VME-DDS Frequency and Phase Control Module, Part II

Memory and Data Processing

June 6, 2011
Revised February 17, 2017

Craig Drennan

I. Introduction

This document is written to provide details on the layout and function of the memory and data processing on the VME-DDS Module so that there is an understanding of

1. How memory is accessed and how the different data processing functions and the VME interface share the memory and the address data busses.
2. How the Flash memory is used to store and restore (back-up and boot-up) memory data and registers used by the module.
3. What is the standard sequence in which the various processes are executed and how the processes are relegated to an Update Interval, when the Booster is not actively accelerating beam and a Run Interval when it is.

With this information we hope to understand how future parameters and processes can be added and what rules those processes would need to adhere to.

II. Managing Process Parameters

The parameters used by the module are typically scale factors, offsets, time delays, and process flow control switch settings. Three memory words in shared memory are associated with each parameter. There is a "Setting", a "Reading" and an "Update Flag". For each parameter there are defined in Read-Only Memory (ROM) a maximum, a minimum and a default. And finally, there is a dispatch table in the ROM memory of address pointers to the Run-Mode parameter registers. It is the value in these register that is actually accessed by the Run-Mode processes.

The Shared Memory Parameter Update Process was described in the document VME-DDS Frequency and Phase Control Module, Part I. The details to follow, will aid in the addition or subtraction of the operator programmable parameters.

1. The maximum, minimum and default parameter values, and the address pointers to the Run-Mode registers are located in ROM. This ROM memory is actually RAM memory in the FPGA that cannot be written and is initialized by the contents of a memory initialization file (*.mif) when the FPGA code is compiled. The particular file used is “SM_0x000000_0x002000_init.mif” in the Upper FPGA project folder.
2. The “mif” file should be opened with a simple text editor like Microsoft’s NotePad. If the file is opened in Altera Quartus the comments cannot be read and the comments would be eliminated if saved from Quartus.
3. Apart from the comments, which are indicated by double dashes “- -“, the “mif” file has two columns delimited using a colon “:”. Both columns are given as Hexadecimal values. The first column is basically an index, but a physical memory address can be determined. The value at each index is 4 bytes, so the physical address is the index times 4 plus the memory offset at which the memory block is located. This ROM memory is part of the Shared Memory Block which begins at address 0x000000.

4. The Integer Parameter Dispatch Table begins at index 0x040, or physical address 0x100. A sample of this section, at the time this document was written, is below.

```
---------------------------------
-- INTEGER PARAMETER DISPATCH TABLE
---------------------------------
     040  :   0003F800; -- Address of the Injection Frequency Variable
     041  :   0003F804; -- Address of the Number of Injections Points Variable
     042  :   0001F000; -- Address of the Curve Start Delay from Trigger Variable
     043  :   0001F004; -- Address of the Reverse Step Size Variable
     044  :   0001F008; -- Address of the Phase Error Gain Exponent, y (gain =2^y)
     045  :   0001F00C; -- Address of the Extraction Phaselock Frequency
     046  :   0001F010; -- Address of the Phase Drive Proportional Gain
     047  :   0001F014; -- Address of the Phase Drive Integral Gain
     048  :   0001F018; -- Address of the Transition Time Interval, microseconds
     049  :   BAD0F349;
     04A  :   BAD0F34A;
            o
            o
            o
```

5. The parameter maximums, minimums and defaults begin at index 0x280, or physical address 0xA00. A sample of this section is below.

```
-----------------------------------------------------------------------
-- BEGIN SM PARAMETER LIMITS
-----------------------------------------------------------------------
-- Frequency parameters are computed assuming a 480 MHz system clock
-- for the DDS components. FTW = 2^32 * Fout / Fsysclk = 8.947849 * Fout
-----------------------------------------------------------------------
     280  :   15555590;  -- Maximum Injection Frequency = 40 MHz
     281  :   1000002C;  -- Minimum Injection Frequency = 30 MHz
     282  :   14369D3B;  -- Default Injection Frequency = 37.9 MHz
-----------------------------------------------------------------------
-- Time delays are based on counting the FPGA logic clocks that have
-- a period of 20 nano-seconds.
-----------------------------------------------------------------------
     283  :   000003E8;  -- Maximum Number of Injection Points = 1000
     284  :   0000000A;  -- Minimum Number of Injection Points = 10
     285  :   000001F4;  -- Default Number of Injection Points = 500
-----------------------------------------------------------------------
     286  :   000249F0;  -- Maximum Curve Delay = 3   milli-seconds
     287  :   000000E1;  -- Minimum Curve Delay = 4.5 micro-seconds
     288  :   000186A0;  -- Default Curve Delay = 2   milli-seconds
-----------------------------------------------------------------------
     289  :   00002710;  -- Maximum Reverse Step = 10000
     28A  :   00000001;  -- Minimum Reverse Step = 1
     28B  :   00001BF6;  -- Default Reverse Step = 7158
-----------------------------------------------------------------------
-- The phase error gain term is actually the exponent y such that the
-- gain = 2^y.
     28C  :   00000010;  -- Maximum Phase Error Gain y = 16
```

```
    28D  :   00000001;  -- Minimum Phase Error Gain y = 1
    28E  :   00000008;  -- Default Phase Error Gain y = 8
-------------------------------------------------------------------------
    28F  :   10000000;  -- Maximum Extration Phaselock Freq = 60 MHz
    290  :   0CCCCCCC;  -- Minimum Extration Phaselock Freq = 48 MHz
    291  :   0E14B64C;  -- Default Extration Phaselock Freq = 52.8034 MHz
-------------------------------------------------------------------------
    292  :   00000080;  -- Maximum RPOS Phase Drive Gain = 128
    293  :   00000001;  -- Minimum RPOS Phase Drive Gain = 1
    294  :   00000001;  -- Default RPOS Phase Drive Gain = 1
-------------------------------------------------------------------------
    295  :   00000080;  -- Maximum RPOS Phase Drive Integral Gain = 128
    296  :   00000001;  -- Minimum RPOS Phase Drive Integral Gain = 1
    297  :   00000001;  -- Default RPOS Phase Drive Integral Gain = 1
-------------------------------------------------------------------------
    298  :   00000019;  -- Maximum Transition Time Interval = 25 us
    299  :   0000000A;  -- Minimum Transition Time Interval = 10 us
    29A  :   00000014;  -- Default Transition Time Interval = 20 us
-------------------------------------------------------------------------
    29B  :   FFFFFFFF;
    29C  :   00000000;
    29D  :   DEFA4170;

          o
          o
          o
```

