bits to allow the above listed events to be monitored. If one of the enabled events is detected, an interrupt will be issued, depending on how the Interrupt Mask register is set

4.25 Interrupt Mask/Enable/Test Register – 0x24

Interrupt Mask Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SR	SRN					LD	CI	FI	~~ARM3~~	ARM2	PPC	SD	ERR	BE	PPS

Master and Slave. This register serves three purposes. 1) Mask that allows certain actions to generate an interrupt. 2) Enables for each of the 3 physical interrupts. 3) Force an interrupt.

Writing a '1' to bit "0" will allow a 1 Pulse per second GPS counter to generate an interrupt, a '1' to bit "1" allows Beam Events to generate an interrupt, and a '1' to bit 2 allows Errors to generate an interrupt. Writing a '1' to bit "3" allows the

Writing a '1' to bit location 4..6 will enable those particular interrupts

Writing a '1' to bit location 7 will force an interrupt to whichever interrupts are enabled.

Bit 0 – 1PPS verification counter ready (Debug firmware only)
Bit 1 – Beam Event
Bit 2 – Error
Bit 3 – Save Delay register values to FPGA eeprom (ARM interrupt 2 only)

Bit 4 – Enable PowerPC interrupt4
Bit 5 – Enable ARM interrupt2
~~Bit 6 – Enable ARM interrupt3~~
Bit 7 – Force an interrupt on whatever interrupt is enabled.

Bit 8 – Clear whatever interrupts are asserted
Bit 9 – Load Delay register values from FPGA eeprom (ARM interrupt 2 only)

Bit 10..13 – Reserved
Bit 14 – "Scrub" reset, No Load. This bit should not be written by the user. If the user wishes to perform a "Scrub" reset on the TDU, the user should set bit "12" high in the Control2 register.

Bit 15 – "Scrub" reset. This bit should not be written by the user. If the user wishes to perform a "Scrub" reset on the TDU, the user should set bit "0" high in the Control2 register.

Ex. – Enable "Beam Events" and "Errors" for an interrupt to the Power PC.
b"0000 0000 0001 0110", x"0016".

4.26 GPS Lock Loss Counter – 0x25

Master Only. When a GPS loss of lock occurs, a counter in the firmware is incremented. The counter value is loaded into the GPS Lock Loss Counter register. The register will be cleared and the counter reset to 0x0000 when either a write to the register occurs, or the counter value reaches 0xFFFF.

4.27 GPS Holdover Counter – 0x26

Master Only. When a GPS holdover occurs, a counter in the firmware is incremented. The counter value is loaded into the GPS Holdover Counter register. The register will be cleared and the counter reset to 0x0000 when either a write to the register occurs, or the counter value reaches 0xFFFF.

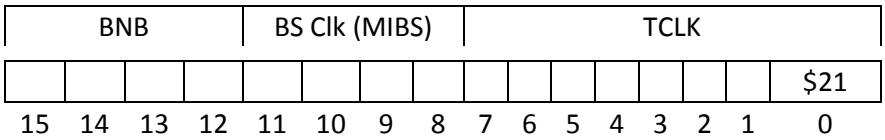
4.28 GPS Antenna Fault Counter – 0x27

Master Only. When a GPS antenna fault occurs, a counter in the firmware is incremented. The counter value is loaded into the GPS Antenna Fault Counter register. The register will be cleared and the counter reset to 0x0000 when either a write to the register occurs, or the counter value reaches 0xFFFF.

4.29 GPS Lock Loss Timer – 0x28

Master Only. This register keeps track of the number of seconds that a loss of GPS lock occurred.

4.30 Event List Mask2 Register – 0x29



Master Only. The user must set the appropriate bits to allow the above listed events to be monitored. If one of the enabled events is detected, an interrupt will be issued, depending on how the Interrupt Mask register is set

4.31 Command, Address, Header, Timestamp, History Registers - Time B1/0 0x30, Time B3/2 0x31, Time B5/4 0x32, Time B7/6 0x33, Data 0x34, Addr 0x35, Header 0x36

Master Only. Whenever a command is sent out on the timing link, a duplicate of that data, address, and header, along with a timestamp of when the command was issued, is saved to the History registers. The user can retrieve the timing link command history information by reading out the history registers. The proper method for retrieving the information is to start by reading the Command History Time, Byte 1,0, increment the address and repeat until the Command History Header has been read. Immediately after reading the Command History Header register, the next saved history command block of information is loaded into the command history registers. If the Command History Header register is not read out last, unread data in any of the others history registers will be overwritten by the next saved command history block.

4.32 Command History Word Count Register – 0x37

Master Only. As timing link commands are issued, and duplicates of these commands are loaded into the command history registers, this register will keep track of the number of words in the command history fifo that stores commands until they have been read out of the command history registers.

4.33 Decoded Time Register – Byte 1/0 0x40, Byte 3/2 0x41, Byte 5/4 0x42, Byte 7/6 0x43

Master Only. When the user either requests a NOvA system time initialization or requests a reload of the NOvA time, a request is sent by the FPGA firmware to the GPS unit asking for GPS Time of Week and GPS week number. The FPGA firmware converts these numbers into the NOvA experiment time. This time conversion is then stored in 4 16-bit registers which are accessible via normal register reads. The four registers each contain ¼ of the NOvA time and are broken down into bytes 1/0, 3/2, 5/4, and 7/6. The upper 8 bits of the "Byte 7/6" register is padded with zeroes, resulting in the actual 56-bit NOvA time.

4.34 NOvA Time Snapshot Registers– Byte 1/0 0x50, Byte 3/2 0x51, Byte 5/4 0x52, Byte 7/6 0x53

Master Only. A user can get a snapshot of the current NOvA time by reading this set of 4 registers. The user must read the register at address 0x50 first, as this read will latch the current time into all snapshot registers. Once address 0x50 is read, the remaining registers can be read out. Address 0x50 holds bits 55..48, 0x51 holds bits 47..32, 0x52 holds 31..16, and 0x53 holds 15..0. The user must have done NOvA time synchronization prior to accessing these registers.

5. Switches, LEDs and Display

5.1 Switches

5.1.1 Reset

There are two pushbutton reset switches on the front panel. The lower button marked "Reset" resets everything on the main board of the TDU. The upper switch marked "PPC Reset" resets components on the Power PC daughter board.

5.1.2 ARM ISP

When set to "on", the ARM ISP switch allows in system programming of the ARM via the USB port, and is disabled when set to "off".

