1 Introduction

This note is intended to be a stand-alone document describing, in detail, the memory/register space of the XTC 2 TDC mezzanine card. This information will be included later in a note detailing the functionality of the entire card.

Note: All 6-bin window number references in this note are based on the range 0 to 5 (as opposed to 1 to 6), and all not-sure window number references are based on the range 1 to 5. This is illustrated in Figure 1.

Figure 1: This diagram shows how all six time bins are laid out and illustrates the numbering scheme used in this document for normal windows and not-sure windows.

2 Overview

The VME bus consists of 13 signals, 8 of which are bidirectional data lines. These signals interface with the TDC through mezzanine connectors 2 and 3 (J1 and J2 on the XTC 2 PCBs). They connect to the FPGA-Programmer CPLD (U23 on the XTC 2 PCBs), and from there they are routed to the Kitchen Sink FPGA (U2 on the XTC 2 PCBs).

The XTC 2 card contains 64 registers, most of which are located in the Kitchen Sink FPGA. Tables 1 and 2 contain summaries of the entire register space. All of these registers are accessible through VME bus reads and writes. The address space is 0x02041Cyy, where yy represents all values between 0x00 and 0xFC that are divisible by 4. The lowest two bits of the yy byte are not passed to the XTC card, so the card sees address values 0x00 through 0x3F. When reading and writing data, only the most significant 8 bits of each 32-bit integer are used. As a result, bit shifting data values in the C code is necessary.
### Table 1: XTC 2 Prototype Board Register Space Summary

*The firmware version number registers were added after a couple versions had already been created. If zero is read from either of these registers, or if data can be written and then read back from either, the firmware predates the addition of these registers. The version numbers begin with 1 and should be changed for each firmware revision.*

**XTC 2 prototype boards 2, 3, and 4 have an alternate (ALT) delay line chip configuration: the chips for all registers marked with a ** were replaced by 0.5ns/step chips. Doubling the values stored in these registers provides adequate compensation.**
<table>
<thead>
<tr>
<th>Address VME</th>
<th>XTC</th>
<th>R/W</th>
<th>First Version</th>
<th>Initial Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>xFC</td>
<td>x3F</td>
<td>R/W</td>
<td>---</td>
<td>16</td>
<td>x00</td>
</tr>
<tr>
<td>xF8</td>
<td>x3E</td>
<td>R/W</td>
<td>---</td>
<td>16</td>
<td>x00</td>
</tr>
<tr>
<td>xF4</td>
<td>x3D</td>
<td>R/W</td>
<td>---</td>
<td>16</td>
<td>x00</td>
</tr>
<tr>
<td>xF0</td>
<td>x3C</td>
<td>R/W</td>
<td>---</td>
<td>16</td>
<td>x00</td>
</tr>
<tr>
<td>xEC</td>
<td>x3B</td>
<td>W</td>
<td>---</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td>xE8</td>
<td>x3A</td>
<td>R</td>
<td>---</td>
<td>Varies</td>
<td>Read CPLD/Flash Status</td>
</tr>
<tr>
<td>xE4</td>
<td>x39</td>
<td>W</td>
<td>---</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>xE0</td>
<td>x38</td>
<td>R</td>
<td>---</td>
<td>1</td>
<td>Ver. #</td>
</tr>
<tr>
<td>xB8-xDC</td>
<td>x22-x37</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>UNUSED</td>
</tr>
<tr>
<td>xAC</td>
<td>x2B</td>
<td>W</td>
<td>32</td>
<td>N/A</td>
<td>x00</td>
</tr>
<tr>
<td>xA8</td>
<td>x2A</td>
<td>R</td>
<td>32</td>
<td>N/A</td>
<td>x00</td>
</tr>
<tr>
<td>xA4</td>
<td>x20</td>
<td>R/W</td>
<td>32</td>
<td>N/A</td>
<td>x00</td>
</tr>
<tr>
<td>xA0</td>
<td>x28</td>
<td>R/W</td>
<td>32</td>
<td>N/A</td>
<td>x00</td>
</tr>
<tr>
<td>x9C</td>
<td>x27</td>
<td>R/W</td>
<td>32</td>
<td>N/A</td>
<td>x03</td>
</tr>
<tr>
<td>x98</td>
<td>x26</td>
<td>R/W</td>
<td>32</td>
<td>N/A</td>
<td>x07</td>
</tr>
<tr>
<td>x94</td>
<td>x25</td>
<td>R/W</td>
<td>32</td>
<td>N/A</td>
<td>x05</td>
</tr>
<tr>
<td>x90</td>
<td>x24</td>
<td>R/W</td>
<td>32</td>
<td>N/A</td>
<td>x15</td>
</tr>
<tr>
<td>x8C</td>
<td>x23</td>
<td>R/W</td>
<td>32</td>
<td>N/A</td>
<td>x00</td>
</tr>
<tr>
<td>x88</td>
<td>x22</td>
<td>R/W</td>
<td>32</td>
<td>N/A</td>
<td>x00</td>
</tr>
<tr>
<td>x84</td>
<td>x21</td>
<td>R/W</td>
<td>23</td>
<td>N/A</td>
<td>x00</td>
</tr>
<tr>
<td>x80</td>
<td>x20</td>
<td>R/W</td>
<td>16</td>
<td>N/A</td>
<td>xFF</td>
</tr>
<tr>
<td>x7C</td>
<td>x1F</td>
<td>R/W</td>
<td>17</td>
<td>xFF</td>
<td>xFF</td>
</tr>
<tr>
<td>x78</td>
<td>x1E</td>
<td>R/W</td>
<td>16</td>
<td>x00</td>
<td>x00</td>
</tr>
<tr>
<td>x74</td>
<td>x1D</td>
<td>R</td>
<td>16</td>
<td>x02</td>
<td>x06</td>
</tr>
<tr>
<td>x70</td>
<td>x1C</td>
<td>R</td>
<td>16</td>
<td>Ser. #</td>
<td>Ser. #</td>
</tr>
<tr>
<td>x6C</td>
<td>x1B</td>
<td>R</td>
<td>16</td>
<td>Ser. #</td>
<td>Ser. #</td>
</tr>
<tr>
<td>x68</td>
<td>x1A</td>
<td>R/W</td>
<td>16</td>
<td>N/A</td>
<td>x0A</td>
</tr>
<tr>
<td>x64</td>
<td>x19</td>
<td>R/W</td>
<td>16</td>
<td>N/A</td>
<td>x0A</td>
</tr>
<tr>
<td>x60</td>
<td>x18</td>
<td>R/W</td>
<td>16</td>
<td>N/A</td>
<td>x0C</td>
</tr>
<tr>
<td>x5C</td>
<td>x17</td>
<td>R/W</td>
<td>16</td>
<td>N/A</td>
<td>x0A</td>
</tr>
<tr>
<td>x58</td>
<td>x16</td>
<td>R/W</td>
<td>16</td>
<td>N/A</td>
<td>x0A</td>
</tr>
<tr>
<td>x54</td>
<td>x15</td>
<td>R</td>
<td>16</td>
<td>Ver. #</td>
<td>Ver. #</td>
</tr>
<tr>
<td>x50</td>
<td>x14</td>
<td>R</td>
<td>16</td>
<td>Ver. #</td>
<td>Ver. #</td>
</tr>
<tr>
<td>x4C</td>
<td>x13</td>
<td>R</td>
<td>19</td>
<td>x00</td>
<td>x00</td>
</tr>
<tr>
<td>x48</td>
<td>x12</td>
<td>R</td>
<td>19</td>
<td>x00</td>
<td>x00</td>
</tr>
<tr>
<td>x44</td>
<td>x11</td>
<td>R</td>
<td>19</td>
<td>x00</td>
<td>x00</td>
</tr>
<tr>
<td>x40</td>
<td>x10</td>
<td>R</td>
<td>19</td>
<td>x00</td>
<td>x00</td>
</tr>
<tr>
<td>x3C</td>
<td>x0F</td>
<td>R</td>
<td>16</td>
<td>x00</td>
<td>x00</td>
</tr>
<tr>
<td>x38</td>
<td>x0E</td>
<td>W</td>
<td>16</td>
<td>x00</td>
<td>x00</td>
</tr>
<tr>
<td>x34</td>
<td>x0D</td>
<td>W</td>
<td>16</td>
<td>x00</td>
<td>x00</td>
</tr>
<tr>
<td>x30</td>
<td>x0C</td>
<td>W</td>
<td>16</td>
<td>x03</td>
<td>x03</td>
</tr>
<tr>
<td>x2C</td>
<td>x0B</td>
<td>R/W</td>
<td>16</td>
<td>x00</td>
<td>x14</td>
</tr>
<tr>
<td>x28</td>
<td>x0A</td>
<td>R/W</td>
<td>16</td>
<td>x00</td>
<td>x15</td>
</tr>
<tr>
<td>x24</td>
<td>x09</td>
<td>R/W</td>
<td>16</td>
<td>x00</td>
<td>x12</td>
</tr>
<tr>
<td>x20</td>
<td>x08</td>
<td>R/W</td>
<td>16</td>
<td>x7F</td>
<td>x87</td>
</tr>
<tr>
<td>x1C</td>
<td>x07</td>
<td>R/W</td>
<td>16</td>
<td>xBC</td>
<td>x00</td>
</tr>
<tr>
<td>x18</td>
<td>x06</td>
<td>R/W</td>
<td>16</td>
<td>x70</td>
<td>x10</td>
</tr>
<tr>
<td>x14</td>
<td>x05</td>
<td>R/W</td>
<td>16</td>
<td>x62</td>
<td>x58</td>
</tr>
<tr>
<td>x10</td>
<td>x04</td>
<td>R/W</td>
<td>16</td>
<td>x06</td>
<td>x03</td>
</tr>
<tr>
<td>x0C</td>
<td>x03</td>
<td>R</td>
<td>25</td>
<td>N/A</td>
<td>xF1</td>
</tr>
<tr>
<td>x08</td>
<td>x02</td>
<td>R/W</td>
<td>16</td>
<td>xC8</td>
<td>xC8</td>
</tr>
<tr>
<td>x04</td>
<td>x01</td>
<td>R/W</td>
<td>16</td>
<td>x06</td>
<td>x03</td>
</tr>
<tr>
<td>x00</td>
<td>x00</td>
<td>R/W</td>
<td>16</td>
<td>x5C</td>
<td>x4A</td>
</tr>
</tbody>
</table>