6. The “mif” file will also initialize the dual-port “Shared Memory” RAM locations that are the Flag, Setting and Reading of the parameters. If the DDS module starts up normally these initial values will be over written with the Boot-up values from the Flash memory. For reference the Flag, Setting and Reading initialization values start at index 0x5C0, address 0x1700.
7. Besides the memory initialization file, the FPGA code uses memory map constants defined in the file “dds_vme_constants.vhd”.
8. In order to add or remove a parameter in the DDS application the memory initialization file, “SM_0x000000_0x002000_init.mif” needs to be edited to include the new address of the Run-Mode register in the Parameter Dispatch Table and the maximum, minimum and default values for the new parameter. The relative order of the register addresses and the set of maximums, minimums and defaults, and the Flag, Setting, and Reading set needs to be maintained. For instance, the third address in the dispatch table will be associated with the third set of maximum, minimum, default and the third set of Flag, Setting, and Reading.
9. In the file “dds_vme_constants.vhd”, the constant “SM_PARAM_BLOCK_LAST” needs to be set to the memory address of the last parameters Reading. This is the variable used by the parameter update process to determine when this process has been completed.

III. Sequence and Timing

There are four main intervals in which different application processes run in the DDS module. The first is power on, boot-up. This involves the timing of resets, the sequencing of power to the analog sections of the module and the initialization of memory and registers from non-volatile, Flash memory. The second interval is the Active interval which begins with the Booster reset event (TCLK trigger) and extends 36 ms. During this interval the Booster acceleration controls, and frequency and phase curves are active. The third interval is the Update interval in which changes made by ACNET through the front-end processor are managed. This interval follows each Active interval and extends 24 ms. And the final interval is the Initialization interval, which follows the Update interval and precedes the next Active interval, in which all of the initial values for the frequency and phase curves are set.

III.1 Power On, Boot-Up Interval

The sequence for the power on reset is shown in Figure III.1.1. The majority of the logic is held in reset by the signal Reset_n (active low). Before this reset is released the PLL's in both the Upper and Lower FPGA's are reset and the power supplies for the analog to digital converters are sequenced on according to the AD7625 requirements.

The ADC's have three different power sources. There is the 2.5V which comes up with the +5V VME crate power. An isolated 2.5V rail comes up next followed by the isolated +5V rail. Finally, once the ADC's are powered the module analog inputs are switched on, that is the input protection relay contacts are closed.

The Upper FPGA transmits a copy of the Reset_n signal to the Lower FPGA. This copy is released 2 logic clocks before releasing its own version in order for synchronization of the reset to occur in the Lower FPGA. The Upper and Lower FPGA will come out of reset at the same time, synchronous to the logic clock.