5.2 LEDs

This manual covers both the MTDU and the STDU. The front panel LEDs differ in the two units. Each LED description mentions whether the LED is used on and MTDU, STDU, or both.

5.2.1 GPS Lock

Master - When lit, the GPS unit notifies the user that the GPS unit has satellite lock. This LED is a direct reflection of the GPS unit's "GPS Lock" signal. The LED will wait some number of seconds before lighting up after satellite lock has been achieved to guarantee stability of the satellite lock.

5.2.2 Holdover

Master – The GPS unit outputs a signal called "Holdover" when the GPS "Lock" signal goes inactive. The Holdover LED is a reflection of that signal.

5.2.3 Ant Fault

Master – When an antenna fault occurs on the GPS unit occurs, this LED will turn on.

5.2.4 GPS_TX

Master – The GPS unit outputs a serial NMEA data stream that contains timing information. If the NMEA stream is active, and not in a frozen state, this LED will flash.

5.2.5 GPS VCCs

Master – There are 3 LEDs on the GPS module that monitor power. VCC1, VCC2, and ANT_V should all be lit if the unit is functioning properly.

5.2.6 Accel

Master – When a TCLK and/or a MIBS accelerator signal is present, this LED will be on.

5.2.7 Fiber

Both - When the SERDES fiber links are being used, this LED will be lit if the SFP fiber modules are synchronized. These LEDs are relevant only to the Master and the first Slave TDU in the chain, as these are the only modules with fiber timing link connections.

5.2.8 Access

Both – If a user is writing to any of the registers, this LED will flash. This servers the purpose of visual verification, and may also alert the user that someone may be unintentionally accessing the registers.

5.2.9 Error

Both – If the error register(s) are non-zero, this LED will be lit

5.2.10 Parity

Master – If the TCLK and/or MIBS accelerator signals are connected to the MTDU and a parity error occurs on either of these signals, this LED will light.

5.2.11 Echo

Both – When a SYNC pulse is sent down the timing link , a return or "echo" SYNC pulse should be seen by all TDUs. If a SYNC goes out, but no echo from downstream STDUs is seen, this LED will light.

5.2.12 Comma

Slave – When commands are not being sent over the timing link, a filler word called a "comma" is sent. Also, a comma must be sent at least once every 1024 transfers so the firmware knows that alignment of the command stream is correct.

5.2.13 L_Clk

Slave – This LED will be lit if the clock is present on the timing link.

5.2.14 Alive

Flashes, indicating that the ARM "heartbeat" is active.

5.2.15 FPGA

Indicates that the FPGA is programmed. LED will be lit constantly.

5.2.16 Power OK

Signal from the onboard power supply

5.2.17 Misc. Power

There are 4 LEDs that check the status of misc power supplies on the TDU. When lit, the power is present, when not lit, there is a problem with that particular power supply.

5.2.18 Timing Link RJ-45

There are 2 LEDs on each of the timing link RJ-45 connectors. Masters are defined differently than Slaves.

5.2.18.1 Master

Timing Chain Out
- Red – Outgoing Sync pulse
- Green – Outgoing Command

Others – Undefined

5.2.18.2 Slave

Timing Chain In
- Red – Incoming Sync pulse
- Green – Incoming Command

Timing Chain Out
- Red – Undefined
- Green – Return (Echo) Sync seen

Timing Out Top
- Red – Undefined
- Green – Return (Echo) Sync seen

Timing Out Side
- Red – Undefined
- Green – Return (Echo) Sync seen

5.2.19 Ethernet RJ-45

The two Ethernet connectors on the front panel of the TDU each have to bi-color LEDs. The LEDs operate as follows:

Link LED (Left Side)		Activity LED (Right Side)	
Color	Definition	Color	Definition
Off	No Link	Off	No Activity
Amber	10 Mbps	Amber	Half-Duplex
Green	100 Mbps	Green	Full-Duplex

5.3 Display

The LCD display on the TDU is completely programmable. The information in this document reflects the latest screen definitions available. Different screens can be accessed by pressing either the top left or top right buttons next to the display screen. The display will show “TDU Status”, “Communications”, “GPS”, and “Nova TDU Board”. As the information on the display screen adequately describes its purpose, this document will not go into more detail of screen definitions.

6. Hardware Setup

1. TDU Unit should be attached to 120V AC source
2. Master TDU should have:
    a. GPS antenna attached
    b. SERDES SFP module installed, along with single mode fiber connected to downstream STDU. Otherwise, an RJ-45 copper cable connected to the "Timing Chain Out" connector to downstream STDU.
    c. TCLK signal from Accelerator
    d. MIBS signal from Accelerator
    e. BNB wall monitor signal
    f. Ethernet cable to both the "PPC Ethernet" and "ARM Ethernet" RJ-45 connectors.
    g. Optionally: Connect USB cables to "PPC Host Port" and/or "ARM Host Port".

3. Slave TDU should have:
    a. SERDES SFP module installed, along with single mode fiber from the Master TDU. Otherwise, an RJ-45 copper cable connected to the "Timing Chain In" connector whose source is the "Timing Chain Out" of the Master TDU.
    b. RJ-45 copper cable from "Timing Chain Out" to "Timing Chain In" of downstream STDU.
    c. RJ-45 copper cable from "Timing Out Top" to downstream top DCM "Timing in 0" (or 1).
    d. RJ-45 copper cable from "Timing Out Side" to downstream side DCM "Timing in 0" (or 1).
    e. Ethernet cable "ARM Ethernet" RJ-45 connectors.
    f. Optionally: Connect USB cables to "ARM Host Port".
    g. Note: If any of the "Timing Out" RJ-45 connectors are left unused, install a loopback connector in the unused RJ-45 connector.