**Table 2: XTC 2 Production Board Register Space Summary**

*Firmware version numbers should be incremented for each firmware revision.*
**XTC 2 production boards 0, 132, 133, 144, 145, and 146 have an alternate (ALT) delay line chip configuration: chips for registers 0x04 and 0x10 were replaced by 0.5ns/step chips, and the chip for register 0x20 was replaced by a 1ns/step chip.**

Note: Version numbers 0-15 (0x00-0x0F) are reserved for prototype FPGA designs. Production board FPGA firmware design numbers begin with 16 (0x10).

The first address column of Tables 1 and 2 contains the last two hex digits of the VME address. The second address column is the address seen by the XTC 2 card on the data lines. The R/W column indicates whether the register is read-only, write-only, or read/write. The “First Version” columns specify the first Kitchen Sink FPGA or FPGA_Prog CPLD firmware version/revision number in which each register is available. The “Initial Value” column indicates the value stored/available in each register immediately after power-up.

Note: All registers in this note are referenced using VME addresses unless otherwise specified.

### 3 Detailed Register Description

All registers except 0xE0 – 0xFC are located in the Kitchen Sink FPGA and cannot be read from or written to until the FPGA has been configured. The Kitchen Sink FPGA reprograms all of the delay line chips—as well as all other register values used by the internal FPGA logic (e.g., the not-sure window width registers)—only when data is written to one of those registers (0x00-0x30, 0x58-0x68, and 0x78-0x9C).

Registers 0xE0 – 0xFC are used by the FPGA-Programmer (FPGA_Prog) CPLD to read, erase, and write to the flash memory. These registers only require that this CPLD be configured.

Note: all register definitions given here are valid for the following firmware versions:
- **FPGA_Prog CPLD:** 16
- **Output Buffers CPLD:** 1
- **2-Bin Prototype Kitchen Sink FPGA:** 2 (2-Bin Data FPGA 0)
- **2-Bin Production Kitchen Sink FPGA:** 19 (2-Bin Data FPGA 1)
- **6-Bin Prototype Kitchen Sink FPGA:** 7 (6-Bin Data FPGA 3)
- **6-Bin Production Kitchen Sink FPGA:** 33 (6-Bin Data FPGA 4)

### 3.1 Delay Line Chip Registers (0x00 – 0x2C)

Eleven registers, all of which can be read from and written to, store the 8-bit values used to program the delay line chips. Registers 0x00 through 0x20, excluding 0x0C, have the same functionality as the corresponding registers in the Michigan XTC card and are the only delay line chip registers necessary for running the XTC 2 two-bin design. Registers
0x24 through 0x2C add the 3 extra window signals necessary for the 6-bin design. Before sending any data through the board, all registers necessary for the design to be used should be programmed first if their values are going to be different from the initial values listed in Tables 1 and 2. Figures 2 and 3 show which window edges are affected by each register.

All window signals are derived from CDFCLK, including DLSTCL, so delaying this signal will delay all others shown in these two figures. All of the window signals, excluding DLSTCL, are dependent on one another, meaning that if one window edge is delayed, all subsequent window edges will be delayed by an equal amount. DLSTCL is only dependent on CDFCLK, so it remains unaffected by changes made to the window edges.