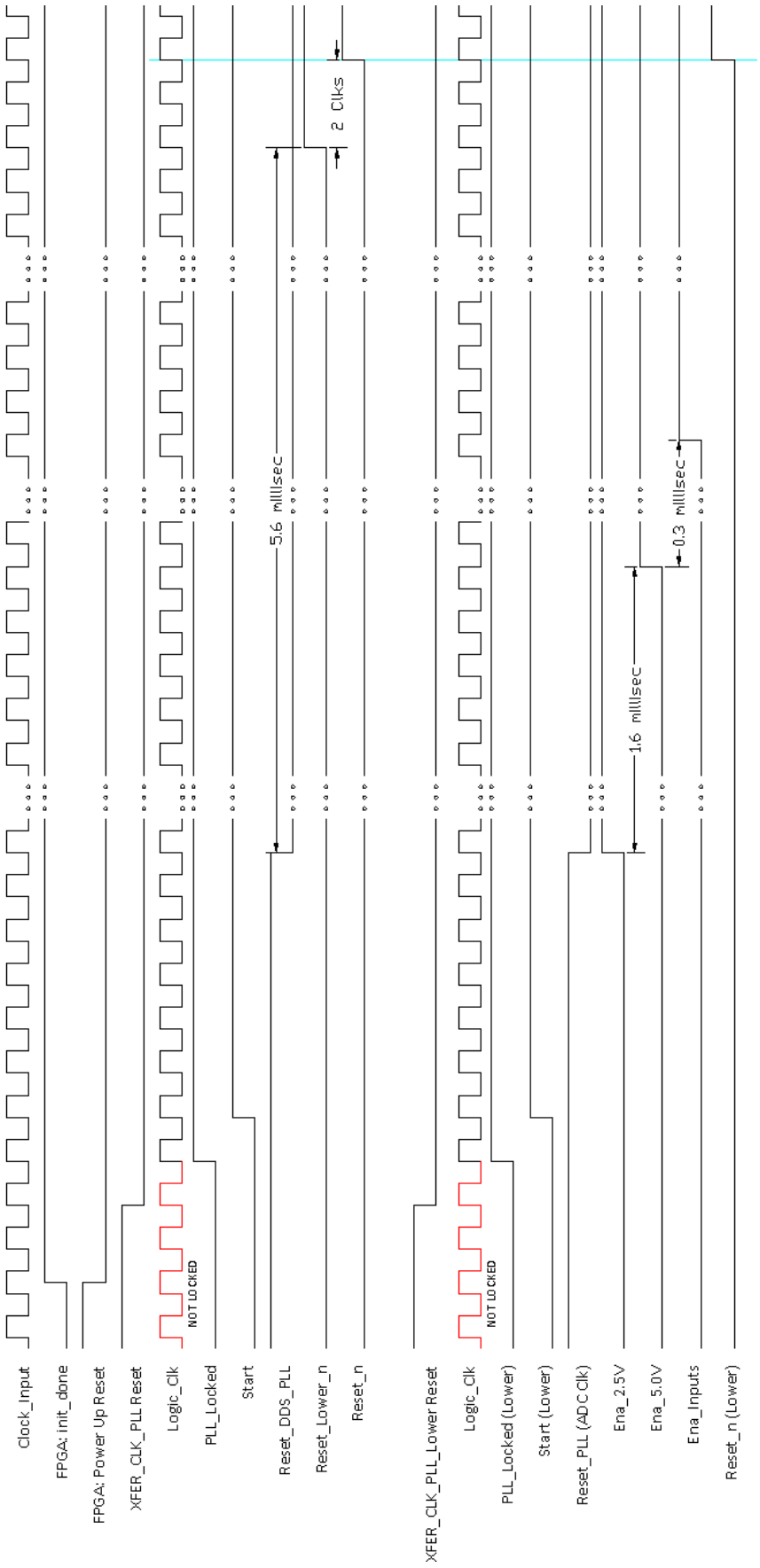

Figure III.1.1 DDS logic reset timing.

Upon coming out of reset, the process *dds_startup_sequence* runs to initialize the shared memory, parameter registers and curve memory with data stored in the FLASH memory. This startup process does pause for 1.3 ms after coming out of reset to ensure the other processes have initialized and stabilized.

The following sequence is executed by the *dds_startup_sequence* process.

1. Trigger the Bootup process which transfers the active FLASH memory page to the Shared Memory RAM.
2. Set the *Verify_All* signal and trigger the *Update_SM_Process*. This process updates the parameter registers with the contents of the parameter Settings word in Shared Memory. Limits on the parameters are checked and default values are set if any of the Settings words are out of range.
3. The *CTBFREE* semaphore used for accessing the curve buffer interface between the DDS-VME module and the front-end processor is initialized (high).
4. The status bits for each curve buffer is set to indicate that the curve needs to be interpolated and or loaded from Shared Memory to the curves memory, and then the Curve Interpolation Process is triggered eight times to process the eight curves implemented in the DDS-VME module.
5. Finally, *DDS_Ready* is signaled indicating that the module is ready to run upon receiving the next 15 Hz Booster trigger.
6. *DDS_Ready* going active will also result in signaling *Reset_Run_Curves* which initializes the DDS Frequency and Phase curves.

III.2 The Active, Update and Initialize Intervals

The Active, Run Interval begins with the rising edge of the 15 Hz Booster trigger input. This input is not synchronous to the Logic Clock and is synchronized within 2 clocks to produce a single clock wide *start_cycle* pulse. The *start_cycle* pulse triggers the *Process_Timer* state machine out of idle into a state that waits for the delay specified by the operator controlled parameter "Curve Delay". After the initial programmed delay, the *Process_Timer* will set the *Run_Curves* signal to begin the computation and updating of the DDS Frequency and Phase words. At this time the Process Timer also starts generating a 1 us update pulse which is used to time the updating of the DDS's.

The Active, Run Interval ends 36 ms after the occurrence of the *start_cycle* pulse. With this, the Update Interval begins by setting the *Start_Update_Period* signal. *Start_Update_Period* triggers the *dds_update_sequence* which executes the following sequence.