7. Functional Requirements

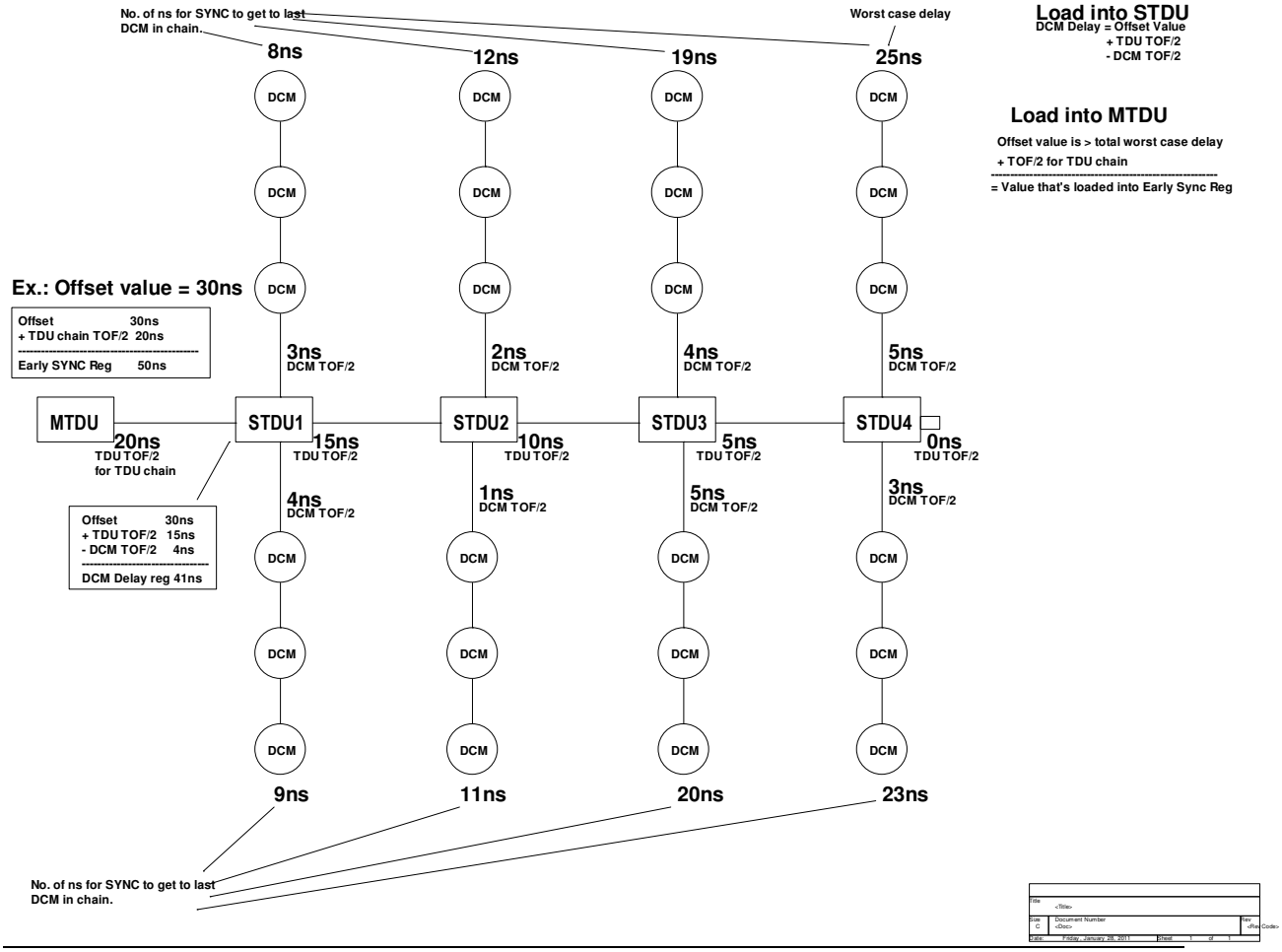
7.1.1 Power on Initialization

On power up, the timing link will remain static (undefined levels) until board initialization is complete. Once TDU initialization completes, it will drive SYNC to its negated level, start driving Clock with the front end clock frequency, and begin transmitting Comma characters on the Timing Link. (ref. Link Interface Specification for the NOvA Front End Board to Data Concentrator Module Connection).

Starting with version 2.19 of the Master firmware and version 2.04 of the Slave firmware, Delay values will be automatically loaded into the TDU's registers on power up or after a scrub. The Delay values will be stored in the FPGA's eeprom. This request was made to allow less expert intervention in the event of a power down of a TDU. If the Delay values in any of the registers that are used for the Delay process are changed (except in the case of a "reset" command to Control2 register), those modified values will be saved to the eeprom. The user should be aware that if a TDU is swapped out and used in another section of the detector, those older hardcoded Delay values will be used. This will result in erroneous delay timing. In this case, the user should either re-run the Delay calculation process, or manually reload Delay values in the proper registers.

Note to user: If either the SFP fiber modules or the fiber cables themselves are disconnected while the TDUs are powered on, the fiber link may get into an unstable state. If the fiber or SFP modules are disconnected while the TDUs are powered on, a "scrub", or manual reset should be executed to ensure proper fiber optic operation.

7.1.2 Execute TDU and DCM Timing Delay Calculation



Section 1: System wide delay calculation

Master TDU Delay Calculation Tasks

1. Wait for LE (Learn Enable) bit to be set in the Master's control register (Addr: 0x0000, Data: 0x0010).
2. Verify that the Delay Offset register contains a non-zero value. If its value is zero, flag an error and exit. The Delay Offset Value is a number that is *greater* than the worst case delay from Master TDU to last DCM in the longest chain measured in 128Mhz clock ticks. The user must enter the Delay Offset Value in the Delay Offset register before the System Delay calculations can be completed. By default, the Delay Offset register is loaded with a value of 0x200.
3. Set the CD (Calculating Delay) bit in the Status register.
4. Wait for a rising edge on the 1 Pulse per Second from GPS to assure a clean calculation.
5. Send Learn Enable command packet to Slave TDUs and DCMs. (Header: 0xC0, Addr: 0x0000, Data: 0x0010).
6. Wait 4usecs for the command to be completed.
7. Write the "Send Sync" bit in the control register
8. Send command packet with the Delay Offset Value to Slave TDUs. (Header: 0x80, Addr: 0x0008, Data: *Delay Offset Value*).
9. Wait for rising edge of GPS 1 Pulse per Second
10. Simultaneously send out a SYNC pulse on the timing link AND start a counter that will keep track of how many 128Mhz clock ticks occur until an Echo (or Return) SYNC pulse is seen.
11. When the Echo SYNC pulse is seen, stop the clock tick counter, and save the counter value. If no Echo SYNC pulse is seen after a reasonable amount of time, flag an error, clear the Calculating Delay bit in the Status register, The Learn Enable bit in the Control register, and exit.
12. Divide the Clock tick counter value by 2. Save the result in the TDU Delay register. This will give us our time of flight/2 (TOF/2), or the number of clock ticks, from the Master TDU to the Slave TDU that is the last in the chain of Slave TDUs.
13. Add the Delay Offset Register Value to the TDU Delay register value. Put the result in the Early SYNC register. The Early SYNC register lets the Master TDU know how many clock ticks *before* a rising edge of the GPS 1 Pulse per Second that a SYNC pulse should be sent out on the timing link. By the time the SYNC pulse travels through timing link wires, fiber optic cables, and electronic logic, the SYNC will arrive at the DCMs at what is the true 1 second rollover. Additionally, the SYNC will all arrive at all of the DCMs simultaneously.
14. Clear the Calculating Delay bit in the Status register, and the Learn Enable bit in the control register.