Figure 2: This figure shows which registers affect each window edge in the 2-bin design. The VME address for each register is given in hex next to the signal name.

All of the delay line chips have a typical step-zero delay of around 16.5ns. This means that setting the register value to 0x00 will delay the signal by 16.5ns. Incrementing the register value by one will then increase the signal delay by the step size (0.25ns, 0.5ns, 1ns, 2ns, or 5ns, depending on the chip). Thus, the total signal delay through any chip will be:

\[
\text{Total Signal Delay} = (\text{Register Value}) \times (\text{Step Size}) + 16.5
\]

Note: delay-line chip U10, corresponding to register 0x0C, was removed from all production boards because it was being used at frequencies beyond the specifications for the chip, which caused intermittent errors. A jumper was placed on each board to connect signals STR22 and CLK22 together. Delays were then added to the appropriate
signals in the Output_Buffers CPLD and KitchenSink FPGA to compensate for the removal of this chip.

Note: register 0x0C in the production board KS FPGA firmware was redefined in version 25. See section 3.5 for details.

Register Functionality:
0x00 – Initial CDFCLK Delay
  Step Size:
  | Standard Prototype: 1ns | Alternate Prototype: 0.5ns |
  | Standard Production: 0.5ns | Alternate Production: 0.5ns |
  Board Signal Affected: CDFCLK
  Delay Chip Ref. Designator: U21
  Latch Enable Signal: LE_A
  Description: This initial delay for the CDFCLK signal is used to compensate for the time required for signals to reach the XTC 2 card from the detector.

0x04 – CDFBC Delay
  Step Size:
  | Standard Prototype: 1ns | Alternate Prototype: 1ns |
  | Standard Production: 5ns | Alternate Production: 0.5ns |
  Board Signal Affected: XTLBC
  Delay Chip Ref. Designator: U20
  Latch Enable Signal: LE_B
  Description: This is used to delay the BC signal to synchronize it with the CDFCLK. If the CDFCLK delay is changed, the value in this register should probably be changed as well.

0x08 – PLL Input Delay
  Step Size:
  | Standard Prototype: 0.5ns | Alternate Prototype: 0.5ns |
  | Standard Production: 0.5ns | Alternate Production: 0.5ns |
  Board Signal Affected: PLLSYNC
  Delay Chip Ref. Designator: U11
  Latch Enable Signal: LE_C
  Description: This is used for delaying the CDFCLK signal for the PLL, as well as creating the SYNCLR signal.
This definition of register 0x0C is now only valid for the prototype boards:

0x0C – PLL Output Delay (Chip removed from production boards)

**Step Size:**

<table>
<thead>
<tr>
<th>Standard Prototype</th>
<th>Alternate Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25ns</td>
<td>0.25ns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Standard Production</th>
<th>Alternate Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25ns</td>
<td>0.25ns</td>
</tr>
</tbody>
</table>

**Board Signal Affected:** STR22

**Delay Chip Ref. Designator:** U10

**Latch Enable Signal:** LE_D

**Description:** This is used to shift the 22-ns board clock around in time. U10 was removed from the production boards, but the register remains in the firmware.

0x10 – CDFB0 Delay

**Step Size:**

<table>
<thead>
<tr>
<th>Standard Prototype</th>
<th>Alternate Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1ns</td>
<td>1ns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Standard Production</th>
<th>Alternate Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>5ns</td>
<td>0.5ns</td>
</tr>
</tbody>
</table>

**Board Signal Affected:** XTLB0

**Delay Chip Ref. Designator:** U18

**Latch Enable Signal:** LE_E

**Description:** This is used to delay the B0 signal to synchronize it with the CDFCLK. If the CDFCLK delay is changed, the value in this register should probably be changed as well.

0x14 – Clock 1 Delay

**Step Size:**

<table>
<thead>
<tr>
<th>Standard Prototype</th>
<th>Alternate Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1ns</td>
<td>0.5ns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Standard Production</th>
<th>Alternate Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5ns</td>
<td>0.5ns</td>
</tr>
</tbody>
</table>

**Board Signal Affected:** DELCLK[1]

**Delay Chip Ref. Designator:** U19

**Latch Enable Signal:** LE_F

**Description:** This is used to delay the CDFCLK signal (board signal DELCLK[0]) to create the trailing edge of window 0/leading edge of the not-sure part of window 1.

0x18 – Clock 2 Delay

**Step Size:**

<table>
<thead>
<tr>
<th>Standard Prototype</th>
<th>Alternate Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1ns</td>
<td>0.5ns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Standard Production</th>
<th>Alternate Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5ns</td>
<td>0.5ns</td>
</tr>
</tbody>
</table>

**Board Signal Affected:** DELCLK[2]

**Delay Chip Ref. Designator:** U14

**Latch Enable Signal:** LE_G

**Description:** This is used to delay the DELCLK[1] signal to create the trailing edge of window 1/leading edge of the not-sure part of window 2.
0x1C – Prompt Set/Clear Delay (PRSTCL)

**Step Size:**
- **Standard Prototype:** 1ns
- **Alternate Prototype:** 0.5ns
- **Standard Production:** 0.5ns
- **Alternate Production:** 0.5ns

**Board Signal Affected:** PRSTCL
**Delay Chip Ref. Designator:** U16
**Latch Enable Signal:** LE_H

**Description:** This is used to delay the CDFCLK signal (board signal DELCLK[0]) to create a short (~10ns) pulse that stores the detected prompt bits in output flip-flops (“set” functionality) while clearing out the prompt bit detection flip-flops (“clear” functionality). This signal is not used in the 6-bin design.

0x20 – Delayed Set/Clear Delay (DLSTCL)

**Step Size:**
- **Standard Prototype:** 1ns
- **Alternate Prototype:** 1ns
- **Standard Production:** 2ns
- **Alternate Production:** 1ns

**Board Signal Affected:** DLSTCL
**Delay Chip Ref. Designator:** U15
**Latch Enable Signal:** LE_I

**Description:** This is used to delay the CDFCLK signal (board signal DELCLK[0]) to create a short (~10ns) pulse that, in the 2-bin design, stores the detected delayed bits in output flip-flops (“set” functionality) while clearing out the delayed bit detection flip-flops (“clear” functionality). It also marks the end of the delayed window. In the 6-bin design, DLSTCL marks the end of window 5.