1. Trigger the *SM_Update_Process.* This process will poll each of the parameter "Flags" to see if new parameter values have been set by the front-end processor. For each Flag found set, the new parameter Setting is tested against the parameter's limits, the parameter register is updated and the parameters Reading word is updated with the new value of the parameter register. If any of the parameters are updated, the *Parameter_Changed* signal is pulsed signaling a backup to FLASH memory.
2. Trigger the *Curve_Req_Process*. This process will poll each of the new curve and curve read request semaphores to see if the front-end processor wishes to Write or Read one of the curve buffers. If so, the specified curve is Written or Read, to or from the Shared Memory interface curve buffer. If a curve

is updated from the front-end processor the *Curve_Changed* signal is pulsed signaling a backup to FLASH memory, and a status bit for the curve is set indicating that the curve buffer needs to be interpolated and or Written to the curve's memory.

3. Trigger the *Curve_Interpolation* process. This process polls the curve status bits to find the "first" curve that needs to be interpolated and or written from the curve's buffer to the curve's memory. Only one curve will be interpolated/updated per Update Interval.
4. The final task of the *dds_update_sequence* is to service the backups to FLASH memory. Until the end of the Update Interval, the backup process will be granted access to the Shared Memory via the main databus.

The Bootup and Backup processes are described in Part I of the VME-DDS Module Design Manual. What we would like to expound on here is that the FLASH Write (or programming) process can take a full couple of seconds to accomplish. Below are some approximate values for how long things take.

FLASH block unlocking and initiation of the erase cycle	1.8 us
FLASH block erase cycle	750 ms
Writing the 32 word (16 bit) programming buffer	7.6 us
Programming the FLASH block from the 32 word buffer	432 us
Writing Shared Memory in 1,280 programming buffer cycles	470 ms

The management of the FLASH backup process is done with cooperation between the *dds_update_sequence* and the *Flash_Interface* process. The *dds_update_sequence* executes the short duration tasks, the FLASH block unlocking and the initiation of the erase cycle and the writing of the 32 word buffer. Access to the Shared Memory and control of the databus is needed for only short intervals. The *Flash_Interface* process triggers and monitors the longer processes, the block erase cycle and the programming of the FLASH block from the 32 word buffer cycle. The *Flash_Interface* process indicates when it is ready to have more data written to the 32 word buffer by setting the *Flash_Bus_Request* signal. This request is granted by the *dds_update_sequence* process setting the *Flash_Bus_Permit*, but only during the Update Interval, after all of the other update tasks listed above have been completed. The *Flash_Bus_Request* signal is cleared each time a 32 word buffer write is completed.

The Update Interval ends 60 ms after the occurrence of the *start_cycle* pulse. With this, the Initialize Interval begins by pulsing the *Reset_Run_Curves* signal. *Reset_Run_Curves* triggers the *Freq_Phase_Controller* process to reset its curve pointers and write the first Frequency and Phase words to the DDS's. Note that the Frequency and Bias curves were ramped from their final values back to their initial values beginning at the start of the Update Interval. Also the 4 us to 1 us curve interpolation processes are also loaded with the first and second curve points.

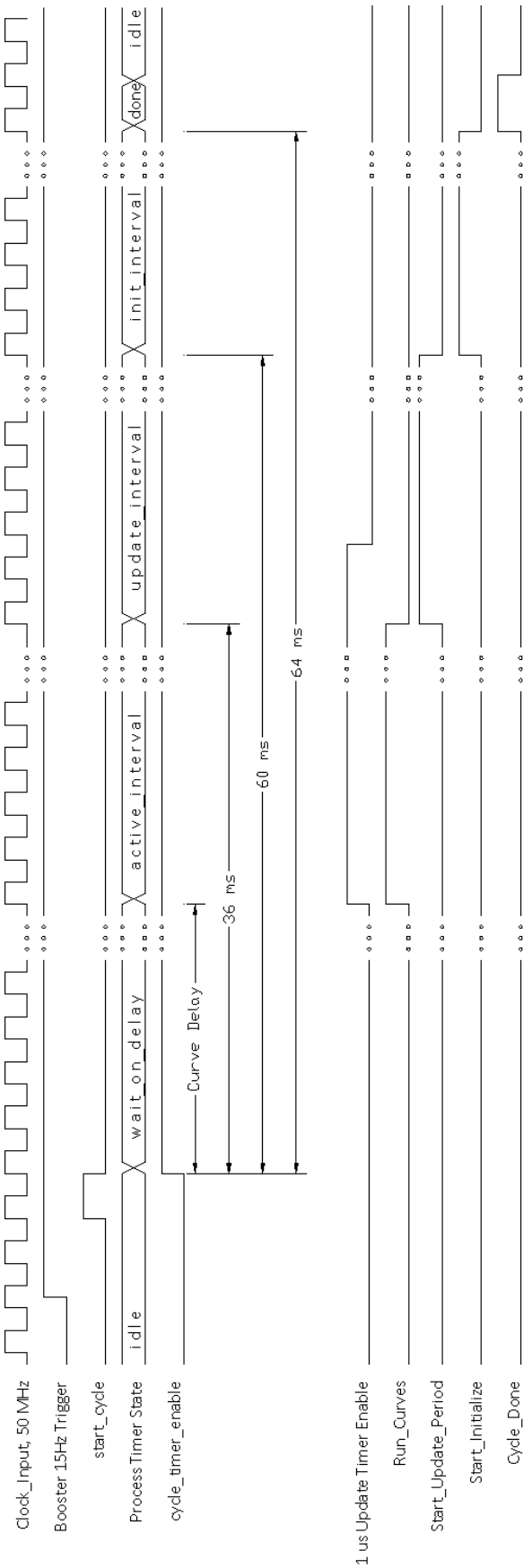

Figure III.2.1 Process Timer timing.