Slave TDU Delay Calculation Tasks

1. Wait for LE (Learn Enable) bit to be set in the Slave TDU control register by the Master via a timing link command (Header: 0xC0, Addr: 0x0000, Data: 0x0010).
2. Wait for a Delay Offset value to be sent over the timing link (Header: 0x80, Addr: 0x0008, Data: *Delay Offset Value*). Start a timeout counter that will set an error flag if the Delay Offset doesn't come within 1 second.

3. If the Delay offset value is greater than zero, load the Delay Offset value into the Delay Offset register. If the delay value is zero, flag an error and exit.
4. Set the Calculating Delay bit in the Status register.
5. Wait for a SYNC pulse from the incoming timing link.
6. When the SYNC pulse occurs, start 3 separate counters that count the number of clock ticks until a return SYNC pulse is seen from the 1) downstream Slave TDUs, 2) Top DCMs, and 3) Side DCMs. Be sure to wait until all return SYNCs are seen.
7. If a return SYNC isn't seen in ~ 32usecs from the downstream TDU, flag an error and exit.
8. If a return SYNC isn't seen from either the top or side DCMs, flag an error, but continue. It's possible that only one of the timing link channels may be hooked to a chain of DCMs, and we don't want to kill the entire process. Be sure to load the value of "2" into the write-to-read delay for the fifo that handles the DCM chain that didn't issue a return SYNC. "2" is the minimum write-to-read-delay value that the fifo will accept. However, if both DCMs do not return a SYNC, flag an error and exit.
9. Save all three time values to 3 different temporary registers.
10. Divide the saved times by 2 to get the delays for each link.
11. Verify that the write-to-read delays in the DCM fifos are set to "2". This is required to make sure that every chain calculation starts out with the same delay parameters.
12. Add the offset loaded earlier from the Master TDU to the TDU TOF/2, and subtract the DCM TOF/2. Do this for both the top and side DCMs. These will be the delay values for the DCMs.
13. Save these two values to the top and side DCM delay registers.
14. Disable communication to DCMs so no garbage data is seen at the DCMs.
15. Set the top and side DCM fifo's write-to-read delay using the value in the delay registers.
16. Enable communication to DCMs.
17. Wait 1 millisecond for the DCM to re-synchronize itself to the timing link stream
18. Clear the "Calculating Delay" bit in the status register.
19. Set the "Delay Calculated" bit in the status register.

7.1.3 Manually Set TDU Delay values

The user may use this utility when he has a set of saved values for the delay calculations. The instructions shown below do not make any changes to DCM delay values.

Master:

1. Re-write the Delay Offset register (Addr 0x0008).

2. Re-write the Early Sync register (Addr 0x000D).

3. Re-write the TDU Delay Value register (Addr 0x0002).

4. Wait for system to stabilize. The previous step will stop, modify, and restart the TDU and DCM timing link.

Slave:

5. Re-write the Delay Offset register (Addr 0x0008).

6. Re-write the TDU Delay Value register (Addr 0x0002).

7. Re-write the Side DCM Delay Value register (Addr 0x0003).

8. Re-write the Top DCM Delay Value register (Addr 0x0004).

9. Set bit 13 of the Control register (Addr 0x0000) to '1'. This bit will clear after completion.

10. Wait for system to stabilize. The previous step will stop, modify, and restart the DCM timing link.

7.1.4 GPS Communication

7.1.5 Interrupt Handling

As of this writing, only two events will generate an interrupt:
A Beam Spill
A "Count Complete" that occurs during a testing of the accuracy of the GPS's 1PPS signal Vs. the GPS's 10Mhz clock. This event occurs only during debug and testing.

~~Verify that ARM/PPC is not asserting the "ARM(PPC) requests FPGA" signal.~~
Write the appropriate data value into the appropriate register
Set appropriate bit in interrupt register.
Generate an interrupt by setting the "interrupt request" signal to "1".
Wait for the ARM/PPC to assert the "ARM(PPC) requests FPGA" signal.
Set the "FPGA grants ARM/PPC" signal.
Verify that the address that is being accessed is the interrupt register
Reset appropriate bit in interrupt register
Clear interrupt.
Wait for the ARM/PPC to de-assert the "ARM(PPC) requests FPGA" signal.
Clear the "FPGA grants ARM/PPC" signal.

7.1.6 Verify the 1PPS vs. 128Mhz clock timing (Debug Mode)

The 1 pulse per second (1PPS) signal from the GPS unit is synchronized with the GPS's 10Mhz clock. To verify the accuracy of the 1PPS, use the GPS 10Mhz signal multiplied to 128Mhz to clock a 32-bit counter that starts on the rising edge of the 1PPS signal. The counter will stop on the next rising edge of the 1PPS signal. The counter value will be written to a register. The FPGA will send an interrupt to the ARM to let it know that a count is complete. The ARM will service the interrupt and read out the register that contains the count. The FPGA will clear the counter and restart the process, waiting for the next rising edge of the 1PPS. This will give us a reading every 2 seconds.

7.1.7 Send Individual Commands on the Timing Link

The MTDUs drive the Command, Sync, and Clock signals, and are passed daisy chain fashion through the chain of STDUs and DCMs. Commands are grouped as command frames. Command frames have the form of a one byte header (with leading byte padded with zeroes), followed by two bytes of address and two bytes of data. A checksum byte is added at the end of the frame as a method of checking the accuracy of the contents of the command frame.

Commands are issued by writing the set of command registers in a specific order. Data register first, then address register, and finally header register. Upon completion of the write of the header register, then entire command frame is sent down the timing link to the Slave TDUs which will keep them, or pass them on to the DCMs and FEBs depending on the content of the header word. User should be careful to always check that the "Command Busy" and "No Command Warning" bits are not set in the Status register before issuing commands.

Users can generate their own command frames, or use one of the predefined commands listed below.