0x24 – Clock 3 Delay

**Step Size:**
- **Standard Prototype:** 1ns
- **Alternate Prototype:** 0.5ns
- **Standard Production:** 0.5ns
- **Alternate Production:** 0.5ns

**Board Signal Affected:** DELCLK[3]
**Delay Chip Ref. Designator:** U17
**Latch Enable Signal:** LE_J

**Description:** This is used to delay the DELCLK[2] signal to create the trailing edge of window 2/leading edge of the not-sure part of window 3. This signal is not used in the 2-bin design.
0x28 – Clock 4 Delay

**Step Size:**

<table>
<thead>
<tr>
<th></th>
<th>Standard Prototype</th>
<th>Alternate Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prototype</strong></td>
<td>1ns</td>
<td>0.5ns</td>
</tr>
<tr>
<td><strong>Production</strong></td>
<td>0.5ns</td>
<td>0.5ns</td>
</tr>
</tbody>
</table>

**Board Signal Affected:** DELCLK[4]

**Delay Chip Ref. Designator:** U12

**Latch Enable Signal:** LE_K

**Description:** This is used to delay the DELCLK[3] signal to create the trailing edge of window 3/leading edge of the not-sure part of window 4. This signal is not used in the 2-bin design.

0x2C – Clock 5 Delay

**Step Size:**

<table>
<thead>
<tr>
<th></th>
<th>Standard Prototype</th>
<th>Alternate Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prototype</strong></td>
<td>1ns</td>
<td>0.5ns</td>
</tr>
<tr>
<td><strong>Production</strong></td>
<td>0.5ns</td>
<td>0.5ns</td>
</tr>
</tbody>
</table>

**Board Signal Affected:** DELCLK[5]

**Delay Chip Ref. Designator:** U13

**Latch Enable Signal:** LE_L

**Description:** This is used to delay the DELCLK[4] signal to create the trailing edge of window 4/leading edge of the not-sure part of window 5. This signal is not used in the 2-bin design.

### 3.2 Level 2 Buffer Registers (0x30 – 0x4C)

These registers are used for accessing the level 2 data stored in the 4 Kitchen Sink memory buffers when a level 1 accept is issued. The **Valid Value Range** fields listed below should be followed when writing to these registers. Unintended effects may occur if this is not done.

**Note:** the first byte (address 0) of each level 2 buffer contains a bunch count value in the range 0-158, which indicates the number of CDFCLK cycles that have elapsed since the bunch 0 crossing that occurred before the data stored in the buffer was written to the buffer. This count value is obtained by subtracting the length of the level 1 pipeline, stored in register 0x30, from the value of the counter when the level 1 accept is issued. For example, if the level 1 pipeline length is set to 10, and if a level 1 accept is issued 28 CDFCLKs after B0, the count value stored will be 28 – 10 = 18.

**Note:** registers 0x40 – 0x4C were formerly known as the Buffer Status Registers. They were changed to the Buffer Content Registers in the 6-bin Kitchen Sink version 19 firmware.
The procedure for reading an entire buffer containing all of the data from a single beam crossing (396ns) is as follows:

1. Write the buffer number to register 0x34.
2. Do either one of the following (the firmware still supports the old way):
   a. [Old way:] Do the following for each byte in the buffer (72 times total): write the address of the byte to be read to register 0x38, and then retrieve the byte by reading register 0x3C.
   b. [New way:] Write the address of the first byte to be read (0x01) to register 0x38, and then read register 0x3C for all bytes in the buffer (72 times). The byte address is automatically updated (add 12; see the definition of register 0x38 for the entire sequence) after each register operation, so it is unnecessary to keep writing to register 0x38 after the first time.

Register Functionality:
0x30 – Level 1 Accept Delay Value (in CDFCLK periods)
   *Valid Value Range:* 0x00 – 0xFF (0 – 255)
   *Functionality:* Write (can be read as well)
   *Description:* The value stored in this register indicates the number of CDFCLK periods between the generation of a set of accepted data and the level 1 accept signal that indicates that that data were accepted. So, if the value stored is 12, when the logic sees a level 1 accept event, it will store the data from 12 CDFCLK cycles ago in a buffer.

0x34 – Buffer Number
   *Valid Value Range:* 0x00 – 0x03 (0 – 3)
   *Functionality:* Write (can be read as well)
   *Description:* The value stored in this register determines the buffer from which data is read when executing read commands via registers 0x38 and 0x3C.
0x38 – Buffer Byte Address

*Valid Value Range:* 0x00 – 0x48 (0 – 72)

*Functionality:* Write (can be read as well)

*Description:* The value stored in this register selects the byte to be read from the buffer indicated by register 0x34. Writing to this register will trigger an automatic read of the byte from the buffer and store the data in register 0x3C. The value in this register is then automatically updated to the next value in the following sequence:

\[
0 \rightarrow 1 \rightarrow 13 \rightarrow 25 \rightarrow 37 \rightarrow 49 \rightarrow 61 \\
2 \rightarrow 14 \rightarrow 26 \rightarrow 38 \rightarrow 50 \rightarrow 62 \\
3 \rightarrow 15 \rightarrow 27 \rightarrow 39 \rightarrow 51 \rightarrow 63 \\
4 \rightarrow 16 \rightarrow 28 \rightarrow 40 \rightarrow 52 \rightarrow 64 \\
5 \rightarrow 17 \rightarrow 29 \rightarrow 41 \rightarrow 53 \rightarrow 65 \\
6 \rightarrow 18 \rightarrow 30 \rightarrow 42 \rightarrow 54 \rightarrow 66 \\
7 \rightarrow 19 \rightarrow 31 \rightarrow 43 \rightarrow 55 \rightarrow 67 \\
8 \rightarrow 20 \rightarrow 32 \rightarrow 44 \rightarrow 56 \rightarrow 68 \\
9 \rightarrow 21 \rightarrow 33 \rightarrow 45 \rightarrow 57 \rightarrow 69 \\
10 \rightarrow 22 \rightarrow 34 \rightarrow 46 \rightarrow 58 \rightarrow 70 \\
11 \rightarrow 23 \rightarrow 35 \rightarrow 47 \rightarrow 59 \rightarrow 71 \\
12 \rightarrow 24 \rightarrow 36 \rightarrow 48 \rightarrow 60 \rightarrow 72 \\
0 \rightarrow
\]

See the note above in this section about the first byte of each level 2 buffer.

0x3C – Read Buffer Byte

*Valid Value Range:* N/A

*Functionality:* Read-Only

*Description:* When an address value is written to register 0x38, the byte is read from the buffer indicated by register 0x34 and stored in this register. The byte address, stored in register 0x38, is automatically incremented each time this register is read.

