CERN Trajectory Measurement System

PUPE API - Preliminary

Project	CERN-TMS
Date	2006-10-24
Reference	Cern-tms/pupe-api
Version	0.6 Preliminary
Author	Dr Terry Barnaby

Table of Contents

1. References

- IT-3384/AB: Technical Specification for a new trajectory measurement system for the CERN Proton Synchrotron.
- Alpha Data's TMS Tender "pre-design-1.5".
- Emailed questions answered by Jeroen Belleman of CERN.
- Visit to CERN on 2006-06-20

2. Introduction

This PUPE API document concerns the API between the PUPE FPGA firmware and the software running on the Module controllers.

3. Overview

The TMS operates in cycles normally lasting 1.2 seconds but can be up to 2 seconds. Machine cycles can

occur consecutively with 30ms between each although this gap can be over 1 second. The start of a cycle is initiated by the **CYCLE_START** timing signal and is terminated by the **CYCLE_STOP** timing signal. A cycle number counter is incremented on the end of a cycle. The cycle number counter can be set in software prior to the **CYCLE_START** event. A C-timer counter is zeroed on the start of cycle and is incremented every 1ms for timing purposes. The 10MHz System clock is used as a reference for this and also for synchronising the incoming timing signals.

During a Cycle, a set of particle bunches will be orbiting within the machine at a rate of about 477KHz (could be as low as 177KHz). There could be from 0 to 25 particle bunches orbiting at any one time. The integrals of the Sigma, DeltaX and DeltaY values for each bunch are stored in memory per orbit (turn). The memory is arranged as a simple two dimensional array, the first dimension being the bunch number within a bunch set and the second being the orbit number.

A Cycle Timing Table has an entry every millisecond and stores the address in memory for the first bunch in the particle bunch set that occurred just after that time. The accuracy of bunch set timing is thus about 2us assuming an orbit rate of 477KHz.

A synchronisation system provides timing signals for the capture, DSP and memory storage system. This system synchronises itself to the bunches orbiting in the machine. As there are no bunches orbiting to start with, an external timing signal, FREF provides a reference to synchronise to prior to having a particle beam to lock on to. The FREF signal will be at the orbit rate. It will need to be effectively delayed to match the position of the particular Pick Up in the ring. A register PLL_PHASE_DELAY defines this delay. Once the particle beam is present then the beam's Sigma signal is used for accurate synchronisation, this will be at h x the orbit rate where h is the current harmonic number of the machine. The harmonic number of the machine defines how many RF Buckets there are and hence how many data sample sets are written to memory every orbit. The PLL system will produce a h x FREF signal synchronised to the incoming FREF signal or the Sigma signal. A well as the h x FREF signal the PLL will produce a Gate, BLR and other timing signals. The FREF signal will continue to provide timing for the orbit of the particle bunch set defining which of the RF Buckets is the first of the bunch set.

The PLL has a Phase Table that can be loaded on demand at any time but synchronised to FREF ie the start of a particle bunch (should this be the end to allow time for the change ?). The PLL's initial frequency can also be configured.

During a cycle a number of events can occur which are signalled by timing signals. The injection signal indicates the point of injection of a BEAM into the machine. The calibration start and stop signals indicates a calibration period. The H-CHANGE signal indicates a harmonic change, ie the number of bunches in a bunch set changes. The timing and memory address at which these events happen are stored in a table for information. The PLL tables will need changing on H-CHANGE and other events.

Each PUPE has test and diagnostics functions. This enables the timing inputs to be driven manually. A set

of data from various parts of the system can be sampled and stored into a memory block for diagnostics.

Data from previous cycles can be accessed from the memory while the system is processing the next cycle of data. In order to access a particular pattern of data from the results there may be a Data Pattern Engine to retrieve particular data patterns from the result data using DMA.

Each PUPE implements 3 Pick-Up processing channels. Each of these runs predominantly independent from each other.

4. Typical Data Required

During a process cycle of 1 second the system will acquire about 40MBytes of data per Pick-Up channel. Not all of this data is required. Generally some pattern of data is required from the system dependent of the measurements being made. The typical measurements required include:

- Trajectory measurements of a single selected bunch through all Pick-Up's over a small number (up to a few tens) of orbits. This would be best accomplished by having an FPGA on-board Data Pattern Engine to DMA the values for one of the bunches.
- The instantaneous position of a selected bunch through two selected pick-ups over a large number of orbits (~100k), hereafter referred to as a `phase space plot', or PSP. This would be best accomplished by having an FPGA on-board Data Pattern Engine to DMA the values for one of the bunches.
- The mean for each Pick-Up over a few hundred orbits. For a small number of selected pick-ups, this measurement may be requested at 1ms intervals in order to form a plot of the evolution of the mean position in these pick-ups over an acceleration cycle. This may be best achieved by calculating the mean on the fly and storing this in the Cycle Timing table. Alternatively an on-board Data Pattern Engine could calculate the mean values and DMA the results.
- The (weighted) mean of a selected bunch over all pick-ups over a few hundred orbits, hereafter referred to as the `Mean Radial Position', or MRP. This measurement may be requested at 1ms intervals to form a plot of the evolution of the MRP over an acceleration cycle. This may be best achieved by calculating the mean of a specified bunch on the fly and storing this in the Cycle Timing table. Alternatively an on-board Data Pattern Engine could calculate the mean values and DMA the results.