7.1.8 Calculate Timing System Delay

After power-up initialization, the user should execute the command to determine and adjust for the timing delays in the timing link chain. This requires the user to set the "Learn Enable" bit in the control register that will initiate a process inside the FPGA that manages all calculations, commands, and syncs necessary to complete the delay calculation task. Be aware that this process will take several seconds to complete. The user should monitor the "Timing Link Busy" bit (15) in the status register before attempting any register/timing link access.

When the "Learn Enable" command is issued to the, two actions occur. The first is the calculation of the delay in the TDU to TDU timing chain. The second is the individual Slave TDU to DCM delay calculation. The delay calculation starts when the Master TDU sends a Sync pulse to the 1st Slave TDU in the chain. The 1st Slave TDU starts a counter using the ~128Mhz PLL clock and sets the "Calculating Delay" bit in the status register. This Sync pulse is repeated and sent on to the next Slave in the chain, and so on. Each of the TDUs will increment their own counters. The last Slave TDU in the chain will return the Sync pulse as the signal "SYNC Echo" back up the TDU chain to the Master. As each of the Slave TDUs see the Sync Echo pulse come back to them, they stop and store the counter value. The Sync Echo pulse is finally returned to the Master TDU where it stops its own counter and saves the counter value.

At the same time that the TDU to TDU delays are calculated, each Slave TDU calculates the TDU to DCM delay. When the Slave TDU starts the DCM delay calculations by sending out a Sync pulse, it also starts a counter using the ~128Mhz PLL clock. The counter will continue to increment until the TDU sees a Sync Echo pulse. Once the echo pulse is seen, the counter stops, is divided by 2, and loaded into a fifo that delays clocks, commands, and syncs to downstream DCMs for that number of clock ticks. The 16-bit value of the register will allow a delay count of up to 512 usecs.

When the Master TDU has seen the Sync Echo pulse, it will calculate the TOF/2 for the entire TDU chain. It will then add that TOF/2 value to the value in the "Delay Offset" register and put the result in the "Early Sync" register. The "Early Sync" register determines how early a Sync pulse will be sent down the timing link chain.

If this counter reaches maximum value, the firmware will assume that the Sync Echo pulse is not coming, and will set the "No SYNC Echo" bit in the error register. The "DCM Delay Value" register is cleared on a reset or a "Calculate Delay" command.

The user should monitor the "Delay Calculated" bit in the status register of the Master TDU. The delay calculation process may take a few seconds to complete, so the user should be patient when monitoring the "Delay Calculated" bit.

Users should verify that timing link loopback connectors are installed on any unused timing link outputs of both STDUs and DCMs before executing this command. Failure to do so will result in inaccurate delay calculations.

After the delay calculation as completed, the Error LEDs will be active on the Slave TDUs. This is due to the fact that timing link clock must be stopped to initialize the new Sync pulse. The user should clear the error registers after a delay calculation has been performed. This can be achieved by writing a value of 0x0000 to address 0x0014, using a header value of 0x0080.

7.1.9 Load NOvA Time

After the delay calculations have completed, the user should synchronize the timing of the STDUs and DCMs by setting the "Time Synchronization" bit in the control register. A process will launch that retrieves the current time from the GPS unit, converts it to NOvA time, loads downstream TDUs and DCMs with that preset time, and sends a Sync pulse that informs all units on the timing link chain to start their NOvA time clock counters. Because of the delay calculations done previously, the values of all NOvA time clock counters on all DCMs will be identical.

7.1.10 Decoding Accelerator Signals

There are four Accelerator signals that are present on the TDU.
RFCLK, which is the 53.10468Mhz clock

Decode Beam Spill(?) from TCLK

Grab and save time of day value to register
Set Interrupt.

See:
Time and Data Distribution Systems at the Fermilab Accelerator
David G. Beechy and Robert J. Ducar

7.1.11 NOvA Counter Initialization

The NOvA timer is a counter that counts the number of 64Mhz clock ticks since NOvA time zero, which was Jan. 01, 2010 00:00.00 UTC. The NOvA counter can be initialized in one of two ways.

After a power up reset, the TDU's FPGA waits for a signal from the ARM that FPGA can set an interrupt for the ARM requesting that the GPS Time of Week (TOW) and GPS week number be written to their appropriate registers. Once these writes have occurred, a flag is set for the counter initialization process to proceed.

The other option requires that the user set the appropriate bit in the control register to start the initialization sequence. A separate bit in the control register will determine whether or not the NOvA counter is automatically reinitialized on each GPS 1 pulse per second, or only when the user writes a bit to the command register.

8. TDU System Initialization

The following are the steps that should be taken to initialize the TDUs and DCMS via the timing link. These actions should be performed after all cabling has been installed and all units are powered up and in their quiescent states. DCMs must be in external timing mode. Refer to the DCM user's manual for directions on register settings and initialization for the DCM.

8.1 Reset the TDU

The TDU can be reset by cycling the power on the TDU, pressing one of the Reset buttons on the front panel of the unit, or by writing a value of x"0001" to the Control Register (0x00). If the TDU has been changed from a SERDES (fiber) to copper timing link, or vice versa, a power on reset or pushbutton reset must be performed. Writing the reset bit in the Control Register will not sense the timing link type change.

8.2 Read the Status Register

Read the contents of the Status Register (0x01). Verify that the "OK to Sync" bit is set. If not wait a few seconds and try again.

8.3 Set Error Disable Register

By default, all errors in the Error Register are enabled. The Error Disable Register allows the user to disable individual error flags. The user should write a value of "1" to any bit location in the Error Disable Register at (0x19) to disallow flagging of that error. The error bit locations in the Error Disable Register match the error bit locations in the Error Registers. Also note that the Error Mask Register is no longer used.

8.4 Clear/Read the Error register

After the TDUs have been powered up and reached their quiescent state, the user should clear the error register in the event that power initialization has left false positive error bits set in the error register. Any errors that appear after the clear are genuine errors.

1. Read the status register. Verify that the "command busy" bit and the "no sync warning" bit is not set.
2. Write the Error register (0x14) with the value x"0000".
3. Read the Error register and respond accordingly.

8.5 Calculate Timing System Delays

An automatic calculation of the cable and logic delays throughout the entire timing system can be performed with the setting of one bit in the control register. Warning: As mentioned previously, verify that unused "timing out" links on STDUs and DCMs have the required loopback connector installed.

1. Read the status register. Verify that the "command busy" bit and the "no sync warning" bit is not set.
2. Write the Control register (0x00) with the value x"0010".