0x40 – Buffer 0 Content Bits

*Valid Value Range:* N/A

*Functionality:* Read-Only

*Description:* This register uses 6 bits (the two most-significant bits are always 0) to indicate, by channel grouping, where non-zero data is located in buffer 0. The mapping between bits and channels is as follows: 5\(\Rightarrow\)95:80, 4\(\Rightarrow\)79:64, 3\(\Rightarrow\)63:48, 2\(\Rightarrow\)47:32, 1\(\Rightarrow\)31:16, 0\(\Rightarrow\)15:0. For example, if any bit for any of the six windows of channels 0 through 15 is a one, then bit 0 of this register will be 1. Otherwise, it will be 0.
0x44 – Buffer 1 Status

Valid Value Range: N/A
Functionality: Read-Only
Description: This register uses 6 bits (the two most-significant bits are always 0) to indicate, by channel grouping, where non-zero data is located in buffer 1. The mapping between bits and channels is as follows: 5\rightarrow 95:80, 4\rightarrow 79:64, 3\rightarrow 63:48, 2\rightarrow 47:32, 1\rightarrow 31:16, 0\rightarrow 15:0. For example, if any bit for any of the six windows of channels 80 through 95 is a one, then bit 5 of this register will be 1. Otherwise, it will be 0.

0x48 – Buffer 2 Status

Valid Value Range: N/A
Functionality: Read-Only
Description: This register uses 6 bits (the two most-significant bits are always 0) to indicate, by channel grouping, where non-zero data is located in buffer 2. The mapping between bits and channels is as follows: 5\rightarrow 95:80, 4\rightarrow 79:64, 3\rightarrow 63:48, 2\rightarrow 47:32, 1\rightarrow 31:16, 0\rightarrow 15:0. For example, if any bit for any of the six windows of channels 48 through 63 is a one, then bit 3 of this register will be 1. Otherwise, it will be 0.

0x4C – Buffer 3 Status

Valid Value Range: N/A
Functionality: Read-Only
Description: This register uses 6 bits (the two most-significant bits are always 0) to indicate, by channel grouping, where non-zero data is located in buffer 3. The mapping between bits and channels is as follows: 5\rightarrow 95:80, 4\rightarrow 79:64, 3\rightarrow 63:48, 2\rightarrow 47:32, 1\rightarrow 31:16, 0\rightarrow 15:0. For example, if any bit for any of the six windows of channels 16 through 31 is a one, then bit 1 of this register will be 1. Otherwise, it will be 0.

3.3 Firmware Version Number Registers (0x50, 0x54, 0xE0)

The firmware version numbers of both FPGA designs and the FPGA_Prog CPLD design are stored in these read-only registers. 0x50 and 0x54 are located in the Kitchen Sink, while 0xE0 is implemented in the FPGA_Prog CPLD. The version numbers begin with 1 and should be incremented each time the firmware is revised. Note that these registers were undefined in the first few firmware versions that were created. As a result, if these registers return a value of 0x00 when read (except for the 2-bin Data FPGA design, which has not been revised), or if the value stored in the register can be modified (meaning any value can be written and then successfully read back), the firmware version predates the addition of these registers. Also note that if a change is made only to the Data FPGA firmware, the Data FPGA version number must be incremented, requiring that the Kitchen Sink FPGA firmware be resynthesized as well (since both FPGA version registers are implemented in the Kitchen Sink).
**Current Design Number Definitions:**

### 2-Bin Data FPGA

<table>
<thead>
<tr>
<th>Version 0x00</th>
<th>• First version to work</th>
</tr>
</thead>
</table>
| Version 0x01 | • Added a counter to the Output_Selector component that sets all data outputs to high impedance if shift_in_bit is not high for at least 1 out of every 64 shift_clk cycles (it should be high 1 in every 16 shift_clk cycles). This was added to provide a way of setting all the data outputs to high impedance using a Kitchen Sink register: just stop sending the shift_in_bit pulse.  
• Added the maxskew attribute to the shift_clk signal |

### 6-Bin Data FPGA

<table>
<thead>
<tr>
<th>Version 0x00</th>
<th>• First version to work</th>
</tr>
</thead>
</table>
| Version 0x01 | • Added not-sure windows (using New_Channel component instead of Channel)  
• Added port ldel(3)  
• Switched to using a gray counting scheme for the window signals (ldel)  
• Added not-sure flip-flops and logic to the Channel component |
| Version 0x02 | • Switched to using 6 clear signals instead of 3, as well as dlstcl  
• Removed ports clr01, clr23, clr45  
• Added port clk_out  
• Added port dlstcl  
• Added internal signal dlstcl_delayed  
• Modified Channel logic appropriately  
• Switched back to using the Channel component instead of the New_Channel component |
| Version 0x03 | • Added not-sure window enable functionality  
• Added port ns_enable  
• Modified Channel logic appropriately  
• Optimized locations of Channel components to minimize channel-to-channel propagation delay differences |
| Version 0x04 | • Added a counter to the Output_Selector component that sets all data outputs to high impedance if shift_in_bit is not high for at least 1 out of every 64 shift_clk cycles (it should be high 1 in every 16 shift_clk cycles). This was added to provide a way of setting all the data outputs to high impedance using a Kitchen Sink register: just stop sending the shift_in_bit pulse.  
• Added the maxskew attribute to the shift_clk signal |

### 2-Bin Kitchen Sink FPGA

| Version 0x00 | • First version to work  
• Use with v. 0x00 of the 2-Bin Data FPGA (prototype board) |
**Version 0x01**
- Added keep_hierarchies attributes to all files
- Added ports str22, l1b0, l1b1, and fpga_reprog_n
- Added L1 accept address lines
- Added firmware version registers (VME 0x50, 0x54)
- Added register for storing the number of time bins (VME 0x74)
- Added register for the Transition Board FPGA Reprogram signal (VME 0x78)
- Changed use of cdfl1a to active-high
- Added flip-flop for detecting edges of cdfl1a
- Added logic for storing the L1 accept address
- Revamped logic for the buffer status registers and the state machine that prohibits L1 accepts for filled L2 buffers
- Modified the clear signal definition to allow writing to all L2 buffers regardless of their status
- Changed the name of the 22-ns clock for the Data_Auto component from clk22 to str22
- Set up clk132 (clk132_int) to pass through the Sync component, where it is delayed by an appropriate number of clk22 clock cycles (becomes clk132_dly), before going into the CDFL1A_Logic component. This properly aligns the 132-ns clock with the time at which data should be stored in the L2 buffers.
- Modified Reg_Prog_Logic and TDC_BusLogic so the registers are only programmed when necessary rather than on all writes
- Modified TDC_BusLogic so the registers are automatically programmed after being initialized
- Added VME register initialization values
- Added a buffer to delay the str22 clock signal in the CDFL1A_Logic component to eliminate the flip-flop input setup time errors on last_clk132 in Data_Auto
- Use with v. 0x00 of 2-Bin Data FPGA (prototype board)