III.3 Run Interval Timing of DDS Frequency and Phase Updates

During the Run interval the frequency and phase of the DDS outputs are updated every microsecond. The value of the frequency and phase is computed as the sum of system feedback variables, such as radial beam position and beam to reference phase error, and feed forward frequency curve and phase curve values. Figure III.3.1 is a block diagram of the computation currently specified.

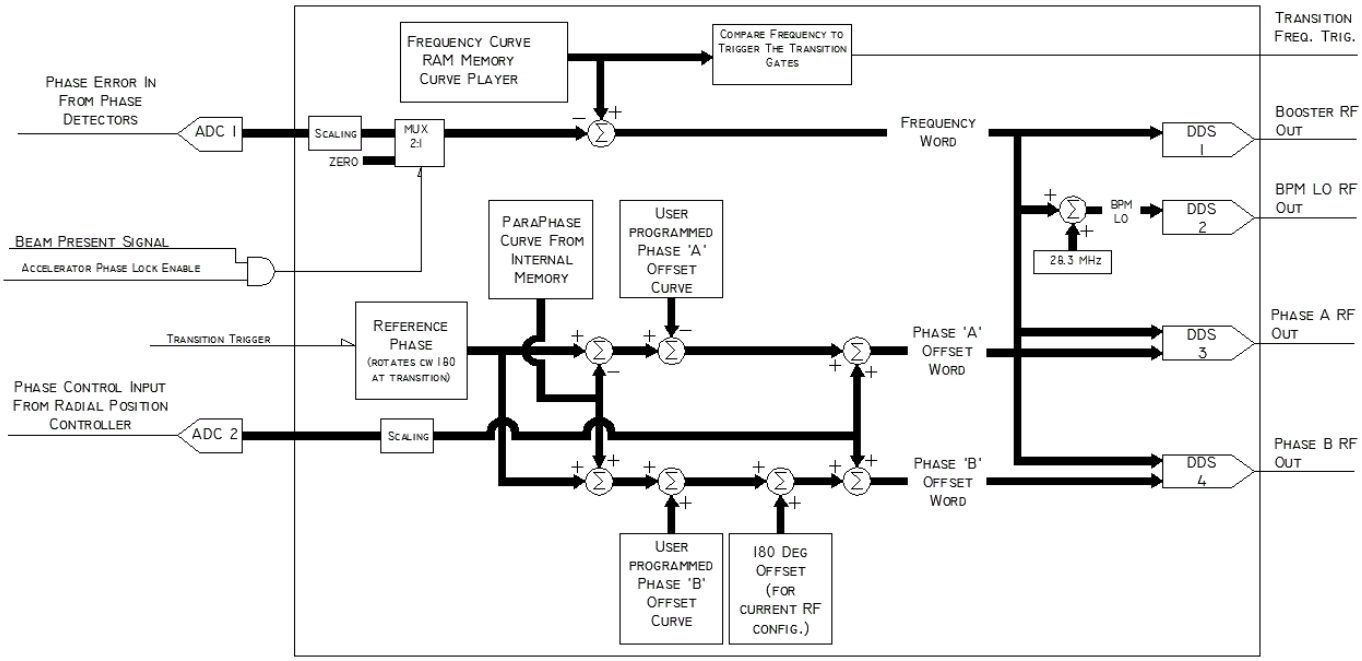

Figure III.3.1 Frequency and phase computation block diagram.

III.3.1 Curve Interpolation

The frequency and phase curves values are stored assuming a 4 us interval between points. Curve points for the 1 us updates are interpolated from these 4 us points as the curves are playing out. Figure III.3.2 is a timing diagram that illustrates the interpolation process. Two of the 4 us points are used in making the interpolation. The 4 us frequency or phase curve points are represented by X(k+1) and X(k+2), where k =0,1,2, …, 10240. The 1 us interpolated result is Y out. Upon power up of the module or within the Initialization Interval, the X(0) and X(1) values are preloaded into the computation and You is initialized to be X(0).