This action may take a few seconds to complete and it is therefore recommended that no activities be performed on the timing link while the system does these calculations.

8.6 Re-Sync System to NOvA Time

After timing system delays have been calculated, the timing system needs to be re-synced to NOvA time.

1. Read the status register. Verify that the "command busy" and the "no sync warning" bits <u>are not</u> set, and the "delay calculated" and the "okay to sync" bits <u>are</u> set.
2. Write the Control register (0x00) with the value x"0020".

8.7 Set up the Event List Mask Register

If the TDU being initialized is the Near Master, the user should determine which accelerator events are to be monitored. Set the appropriate bits in the Event List Mask Register. By default, no events are enabled. To enable all of the events that can be monitored, write the value 0x110F to the Event List Mask Register (0x23).

8.8 Enable Accelerator Events

Tracking of accelerator events will only occur when the "Enable Events" bit is set in the Control Register. Be aware that once the user sets this bit, any further writes to the Control Register must have this bit set. Failure to do so will disable accelerator event tracking. This action is only necessary for the Near Master as it is the only unit that is cabled to accelerator signals.

8.9 Enable Interrupts

Setup interrupts using the Interrupt Mask Register (0x24). Errors and Beam Events can be enabled/disabled, and interrupts can be directed to either the Power PC or the ARM.

9. Misc TDU Operation Examples

9.1 Send a command on the timing link

A command block may be sent out on the timing link by:

1. Read the status register. Verify that the "command busy" bit and the "no sync warning" bit is not set.
2. Write the Command Data register (0x20) with the appropriate data value.
3. Write the Command Address register (0x21) with the appropriate address, usually a register.
4. Write the Command Header register (0x22) with the appropriate header information. Header information is encoded such that each unit type has its own unique bit defined for access. The STDU is x"0080", the DCM is x"0040", and the FEB is x"0020". These bit locations can be combined to communicate with multiple unit types at the same time. Ex. x"00E0" will write the same data at the same address to all STDUs, DCMs, and FEBs.

After the Command Header register is written, the entire command block is sent down the timing link to the units that were addressed.

9.2 Re-initialize the NOvA Timer Without Re-synchronization.

The NOvA timer can be re-initialized on just the MTDU without a corresponding re-synchronization of the entire NOvA system. Writing the "Re-initialize NOvA Timer" bit in the Control Register will prompt the TDU to get the current GPS time from the GPS unit, convert it into NOvA time, and load the Preset Time Registers (0x10, 0x11, 0x12, 0x13) with the new value. This operation is a good sanity check of the GPS unit.

9.3 Send a Sync pulse on the Timing Link

A single Sync pulse can be sent at the next 1 second boundary by writing the "Send Sync" bit in the Control Register.

9.4 Reset the GPS unit

For unknown reasons, the GPS unit on the MTDU will occasionally get into an unstable state. The user can return the GPS unit to normal operation by setting the "GPS Reset" bit in the Control Register. It may take up to a half minute for the GPS unit to fully respond to this reset, so patience should be exercised.

9.5 Read history registers

There is a set of registers on the MTDU that are record keepers of command that have been sent out on the timing link. Besides a record of the data, address, and header for the command block, there are also registers which contain a timestamp of when the command was sent.

9.6 Clear history registers

10. TDU USB/Ethernet Command Packet Format

Each TDU module has a Lantronix XPort Ethernet module that can be used to send the TDU commands. The TDU will only respond to commands sent by a host, it will never send data without getting a valid command first.

To send a TDU a command the host will send a data packet to port 10001. Two types of messages can be sent to the TDU; a command message and a data message. The command message protocol is a simple ASCII string like the example below:

$01*

The '$' char tells the TDU that the incoming message is a command message. The "01" is the command number, in this case a TDU_PING command. The command is a two char ASCII string that gives us a range of 99 commands (01-99). The '*' is the end of command string char. If the TDU replies to the command, the response will be in the form of unsigned char's ordered LSB to MSB.

The data message is used to send 256 bytes of data to the TDU. This data is the source data used to program a page in the FPGA's boot device. A data message has the following format:

%Byte0Byte1Byte2…Byte255

The '%' char tells the TDU that the incoming message is a data message. After the host sends the '%' data start char, it should send 256 bytes of data in the form of unsigned char's.

Here is a list of current TDU commands:

01 PING_TDU.

02 TDU_ID

03 TDU_STATUS

04 TDU_READ_REG

05 TDU_WRITE_REG

06 BOOT_FPGA

07 ERASE_DEVICE

08 READ_PAGE

09 WRITE_PAGE

10 VERIFY_PAGE

11 GPS_STATUS 2 (TDU Master only)

12 GPS_STATUS (TDU Master only)

13 GPS_COMMAND_SEND (TDU Master only)

14 GPS_COMMAND_READ (TDU Master only)

15 BEAM_CLK_STATUS (TDU Master only)

16 PROTECT_DEVICE

17 RELOAD_GPS_TIME (TDU Master only)

10.1 PING_TDU: $01*

This command is used to see if the TDU is on the network and receiving USB/Ethernet messages. The TDU responds with a '*' after it receives this command.

10.2 TDU_ID: $02*

This command is used to read out the TDU's various version & serial numbers. When the TDU receives this command it responds with 13 bytes:

Byte 8	Byte 7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
64 Bit TDU Serial Number MSB-LSB								TDU TYPE

Byte12	Byte11	Byte10	Byte9
FPGA Version Major	FPGA Version Minor	Software Version Major	Software Version Minor

Byte 0 TDU_TYPE:
M_TDU_SERDES = 0x03 M_TDU = 0x02 S_TDU_SERDES = 0x01 S_TDU = 0x00

Bytes 8:1 TDU_SERIAL:
64bit 1-Wire serial number

Bytes 10:9 TDU_SW_VERSION:
Byte 10 = Major version number Byte 0 = Minor version number

Bytes 12:11 TDU_FPGA_VERSION:
Byte 12 = Major FPGA version number LSB = Minor FPGA version number

10.3 TDU_STATUS: $03*

This command is used to read out the TDU's current operating conditions. When the TDU receives this command it responds with six bytes:

Byte7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
FAN_ERROR	FPGA_STATUS	TDU_PCB_TEMP MSB-LSB		TDU_SUPPLY_CURRENT MSB-LSB		TDU_SUPPLY_VOLTAGE MSB-LSB	

TDU_SUPPLY_VOLTAGEIN: Two bytes. Actual voltage is (16bit number * .00644).
TDU_CURRENT: Two bytes. Actual current is: (((16bit number * .00322) / 50) / .033)
TDU_PCB_TEMP: Two bytes. Temperature in Celsius is: (16bit number * .0625)
FPGA_STATUS: 0x00 = Not Booted 0x01 = Booted.
FAN_ERROR: 0x00 = Fan Normal 0x01 = Fan Error.