5. Control and Status Registers

The control and status registers are mapped into memory. Each register is 32bits wide and is situated on a 64bit address boundary.

Address	*Name*	*Description*
0x000000	IMEM_REG	Internal block ram register (bank index)
0x000008	LOCKED	DCM lock status
0x000010	ADDR_REG	Memory page address register
0x000018	MEM_REG	Memory access register (bank, internal/external)
0x000020	IER	Interrupt enable register
0x000028	ISR	Interrupt status register
0x000800	FIRMWARE	Firmware magic and version. ‘C’ + ‘N’ + VersionMajor + VersionMinor.
0x000808	CONTROL_0	Reset, Start, Stop, LoadPhaseTable, LoadSwitchTable,TimingMaster,TimingTerminate,Running,Error (RW)
0x000810	CYCLE_0	The current cycle number. On write will be used as the cycle number for the next cycle on CYCLE_START. (RW)
0x000818	TIME_0	The current cycle time in ms (RO) (could be in us)
0x000820	NUM_BUCKETS	The number of RF buckets to be processed. Will be updated to the value written here on CYCLE_START and HChange. (Maybe this should be a list of NUM_BUCKT values ??) (RW)
0x000828	PLL_FREQUENCY	The initial value of the orbit frequency in Hz
0x000830	PLL_FREQUENCY_DELAY	The delay in ms after CYCLE_START at which the PLL frequency register is loaded with ORBIT_FREQUENCY.
0x000838	PLL_PHASE_DELAY	PLL Phase delay setting. This is based on the position in the ring the PU is located and is used to effectively phase offset the FREF source. (what values ?) (RW)
0x000840	PLL_GAIN	Sets the gain of the PLL (what values ?) (RW)
0x000848	DPE_CONTROL	Data pattern engine control ???

Address	*Name*	*Description*
0x000850	TEST_0	Sets test parameters (RW)
0x000858	TEST_1	Sets diagnostics output (undefined at the moment)(RW)
0x000860-	Pick Up Channel 1 registers. Same as for Pick Up channel 0	
0x0008C0-	Pick Up Channel 2 registers. Same as for Pick Up channel 0	
0x200000:0x3FFFFF	MEMORY	Memory window. 64 bits wide, suitable for slave mode DMA access

The system memory is organised as follows:

Bank	*MEM_REG*	*IMEG_REG*	*Address*	*Description*
DDR 0	0x08	0xXX	0x00000000	Channel 0 data
DDR 1	0x09	0xXX	0x00000000	Channel 1 data
DDR 2	0x0A	0xXX	0x00000000	Channel 2 data
DDR 3	0x0B	0xXX	0x00000000	Test data
BlockRAM 0	0x00	0x00	0x200000	Cycle Information table channel 0
BlockRAM 0	0x00	0x00	0x200200	Timing Phase Table channel 0. This is loaded into the systems block ram on the LoadPhaseTable command.
BlockRAM 0	0x00	0x00	0x200400	Timing Switch Table channel 0. This is loaded into the systems block ram on the LoadSwitchTable command
BlockRAM 1	0x00	0x01	0x200000	Cycle Timing table channel 0 (8KBytes)
BlockRAM 2	0x00	0x02	0x200000	Pattern Read Buffer channel 0

Bank	*MEM_REG*	*IMEG_REG*	*Address*	*Description*
BlockRAM 4	0x00	0x04	0x200000	Cycle Information table channel 1
BlockRAM 4	0x00	0x04	0x200200	Timing Phase Table channel 1. This is loaded into the systems block ram on the LoadPhaseTable command.
BlockRAM 4	0x00	0x04	0x200400	Timing Switch Table channel 1. This is loaded into the systems block ram on the LoadSwitchTable command.
BlockRAM 5	0x00	0x05	0x200000	Cycle Timing table channel 1(8KBytes)
BlockRAM 6	0x00	0x06	0x200000	Pattern Read Buffer channel 1
BlockRAM 8	0x00	0x08	0x200000	Cycle Information table channel 2
BlockRAM 8	0x00	0x08	0x200200	Timing Phase Table channel 2. This is loaded into the systems block ram on the LoadPhaseTable command.
BlockRAM 8	0x00	0x08	0x200400	Timing Switch Table channel 2. This is loaded into the systems block ram on the LoadSwitchTable command.
BlockRAM 9	0x00	0x09	0x200000	Cycle Timing table channel 2 (8KBytes)
BlockRAM A	0x00	0x0A	0x200000	Pattern Read Buffer channel 2