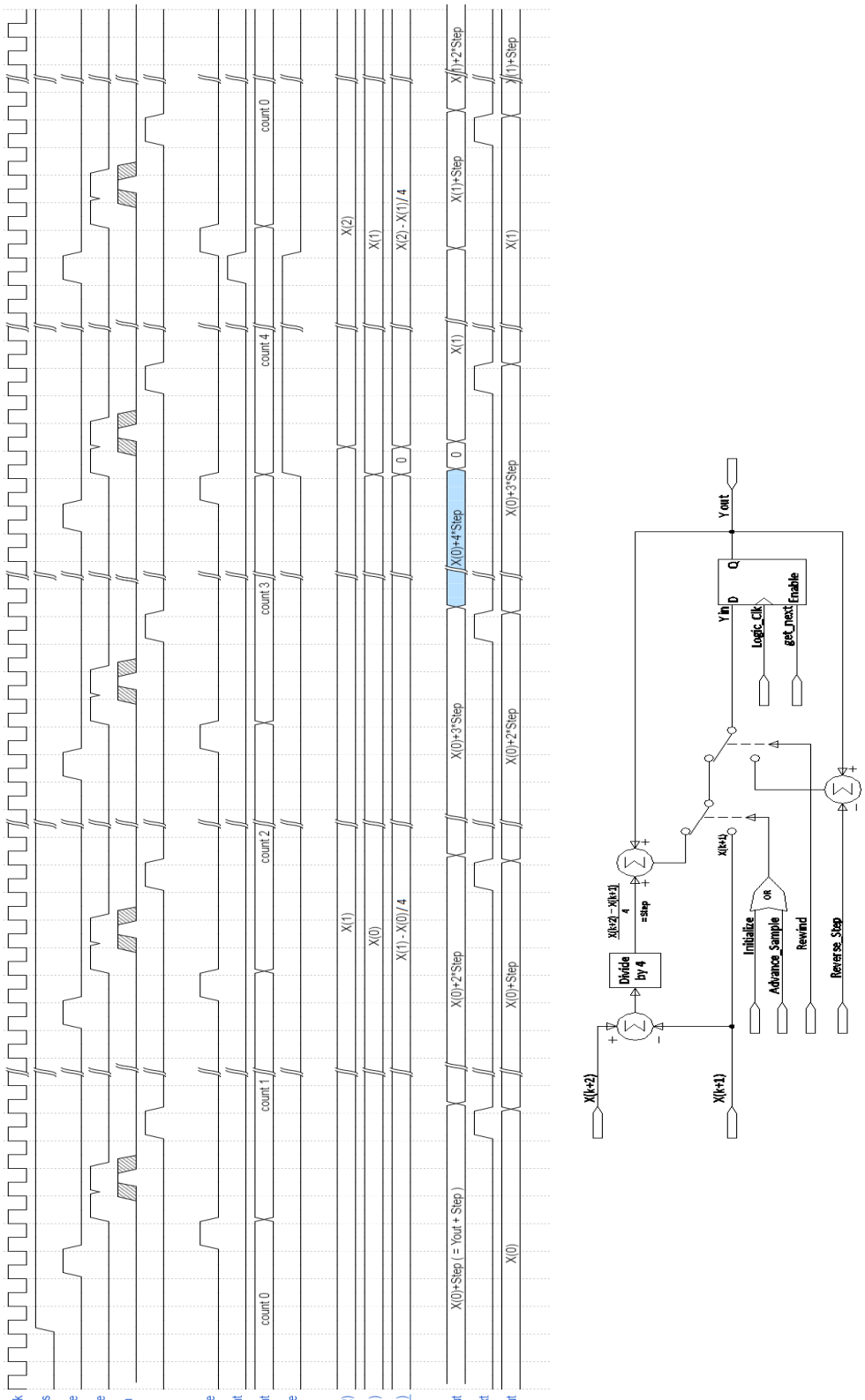

Figure III.3.2 The interpolation computation and timing

At the end of the Run Interval and the beginning of the Update Interval the frequency curve needs to be ramped, or rewound, back to its initial value. Step changes in the LLRF frequency reference will cause sparking in the Booster Accelerating Cavities, and hence the DDS frequency must be changed gradually as it is returned to the initial curve value. Since this rewind process occurs during the interval when other processes are using the data bus to update and backup memory, the rewind process has been implemented to avoid needing to use the data bus to access the curve. The approach is to use a constant slope ramp by applying a constant decrement, *Reverse_Step*, to the frequency value each microsecond. The value *Reverse_Step* is an operator programmable parameter.

III.3.2 Analog Input Synchronization

It is desired that the playing out of the frequency and phase curves and sampling of LLRF feedback variables through the Booster acceleration cycle be as consistent and repeatable as possible. With regard to the timing, we try to synchronize everything to the beginning of the Run Interval, which begins after the variable, operator programmed delay, after the occurrence of the 15 Hz trigger. The frequency and phase curves are played out in sync with the Update Pulse, which is synchronized to the beginning of the Run Interval. The ADC, analog to digital converters, which digitize the LLRF feedback variables sample at a constant 5 MHz and need to be re-synchronized each cycle. By re-synchronizing the ADC samples the timing between the arrival of the samples and the control computations is fixed. We can also minimize the processing latency contributed by the ADC sampling.

The analog to digital converter interface is explained in detail in Part I, the Module Design Manual. Figure III.3.2.1 illustrates the synchronization timing. The ADC interface outputs the convert pulse and the serial data clock to the ADC converters. The ADC converters echo the data clock back to the interface along with and in sync with the serial data signal. The basic timing of the ADC interface is to begin clocking data out of the ADC as soon as the MSB, most significant bit, of the data becomes available, and then to trigger another conversion as soon as the ADC has completely finished the previous acquisition. Data will still be clocking out of the ADC when the next conversion is signaled.