**Version 0x02**
- Modified the L2 buffers to include the bunch count number, which indicates how many CDFCLK cycles after B0 the recorded data occurred. This value is stored in location 0 of each buffer, which means that the data in each buffer is now in locations 1-72 instead of 0-71.
  - Modified Buffers component
  - Modified CDFL1A_Logic component
  - Modified Buffer_Read_Logic component
- Use with v. 0x00 of 2-Bin Data FPGA (prototype board)
| Version 0x10  | • Added board serial number registers (VME 0x6C, 0x70)  
| (Created from Version 0x02) | • Added port board_serial_num  
| | • Modified TDC_Bus_Logic appropriately  
| | • Changed pin assignment of port le_c from P139 to P125  
| | • Modified the initial register contents in Reg_Init for the standard production boards  
| | • Use with v. 0x00 of 2-Bin Data FPGA (production board) |

| Version 0x11 | • Modified the functionality of registers 0x38 and 0x3C: after each buffer read operation, the buffer address is incremented  
| | • Modified Buffer_Read_Logi component  
| | • Modified the Data_Align component in the CDIF1A_Lo gic so that the 32 data bits that come directly from the data port are delayed by a few nanoseconds before being sent to the BRAM  
| | • Modified the Sync component to allow enabling and disabling of the shift_in_bit signal  
| | • Added shift_in_bit enable register (VME 0x7C)  
| | • Changed the initial value of register 0x7C in Reg_Init from 0x00 to 0xFF  
| | • Changed the definition of prog_regs_clk in TDC_Bus_Logic so the registers in Reg_Prog_Logic are reprogrammed on a write to register 0x7C  
| | • Added a process to generate an output test pattern (alternating ones and zeros every output wrd0 period) when shift_in_bit is disabled (shift_in_bit_enable = '0')  
| | • Changed direction of the data port from 'in' to 'inout'  
| | • Updated the values in Reg_Init to reflect the delay line chip changes in the standard production boards  
| | • Use with v. 0x01 of 2-Bin Data FPGA (production board) |

| Version 0x12 | • Changed the NUMBER_OF_EXTRA_BUFFERS in STR22_Delay of the CDIF1A_Lo gic component from 0 to 5 to compensate for the loss of the DS1023S-25 delay line chip on the XTC 2 board, which delayed clk22 by ~8ns, creating str22. The extra delay added should be between about 3.5ns and 10ns.  
| | • Use with v. 0x01 of 2-Bin Data FPGA (production board) |
**Version 0x13**
- Added the Buffer_Contents component (which includes the Contents_Storage and Storage_Regs components) to the CDFL1A_Logic component
- Added four buffer_x_contents signals that travel up the hierarchy from Buffer_Contents and into the TDC_BusLogic component
- Removed the old functionality of registers 0x40 - 0x4C and replaced it with logic that specifies the contents of the buffers
  - All of the signals used to generate the buffer status signals (mostly TDC bus signals) were removed from the Buffers and CDFL1A_Logic components
  - The buffer_status signals were removed from these, as well as from the main architecture and the TDC_BusLogic component
  - These registers are now read-only
- Changed the name of signal we to we_int and write_buf_addr to write_buf_addr_int in the Buffers component. Ports we and write_buf_addr were then added so Buffer_Contents has access to them.
  - Added a delay to clk132_dly in the BRAM component to create the clk132_dly_dly signal, which is then used as the clock for the BRAM write address counter so there are no more address setup and hold errors
- Use with v. 0x04 of 6-Bin Data FPGA (production board)

**6-Bin Kitchen Sink FPGA**

| Version 0x00 | First version to work
| Use with v. 0x00 of 6-Bin Data FPGA (prototype board) |
| --- | --- |
| Version 0x01 | Added not-sure windows
  - Added port ldel(3)
  - Switched to using a gray counting scheme for the window signals (ldel)
  - Use with v. 0x01 of 6-Bin Data FPGA (prototype board) |
| Version 0x02 | Added register for simulating L1 accepts (VME 0x6C)
  - Changed use of cdfl1a to active-high
  - Use with v. 0x01 of 6-Bin Data FPGA (prototype board) |
| Version 0x03 | Added L1 accept address lines
  - Removed register for simulating L1 accepts from v. 0x02
  - Added flip-flop for detecting edges of cdfl1a
  - Added logic for storing the L1 accept address
  - Revamped logic for the buffer status registers and the state machine that prohibits L1 accepts for filled L2 buffers
  - Changed the name of the 22-ns clock for the Data_Align component from clk22 to str22
  - Added register 0x78 (Transition Board FPGA Reprogram signal)
  - Use with v. 0x01 of 6-Bin Data FPGA (prototype board) |
| Version 0x04 | Removed ports clr01, clr23, and clr45  
| | Added clk_out signal  
| | Modified the clear signal definition to allow writing to all L2 buffers regardless of their status  
| | Use with v. 0x02 of 6-Bin Data FPGA (prototype board) |
| Version 0x05 | Set up clk132 (clk132_int) to pass through the Sync component, where it is delayed by an appropriate number of clk22 clock cycles (becomes clk132_dly), before going into the CDFL1A Logic component. This properly aligns the 132-ns clock with the time at which data should be stored in the L2 buffers.  
| | Use with v. 0x02 of 6-Bin Data FPGA (prototype board) |
| Version 0x06 | Added register for the number of time bins in the design (VME 0x74)  
| | Added not-sure window enable register (VME 0x80)  
| | Modified reg_data(7) to be used for both register programming and the not-sure window enable  
| | Modified Reg_Prog_Logic and TDC_Bus_Logic so the registers are only programmed when necessary rather than on all writes  
| | Modified TDC_Bus_Logic so the registers are automatically programmed after being initialized  
| | Added VME register initialization values  
| | Added a buffer to delay the str22 clock signal in the CDFL1A Logic component to eliminate the flip-flop input setup time errors on last_clk132 in Data_Align  
| | Replaced the asynchronous generation of b0del with a shift register implementation that also triples the width of the signal  
| | Modified not-sure window widths so their range is closer to 5-25 ns  
| | Use with v. 0x03 of 6-Bin Data FPGA (prototype board) |
| Version 0x07 | Modified the L2 buffers to include the bunch count number, which indicates how many CDFCLK cycles after B0 the recorded data occurred. This value is stored in location 0 of each buffer, which means that the data in each buffer is now in locations 1-72 instead of 0-71.  
| | Modified Buffers component  
| | Modified CDFL1A Logic component  
| | Modified Buffer_Read_Logic component  
| | Use with v. 0x03 of 6-Bin Data FPGA (prototype board) |
### Version 0x10
(Released from Version 0x06)
- Added board serial number registers (VME 0x6C, 0x70)
- Added port board_serial_num
- Modified TDC_Bus_Logic component
  - Changed pin assignment of port le_c from P139 to P125
  - Modified the L2 buffers to include the bunch count number, which indicates how many CDFCLK cycles after B0 the recorded data occurred. This value is stored in location 0 of each buffer, which means that the data in each buffer is now in locations 1-72 instead of 0-71.
  - Modified Buffers component
  - Modified CDFL1A_Logic component
  - Modified Buffer_Read.Logic component
  - Modified the initial register contents in Reg_Init for the standard production boards
- Use with v. 0x03 of 6-Bin Data FPGA (production board)