10.4 TDU_READ_REG: $04,REGISTER,*

This command is used to read a 16bit register in the TDU's memory map. The REGISTER field can be any ASCII number from 0x0000-0xffff. When the TDU receives this command it responds with the registers value split into two bytes, LSB then MSB.

10.5 TDU_WRITE_REG: $05,REGISTER,DATA,*

This command is used to write a 16bit register in the TDU's memory map. The REGISTER field can be any ASCII number from 0x000-0xffff. The DATA field can be any number between 0x0000-0xffff. When the TDU receives this command it responds with '*' char.

10.6 BOOT_FPGA: $06*

This command is used to re-boot the FPGA. The TDU responds with an '*' when the operation completed successfully, 0x00 when the operation failed.

10.7 ERASE_DEVICE: $07*

This command is used to erase the FPGA Boot Device. The TDU responds with an '*' when the operation completed successfully, 0x00 when the operation failed.

10.8 READ_PAGE: $08, PAGE,*

This command is used to read a 256 byte page of data out of the boot flash. The PAGE field can be any number between 0 and 0xffff. The TDU responds with 256 bytes of data. The LSB is sent 1st.

10.9 WRITE_PAGE: $09,PAGE,*

This command is used to write the WRITE_DATA_BUFFER to the selected page of boot flash memory. The PAGE field can be any number between 0 and 0xffff. The TDU responds with an '*' when the operation completed successfully, 0x00 when the operation failed.

10.10 VERIFY_PAGE: $10,PAGE,*

This command is used to verify the WRITE_DATA_BUFFER to data to the selected page in the FPGA boot flash. The TDU responds with an '*' when the operation completed successfully, 0x00 when the operation failed.

10.11 GPS_STATUS2: $11*

This command is used to read the status of the GPS unit (if installed). If a GPS unit is installed the TDU replies with 13 bytes:

Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
UTC TIME HR-MIN-SEC			FIX STATUS	SAT COUNT	TIMEOUT COUNT	GPS_TIMEOUT

Byte12	Byte11	Byte10	Byte9	Byte8	Byte7
UTC WEEK MSB-LSB		GPS TOW MSB-LSB			

Byte0 GPS_TIMEOUT:
This byte will be 0x00 when the GPS unit is sending NMEA data properly. It will read 0x01 when the GPS has stopped sending NMEA data for more than 5 seconds.

Byte1 TIMEOUT COUNT:
This byte holds the number of times the TDU has detected a GPS NMEA timeout.

Byte2 SAT COUNT:
This byte holds the current GPS locked satellite count. Can be any number between 0 – 12.

Byte3 FIX STATUS:
This byte holds the current GPS satellite fix status. Bits 1:0 of this byte contain the current GPS satellite fix status. FIX STATUS is obtained from the NMEA data stream not the LOCK bit out of the GPS unit.
Bits 1:0 – FIX_STATUS:
0x01: No Fix 0x02: 2D Fix 0x03: 3D Fix

Bytes 6:4 UTC TIME:
These three bytes hold the current UTC time obtained from the GPS units NMEA data stream.
Byte6: Hours Byte5: Minutes Byte4: Seconds

Bytes 10:7 GPS TOW:
These four bytes hold the current 32bit GPS TOW (time of week) value.
Byte10: MSB - Byte7: LSB

Bytes 12:11 GPS WEEK:
These two bytes hold the current UTC week obtained from the GPS units NMEA data stream.
Byte12: MSB - Byte 11: LSB

10.12 GPS_STATUS: $12*

This command is used to read the status of the GPS unit (if installed). If a GPS unit is installed the TDU replies with two bytes:

Byte 1	Byte 0
GPS_SATELITE_COUNT	GPS_STATUS_BITS

Byte 0: GPS_STATUS_BITS

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANTENNA_FAULT	GPS_HOLDOVER	LOST_LOCK	1PPS	Unused	FIX_STATUS		GPS_PRESENT

Bit0 - GPS_PRESENT:
Bit will be a ‘1’ when a GPS unit is present, and a ‘0’ when no GPS unit is present.

Bits 2:1 – FIX_STATUS:
0x01: No Fix 0x02: 2D Fix 0x03: 3D Fix

Bit 3 – Unused

Bit 4 – 1PPS:
Bit will be a ‘1’ when no 1PPS is detected, and a ‘0’ when detected.

Bit 5 – LOST_LOCK:
Bit will be a ‘1’ when the GPS unit has lost satellite lock, and a ‘0’ when locked.

Bit 6 – GPS_HOLDOVER:

Bit will be a '1' when the GPS unit is in hold over mode (no lock), and a '0' when locked.

Bit 7 – ANTENNA_FAULT:
Bit will be a '1' when the GPS unit has detected an antenna fault, and a '0' when running normally.

Byte 1: GPS_SATELLITE_COUNT

Bits7:4	Bits3:0
Unused	GPS_PRESENT

Bits 3:0 – GPS_SATELITE_COUNT:
This bits hold the current GPS locked satellite count. Can be any number between 0 – 12.

10.13 PROTECT_DEVICE: $16,LOCK_DATA,*

This command is used to set/reset a write/erase protect lock in the TDU's FPGA boot flash. To unprotect the boot flash the LOCK_DATA value must be 0x55. All other LOCK_DATA values will put the TDU in protected mode. The TDU responds with an '*' when the operation completed. The TDU powers up with the flash protect active.

10.14 RELOAD_GPS_TIME: $17*

This reloads the GPS TOW and UTC_WEEK FPGA registers. The TDU responds with a '*' after it receives this command.

11. Procedure for Programming the TDU's FPGA Boot Device

This procedure will work for both the TDU's USB host port or the XPort Ethernet interface. A server can connect to the TDU's Ethernet interface via port 10001.

1. The TDU's boot device must 1st be un-protected so it can be erased and re-written. Use the following command to do this:
   $16,0x55,*

   The TDU will reply with a '*' if the un-protect command was successful. Also when in un-protected mode the TDU's LCD displays back light will be RED.