The Register bit definitions are:

Name	*Bits*	*Description*
Control.Reset	0	Resets the Pick Up channel
Control.Start	1	Starts the next processing cycle. The actual start will happend on the next CYCLE_START timing signal.
Control.Stop	2	Manual stop cycle. Normally the CYCLE_STOP timing signal will terminate the cycle
Control.LoadPhaseTable	3	Loads the new phase table on next system clock edge. (Probably just swaps between two tables)

Name	*Bits*	*Description*
Control.LoadSwitchTable	4	Loads the new switch table on next system clock edge. (Probably just swaps between two tables)
Control.TimingMaster	5	Sets the logic to connect the digital timing signals to the timing bus. Used on the card connected to the digital timing signal input panel.
Control.TimingTerminate	6	Sets the logic to terminate the timing bus. Used in the last card on the timing bus.
Control.Running	16	System is running (RO)
Control.Error	17	System is in Error (RO)
Test.TestOutput	0-3	Selects the signal which will be output on the channels digital output connector. 0 – none, 1 – generated FREF.
Test.DataSource	4	Selects input data source from DDR3 bank rather than from ADC's for testing
Test.ClockSource	5	Selects internal 10MHz clock rather than digital input system clock
Test.FRefSource	6	Selects internal Fref source (477 Khz ?)
Test.CycleStart	16	Timing bus signal
Test.CycleStop	17	Timing bus signal
Test.CalStart	18	Timing bus signal
Test.CalStop	19	Timing bus signal
Test.Injection	20	Timing bus signal
Test.HChange	21	Timing bus signal
Test.TimingDrive	23-31	Sets the board to drive the systems timing bus with the values from this register. This is a bit mask allowing each timing signal to be driven independently.

The table sizes are as follows:

Table	*Size*	*Time*
Cycle Data table	256 MBytes actual used size rounded down to be a multiple of 6 bytes the data entry size	4.46 seconds 21 bunches
Cycle timing table	17840 Bytes. Each entry is 8 bytes. Does this need to be bigger to allow the system to be set to capture single bunches with more past data held in the Cycle Data table ?	4.46 seconds 21 bunches
Cycle Information table	512 Bytes for 4 Cycles of information	
Timing Phase Table	512 Bytes	
Timing Switch Table	8 Bytes for an 8 x 8 bit matrix	

6. Synchronisation and Timing

One of the major aspects of the PUPE is its synchronisation system. The synchronisation system produces internal timing signals for the PUPE based on external timing signals and the analogue Sigma data source. Each Pick-Up Channel has the following input timing signals:

Name	*Description*
SYSTEM_CLOCK	Master 10MHz system clock. The ADC's 125 Mhz sampling clock will be synchronised to this clock and all of the digital timing signals, except the Injection signal, will re-synchronised to this clock within each FPGA
FREF	Reference frequency. (Up to 477KHz)
CYCLE_START	Start of a machine cycle
CYCLE_STOP	End of Last Flat Top, effectively end of cycle
CAL_START	Start of calibration period
CAL_STOP	End of calibration period
INJECTION	Injection
H_CHANGE	Harmonic changes
SPARE	Spare input not used
SIGMA	The amplitude of the signal from the pick up. This is an analogue signal

There are two tables to control the synchronisation process. A phase table and a switch table. The phase table consists of an 512 element array of 8 bit values. The switch table is an 8 x 8 bit matrix. These tables can be loaded by software at any time before, after or during a processing cycle. The software loads the table via the shared memory interface and then sets an appropriate bit in the CONTROL register to load the table. The following shows a typical set-up of the switch table, the actual parameters and positions are yet to be defined:

	CYLE_START 0	*CAL_START 1*	*CAL_STOP 2*	*INJECTION 3*	*H_CHANGE 4*	*CYCLE_STOP 5*	*SPARE 6*
Clear All	*		*			*	
Clear C-timer	*						
Start Acquisition		*		*			
Stop Acquisition			*			*	
LO Toggle			*	*	*		
Gate/BLR Toggle			*		*		
RF Toggle			*	*			
Next Phase Table					*		

7. Interrupts

Interrupts can be generated on each of the following events:

Event	*Interrupt Bit*	*Description*
CycleStart	0	A processing cycle has started
CycleEnd	1	A processing cycle has completed

8. Data Access

Access to the data from the PUPE is provided through a shared memory interface. To reduce the shared memory access window a simple 2MByte paged system is used to access the FPGA's block RAM and on-board DDR. The on-board DDR is organised as four banks of 256 Mbytes. Within this memory is stored the main data tables for each of the Pick-Up channels.