### Version 0x11
- Modified the functionality of registers 0x38 and 0x3C: after each buffer read operation, the buffer address is incremented
  - Modified Buffer_Read.Logic component
- Modified the Data_Align component in the CDFL1A_Logic so that the 32 data bits that come directly from the data port are delayed by a few nanoseconds before being sent to the BRAM
- Modified the Sync component to allow enabling and disabling of the shift_in_bit signal
  - Added shift_in_bit enable register (VME 0x7C)
  - Changed the initial value of register 0x7C in Reg_Init from 0x00 to 0xFF
  - Changed the definition of prog_regs_clk in TDC_Bus_Logic so the registers in Reg_Prog.Logic are reprogrammed on a write to register 0x7C
  - Added a process to generate an output test pattern (alternating ones and zeros every output wrd0 period) when shift_in_bit is disabled (shift_in_bit_enable = '0')
    - Changed direction of the data port from 'in' to 'inout'
  - Updated the values in Reg_Init to reflect the delay line chip changes in the standard production boards
- Use with v. 0x04 of 6-Bin Data FPGA (production board)

### Version 0x12
- Changed the NUMBER_OF_EXTRA_BUFFERS in STR22_Delay of the CDFL1A_Logic component from 0 to 5 to compensate for the loss of the DS1023S-25 delay line chip on the XTC 2 board, which delayed clk22 by ~8ns, creating str22. The extra delay added should be between about 3.5ns and 10ns.
- Use with v. 0x04 of 6-Bin Data FPGA (production board)
| Version 0x13 | • Added the Buffer_Contents component (which includes the Contents_Storage and Storage_Regs components) to the CDFL1A_Logic component  
• Added four buffer_x_contents signals that travel up the hierarchy from Buffer_Contents and into the TDC_Bus Logic component  
• Removed the old functionality of registers 0x40 - 0x4C and replaced it with logic that specifies the contents of the buffers  
  • All of the signals used to generate the buffer status signals (mostly TDC bus signals) were removed from the Buffers and CDFL1A_Logic components  
  • The buffer_status signals were removed from these, as well as from the main architecture and the TDC_Bus Logic component  
  • These registers are now read-only  
• Changed the name of signal we to we_int and write_buf_addr to write_buf_addr_int in the Buffers component. Ports we and write_buf_addr were then added so Buffer_Contents has access to them.  
• Use with v. 0x04 of 6-Bin Data FPGA (production board) |
| Version 0x14 | • Increased the delays of the clk_out signals in the Window_Maker component  
• Changed the NUMBER_OF_EXTRA_BUFFERS in STR22_Delay of the CDFL1A_Logic component from 5 to 6 to compensate for delaying the clk_out signals in the Window_Maker component  
• Use with v. 0x04 of 6-Bin Data FPGA (production board) |
| Version 0x15 | • Removed the Data_Align component from the CDFL1A_Logic  
• Modified the BRAM component so that 32 bits are written every 22 ns rather than 192 bits every 132 ns  
• Modified the BRAM component so that the input data is buffered with flip-flops  
• Added the Write_Enable_Generator component to the BRAM component  
• Use with v. 0x04 of 6-Bin Data FPGA (production board) |
| Version 0x16 | • Updated the values in Reg_Init to those that will likely be used in the system  
• Changed the NUMBER_OF_EXTRA_BUFFERS in STR22_DELAY of the BRAM component from 1 to 6  
• Changed the NUMBER_OF_EXTRA_BUFFERS in CLK132_DLY_DELAY of the BRAM component from 22 to 29  
• Changed the CLOCK_PULSE_DELAY in the Write_Enable_Generator component from 6 to 1  
• Use with v. 0x04 of 6-Bin Data FPGA (production board) |
| Version 0x17 | • Added register 0x84 (Transition Board Mode)  
• Added port trans_board_mode  
• Reg_Prog_Logic was updated appropriately to include the Transition Board mode register  
• Modified the definition of prog_regs_clk in TDC_Bus_Logic so the registers in Reg_Prog_Logic are also reprogrammed on a write to register 0x84  
• Changed the NUMBER_OF_EXTRA_BUFFERS in CLK132_DLY_DELAY of the BRAM component from 29 to 31  
• Changed the NUMBER_OF_EXTRA_BUFFERS in STR22_DELAY of the BRAM component from 6 to 9  
• Use with v. 0x04 of 6-Bin Data FPGA (production board) |
| --- | --- |
| Version 0x18 | • Changed the sequence of automatic counting of the buffer byte address in the Buffer_Read_Logic component  
• Changed the NUMBER_OF_EXTRA_BUFFERS in STR22_DELAY of the BRAM component from 9 to 5  
• Use with v. 0x04 of 6-Bin Data FPGA (production board) |
| Version 0x19 | • Modified register 0x0C so that it is a read-only register that alternates between values 0xF1 and 0x66 on successive reads  
• Added logic to the TDC_Bus_Logic component to implement the new register 0x0C  
• Use with v. 0x04 of 6-Bin Data FPGA (production board) |
<table>
<thead>
<tr>
<th>Version 0x20</th>
<th></th>
</tr>
</thead>
</table>
| • Revamped the clock system  
  • The clk22 port is now used for all 22-ns clock signals; str22 is no longer used  
  • Removed the Variable Delay components from the BRAM component for delaying the clock signals  
  • Added register 0x8C (Shift_In_Bit Clock Count/Ck132 Delay) to make the shifting of clk132 register-selectable  
  • Added Config_Delay_32 components for clk132_dly and clk22 to create the delayed 132-ns and 22-ns clocks for the BRAM component  
  • Added registers 0x90 (BRAM Clk132 Delay) and 0x94 (BRAM Clk22 Delay) for selecting the Config_Delay_32 delay values  
  • Moved the BUFG for clk132 so it is used after the shift register in the Sync component instead of before  
  • Moved the BUFG for str22 to a location after the clk22 configurable delay before it enters the BRAM  
  • Added the ODLD functionality  
  • Added register 0x88 (KS Test Mode) to choose between the two different Kitchen Sink test modes (alternating ones and zeros, ODLD)  
    • Note: Shift_In_Bit (register 0x7C) must be disabled before the KS firmware will enter any test mode  
  • Added a Config_Delay_32 component to create the ODLD clk22 signal  
  • Added register 0x98 (ODLD Clk22 Delay) to select the delay for the ODLD clk22 signal  
  • Added registers 0x9C and 0xA0 (ODLD Memory Address)  
  • Added register 0xA4 (ODLD Buffer Byte)  
  • Added register 0xA8 (Write Byte to ODLD Buffer)  
  • Added the ODLD component  
  • Added MUXes to choose between the different test mode outputs  
  • Added register 0xAC (Clock Delay Register Lock)  
  • Use with v. 0x04 of 6-Bin Data FPGA (production board) |
| Version 0x21 |  |
| • Modified the logic for the wrd0 signal in the Sync component so that bc_event is required to be high for wrd0 to be high - this is to eliminate wrd0 pulses during abort gaps  
  • Use with v. 0x04 of 6-Bin Data FPGA (production board) |
<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 0x00</td>
<td>• First version to work</td>
</tr>
</tbody>
</table>
| Version 0x01 | • Modified Flash Interface so the state machine starts in PROG_DONE rather than DO NOTHING, which configures the FPGAs on power-up  
• Modified Flash Interface so that the FPGA program signal is only asserted on PROG_COMPLETE_1 (and no longer on PROG_COMPLETE_2 as well)  
• Added the max_addr signal that makes the Flash Interface state machine break out of the configuration sequence once the address counter has counted too high (prevents board from locking up)  
• Modified register 0xE8 (VME) to include the FPGA configuration status  
• Added logic to reset the Reset Counter and hence the chip logic using signal reset_reset when a write is performed to register 0xE4 (VME)  
• Added a CPLD firmware version number register (VME 0xE0) |
| Version 0x02 | • Manually encoded FLASH_STATE_TYPE in the Flash Interface so that only 1 bit changes between WAIT_FOR_CMD and the 4 states to which WAIT_FOR_CMD leads  
• Manually encoded COMMAND_TYPE in the Flash Interface (the same gray-count sequence that is automatically generated by Leonardo is used)  
• Added a register for flash_io(6) (internal signal flash_io_6) to prevent the main Flash Interface state machine from jumping states if flash_io(6) changes near a rising edge of clk264 (this eliminated unintentional erase commands from being issued)  
• Replaced all instances of flash_io(6) with flash_io_6 in Flash_Interface_Proc in the Flash Interface  
• Added Inc_Proc to the Flash Interface to make the int_inc signal synchronous  
• Added Reset_Proc to the Flash Interface to make the int_reset signal synchronous  
• Added Programming_Proc to the Flash Interface to make the programming signal synchronous |
| Version 0x03 | • Modified Reset Counter so reset is only asserted when the count = 2 instead of 0-2 |
| Version 0x10 (Created from Version 0x03) | • Except for most of the VME bus logic, the reset logic, and the core VSPROM logic, this design is completely different from the previous versions  
• The VSPROM component now contains the address counter and a state machine to control FPGA configuration  
• The Flash Interface component now only contains registers for the flash signals, which are accessible through the VME bus (there is no longer a state machine for controlling flash reads, writes, and erases)  
• Registers 0xE8-0xFC were completely redefined |
Output Buffers CPLD (Note: version number not available in registers)