To synchronize the ADC conversions with the start of the Run Interval, the signal *Run_Curves* that goes high at the start of the Run Interval triggers the *Convert_Sync* signal to go high for 340 ns. The ADC interface will test the *Convert_Sync* signal just before it would normally signal the next ADC conversion. When *Convert_Sync* is found active, another ADC conversion is not signaled, but the clocking of ADC data out from the previous conversion is continued. Once this data transmission is completed the ADC interface waits for the end of the 340 ns sync interval, and then restarts the cycle of ADC conversions in sync with the end of the *Convert_Sync* signal.

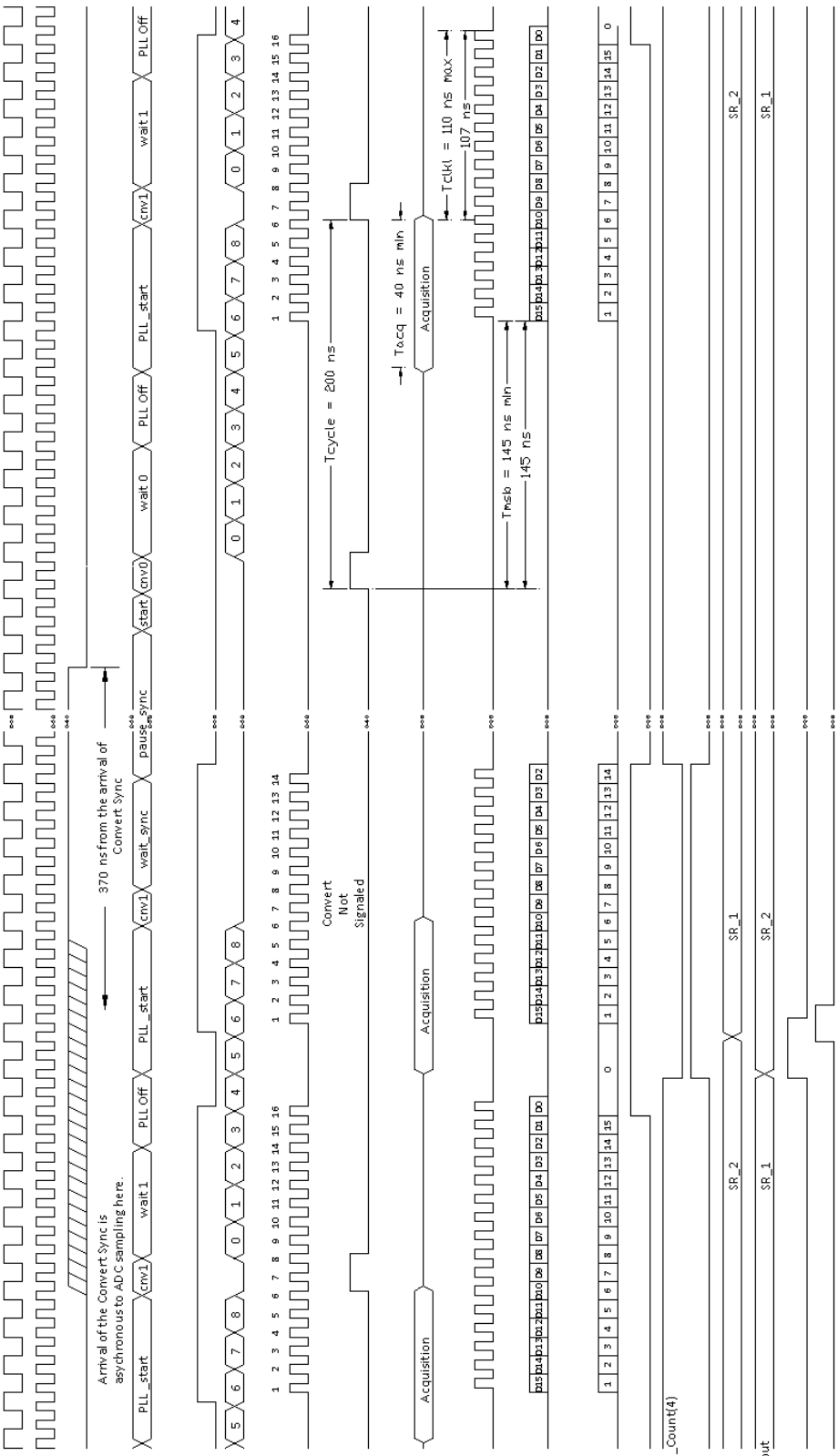

Figure III.3.2.1 ADC synchronization to the start of Run Interval.

III.3.3 Overall Frequency and Phase Update Computation Timing

From the computation block diagram in Figure III.3.1 we have that there are five values that are predetermined and can be setup before the 1 us Update pulse fires. These are the values from the frequency curve, the reference phase, the paraphase, and the A and B phase offsets. The LLRF feedback control variables, the Phase Error and the RPOS Phase Drive, need to be sampled and applied to the computation as close to the moment when the DDS output frequency and phase are updated as possible. The interval of time between when these analog inputs are sampled and their effect appears on the DDS outputs adds to the pure delay in the closed loops they are associated with.