Each of the Pick-Up channels has its own data table stored within its own bank of 256MBytes of DDR.

The format of this data table is a circular buffer of Sigma, DeltaX, DeltaY and Spare 16bit signed values in little endian order. The Spare entry is there to give 64bit data alignment for efficiency and simplicity of access, it may be used to contain other data in the future (microsecond timer ?). For each particle bunches orbit there will be h sets of data stored where h is the machines harmonic number. The machines harmonic number can change on the H-CHANGE timing event and hence the number of sets of data stored per orbit will change. The points of change are recorded in the Cycle information table. The data table can contain many cycles of many orbits of data.

0	*2*	*4*	*6*	*8*	*10*	*12*
Sigma	DeltaX	DeltaY	Spare	Sigma	DeltaX	...

In the Block RAM of the FPGA memory there is a Cycle Timing table. This circular buffer consists of one timing entry per millisecond. Each entry in the timing table contains a 32bit unsigned “Cycle Number” field and a 32bit unsigned memory address. An entry is made in the timing table every millisecond. The memory address points to offset in the data table which contains data for that time. This will be the address of the first bunch in the next orbit after this time.

0	*4*	*8*	*16*	*20*
Cycle Number	Data Address	Cycle Number	Data Address	....

There is a Cycle Information table, contained within internal BLOCKRAM memory which contains timing positions for particular events. There will be information for up to 4 machine cycles. An example Cycle information table follows.

Address	*Type*	*Millisecond Time*	*Orbit Number*
0	CycleStart, Nbuckets, CycleNumber	0	0
4	CalStart	100	0
8	CalStop	150	10
12	Injection	200	2
16	HChange, NBuckets	300	100
20	HChange, NBuckets	400	250
24	HChange, NBuckets	450	0
28	CycleStop	800	33

9. Software Usage

There follows a typical TMS Cycle measurement scenario from the controlling software's perspective.

- Software configures the Pick Up engines main registers including the CONTROL, PLL_FREQUENCY_DELAY, PLL_PHASE_DELAY, PLL_GAIN and TEST registers.
- The Software starts of the next processing cycle. It loads the set of Phase Tables and the switch table for the cycle. It sets the PLL_FREQUENCY and CYCLE number register and initiates a processing CYCLE by setting the START bit in the Control register.
- The PUPE Firmware loads the initial Phase table and Switch table on the CYCLE_START timing signal and starts processing the incoming data. On timing input events the PUPE Firmware carries out appropriate actions based upon the Switch table. For example on H-CHANGE events the system will change to use the next Phase table.
- The PUPE Firmware will record each event into the Cycle information table together with the millisecond time and orbit number when it occurred.
- On the CYCLE_STOP timing input the processing will stop and the software will be informed by interrupt. The software can now access the resulting data possibly through a Data Pattern Engine implemented in the FPGA or by direct memory accesses.
- The next processing cycle can be started immediately after the previous cycle even while the previous cycles data is being collected.

10. Notes

- The C-Timer could be implemented as a microsecond counter. This would give the ability for more accurate timing for future developments.
- Could store microsecond time in the data table. This would increase memory requirements but would give more accurate timing information.
- We might want the ability for the FPGA to DMA a selected pattern of data to the host. For example select the data for bunch 1 and copy that data to the host. This would be to improve the performance of the data retrieval.
- One thing we will have to consider is how fast we need to get at the data in memory. CERN will want to be able to take out a selection of data before the next machine cycle. With the paging registers, the table indirection (Cycle Timing to Data) and the fact that they may only want 6bytes here and there over a PCI bus this may be slow. We thus may need to add a data gathering facility in the FPGA itself such as a function that would DMA every 21st set of 6byte samples stating from a given address to the host.

- Another option on the data, as we have more memory that the original spec, is to store data timing information with the data itself rather than have a separate table. We could packetise the data with a simple header containing the channel number and time followed by 21 sets of Sigma/DeltaX/DeltaY data. This would make finding a particular time a bit harder/slower but we could store more accurate times on the data.
- It might be best to calculate and store some of the data that requests for data would require on the fly. For example it would be relatively easy to calculate the mean of DeltaX for bunch 1 over a 1ms period on the fly.
- The Cycle Timing table might want to be bigger to allow the system to be set to capture single bunch data over a longer time period.