2. Next the Boot Device must be erased. Use the following command to do this:
   $07*

   When the Boot Device has been erased, the TDU will reply with a '*' if successful, a 0x00 if not. The erase can take several seconds; make sure to wait for the '*' before moving on.

3. Now each page of the Boot Device needs to be programmed. Each page is 256 bytes and there are 8192 pages. The boot device data is read in from an "rbf" file generated by the Altera FPGA software. Each byte read from the rbf file must be bit swapped before

it is sent to the TDU; i.e. A byte read from the rbf file is 0x05 must be bit swapped to 0xA0. 256 bit swapped bytes must now be sent to the TDU. To do this send the TDU the following command:
%Byte0-Byte255
Once all 256 bytes have been sent, the TDU will reply with a 16bit checksum. The checksum is a standard CRC16 checksum and the LSB will be send 1st.

4. Next generate a checksum for the 256 bytes of data that was sent to the TDU and compare it to the checksum the TDU replied with. If it does not match, the 256 bytes need to be re-sent.

5. Now the page of data that was sent to the TDU must be programmed into a page of the Boot Device. To program page 0, send the following command:
$09,0x00,*

   When the page of data has been written to the Boot Device the TDU will reply with a '*' if successful, a 0x00 if not. If a page write error was received the entire Boot Device must be erased and the procedure started over from the 1st page of data.

Using the above procedure, program all the pages of data that have been read out of the rbf file. Once all pages have been written the TDU's FPGA must be re-booted.

6. To do this, send the following command:
$06*

   The TDU will reply with a '*' if the re-boot was successful, and the FPGA LED on the TDU's front panel should be on.

7. The last thing you need to do is to re-protect the Boot Device. To do this, send the following command:
$16,0xaa,*

   The TDU will reply with a '*' if the protect command was successful. The TDU's LCD's displays backlight should return to WHITE.

   You are finished…

12. Timing Link Pass-through Delay

The worst case delay path for all timing signals between the "Timing Chain In" and "Timing Chain Out" on the STDU is ~25ns. This includes LVDS transceiver delays, FPGA firmware delays, and routing delays.

13. TBD

14. TDU FPGA firmware "To Do" list

Master:

1. ~~Get the option to generate an early SYNC working. This will allow us to get the NOvA counter started at the 1 second rollover instead of 1 second + cable and chip delays.~~
2. ~~Correct drift on NOvA counter. No longer an issue according to Andrew Norman~~
3. ~~Modify the NOvA Time Zero constant after I get the documentation from Ron/Peter.~~
4. ~~GPS TOW not always loaded after power on reset. May be driver that is inadvertently hogging the register access bandwidth. (This problem may be fixed in version 01.06 of the Master firmware)~~
5. ~~Constantly check the time between GPS 1PPS.~~
6. ~~Add NOvA counter verification code checking.~~
7. Add "command write error" bit to status register
8. ~~Make sure that command packets have enough setup time at their destination before a SYNC is sent. A bit was added to the Status register to warn the user that the TDU is close to the time when a Sync pulse can be issued, and that they should not send a command block at this time.~~
9. ~~Add Auto/Manual Time Re-sync. Default to Auto re-sync. (Manual Re-Sync done in version 01.06). Not needed. Manual re-sync via the control register is all that's needed. And it's done.~~
10. Buffer SPI and ARM writes.
11. ~~Get Fiber working with Master/Slave~~
12. ~~Define LED functions~~
13. Determine what additional error checking needs to be done.
14. ~~Finish TOF/2 calculations. (Need to get slave operational first)~~
15. ~~Convert the SYNC pulse to a 16Mhz clock with an inserted 8Mhz period used to signify a Sync.~~
16. ~~Disable the Error Mask register, and instead create an Error Disable Mask register at a new address. This will allow all errors to be seen unless the user disables a particular error.~~
17. ~~Re-institute the NOvA time registers. Writing a bit will grab the current NOvA time and place them in 4 contiguous registers.~~
18. ~~The NOvA counter needs to initialize on startup, after a GPS lock has been established. The "OK to SYNC" bit in the Status register will go high after this occurs. Not needed. An option to initialize the counter via the control register is acceptable and is completed.~~
19. ~~Add check for MIBS signal. Send error (status?) bit if not present.~~
20. ~~Modify the History fifos and registers. Add time information.~~
21. ~~Remove TDU_TYPE reset that occurs when writing bit 0 of the control register. The TDU will set itself to a type "0" TDU (Copper Slave) if a bit 0 reset is done as it is currently configured.~~
22. Determine the reason for missing events at rate of about 1%.
23. ~~Change stack depth for accelerator events from 16 to 128. DONE~~

24. Add snapshot of NOvA counter initial value? Not sure if this is needed.
25. ~~Add GPS reset to pushbutton/soft reset? Problem is that GPS unit take about ½ minute to recover from reset. Can user be forced to wait?~~
26. ~~Add a Power PC interrupt when the GPS serial stream stops? User must service interrupt. NO, as determined by Andrew Norman.~~
27. ~~Review the firmware for loading accelerator events into the fifo. Is there a possibility of events from tclk vs MIBS colliding at the input of the fifo? Yes. Additional fifos were added so that collisions can no longer occur~~
28. ~~Check into parity errors on the accelerator clock signals. We are occasionally seeing them. Is this normal? These are bad events. Greg Vogel has been made aware.~~
29. Occasionally, the NOvA time counter loses 64usecs. A re-sync cures the problem.

Slave:

1. ~~Get the fiber working~~
2. ~~Find out why slave not passing CMDs to DCM~~
3. ~~Flip TX clock 180 degrees~~
4. ~~TOF/2 calculations~~
5. ~~Decode 8b/10b stream~~
6. ~~Verify that PLL clock switching between fiber and copper link is done correctly and automatically.~~
7. ~~Change SYNC pulse decoding to look for 8Mhz period in a 16Mhz clock.~~
8. ~~Recover from Master reset on the fiber link automatically. Currently, a pushbutton reset or power cycle is required. This problem does not occur while using copper.~~
9. ~~Disable output links if fiber loses lock via Serdes sync error or loss of power. Wait until certain number of comma characters have been seen before re-establishing link outputs to downstream TDUs and DCMs.~~
10. ~~Disable the Error Mask register, and instead create an Error Disable Mask register at a new address. This will allow all errors to be seen unless the user disables a particular error.~~
11. ~~Fix SERDES_ERR LED~~
12. ~~Manually modify the DCM delay fifos.~~