Early design documents for the LLRF Booster controls estimated that the acceleration phase lock and the RPOS, radial position control, could tolerate a delay of 3 us. Due to the long cables that bring the beam phase and beam position from pickups in the accelerator, and the long cables that distribute the LLRF reference to the accelerating cavities, a delay of approximately 1.13 us is incurred. The estimate of the current delay in the LLRF control electronics is

Average analog input sampling delay (0.5 F sample)^-1	0.050 us
ADC converter pipeline delay	0.300 us
Processor instruction and computation delay	0.400 us
DDS pipeline delay	0.360 us
Cable delay	1.134 us
TOTAL =	2.244 us

Current estimates for the delay associated with the DDS-VME module are below. The new system increases the closed loop delay by approximately 0.434 us, but the total delay still remains below the 3 us specification.

ADC converter pipeline and data transfer	0.300 us
FPGA computation delay	0.100 us
DDS data update transfer delay	1.000 us
DDS pipeline delay (at 480 MHz system clock)	0.164 us
Cable delay	1.134 us
TOTAL =	2.698 us

By adjusting the ADC conversion synchronization at the start of the Run Interval, we can time the ADC interface output strobe to fire just before the Update pulse. The ADC value is immediately applied to the computation and the result is ready to write into the DDS interface within a couple clocks. Figure III.3.3.1 shows the timing between the ADC input, the 1 us update strobe, and the changing of the DDS frequency and phase words at the output of the computations.

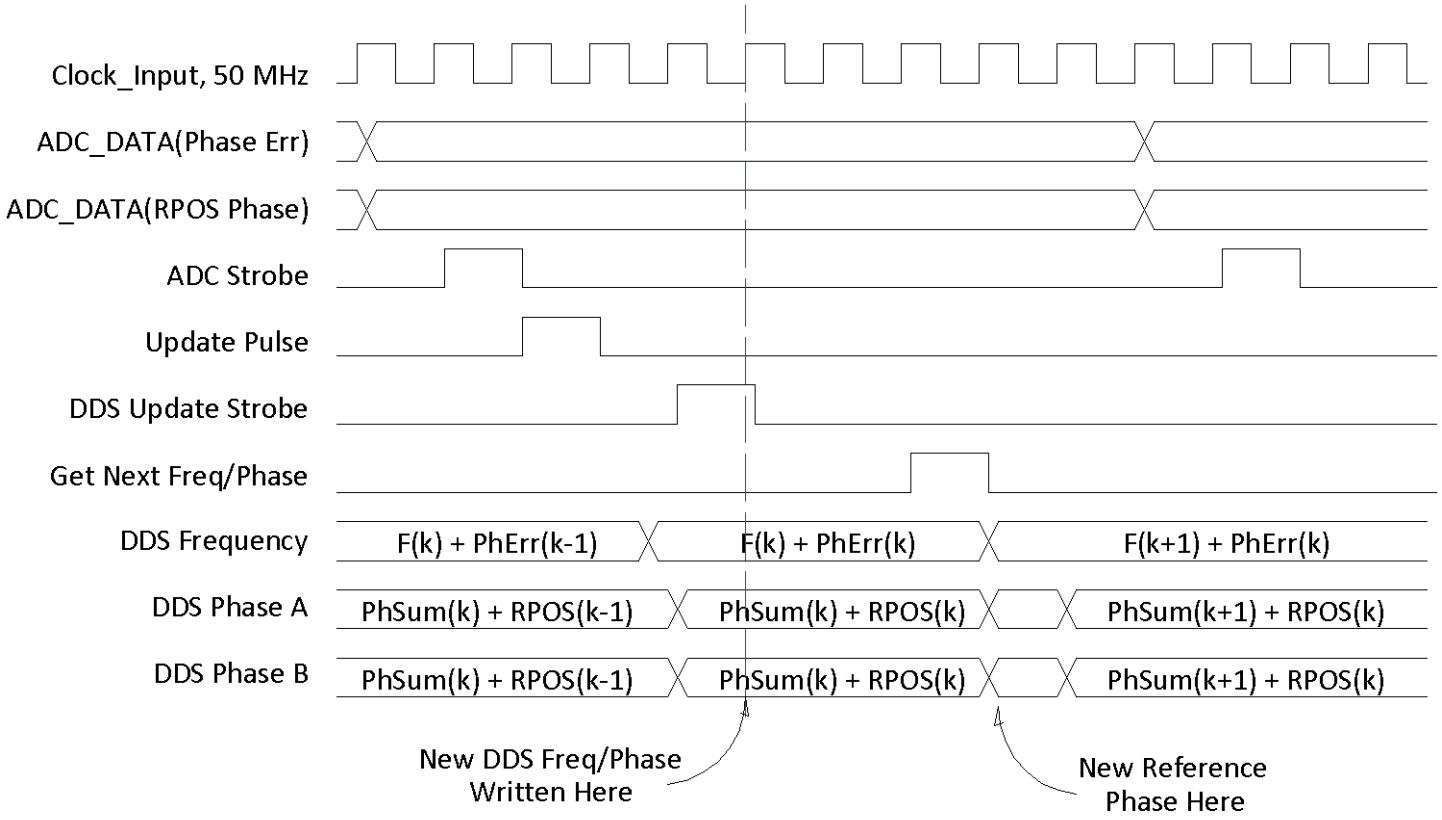

Figure III.3.3.1 DDS frequency and phase computation timing.