| Version 0x00 | • First version to work |
| Version 0x01 | • Added a 1-buffer delay to the str22 signal to compensate for the removal of U10 |

Register Functionality:

0x50 – Data FPGA Firmware Version Number
- **Functionality:** Read-Only (can write to early versions)
- **Description:** The value stored in this register indicates the current version of the Data FPGA firmware.

0x54 – Kitchen Sink FPGA Firmware Version Number
- **Functionality:** Read-Only (can write to early versions)
- **Description:** The value stored in this register indicates the current version of the Kitchen Sink FPGA firmware.

0xE0 – FPGA_Prog CPLD Firmware Version Number
- **Functionality:** Read-Only (use was forbidden in earlier versions)
- **Description:** The value stored in this register indicates the current version of the FPGA_Prog CPLD firmware.

### 3.4 Not-Sure Window Width Registers (0x58 – 0x68)

These registers are used by the Kitchen Sink logic to determine the width of the not-sure windows. Similar to delay line chips, a value of 0x00 represents the step-zero, or minimum, width value. Only the values listed in the **Valid Value Range** field should be used. Figure 4 shows which register affects each window edge, and Table 3 lists the different window widths, obtained from simulation, for each valid register value.

*Note: these registers are undefined in Kitchen Sink firmware version 0.*

![Window Diagram](image)

Figure 4: This figure shows which registers affect each not-sure window edge in the 6-bin design. The VME address for each register is given in hex next to the signal name.
<table>
<thead>
<tr>
<th>Register Value</th>
<th>NS Win 1 Width (ns)</th>
<th>NS Win 2 Width (ns)</th>
<th>NS Win 3 Width (ns)</th>
<th>NS Win 4 Width (ns)</th>
<th>NS Win 5 Width (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>12.3</td>
<td>11.5</td>
<td>13.0</td>
<td>12.5</td>
<td>10.6</td>
</tr>
<tr>
<td>0x01</td>
<td>13.5</td>
<td>12.9</td>
<td>14.5</td>
<td>14.0</td>
<td>11.9</td>
</tr>
<tr>
<td>0x02</td>
<td>14.5</td>
<td>13.8</td>
<td>15.2</td>
<td>14.6</td>
<td>12.8</td>
</tr>
<tr>
<td>0x03</td>
<td>15.4</td>
<td>14.9</td>
<td>16.4</td>
<td>15.8</td>
<td>14.1</td>
</tr>
<tr>
<td>0x04</td>
<td>16.5</td>
<td>16.0</td>
<td>16.9</td>
<td>16.8</td>
<td>14.6</td>
</tr>
<tr>
<td>0x05</td>
<td>18.2</td>
<td>16.5</td>
<td>18.7</td>
<td>18.2</td>
<td>15.6</td>
</tr>
<tr>
<td>0x06</td>
<td>18.8</td>
<td>17.9</td>
<td>19.6</td>
<td>19.0</td>
<td>16.8</td>
</tr>
<tr>
<td>0x07</td>
<td>20.1</td>
<td>19.4</td>
<td>21.0</td>
<td>20.0</td>
<td>18.1</td>
</tr>
<tr>
<td>0x08</td>
<td>20.8</td>
<td>19.8</td>
<td>22.0</td>
<td>20.4</td>
<td>19.5</td>
</tr>
<tr>
<td>0x09</td>
<td>22.4</td>
<td>20.9</td>
<td>23.5</td>
<td>22.0</td>
<td>20.9</td>
</tr>
<tr>
<td>0x0A</td>
<td>23.2</td>
<td>21.7</td>
<td>24.3</td>
<td>22.9</td>
<td>21.8</td>
</tr>
<tr>
<td>0x0B</td>
<td>24.0</td>
<td>23.1</td>
<td>25.4</td>
<td>23.8</td>
<td>23.2</td>
</tr>
<tr>
<td>0x0C</td>
<td>24.7</td>
<td>24.0</td>
<td>25.9</td>
<td>24.4</td>
<td>23.7</td>
</tr>
<tr>
<td>0x0D</td>
<td>26.8</td>
<td>25.2</td>
<td>28.0</td>
<td>26.5</td>
<td>25.7</td>
</tr>
<tr>
<td>0x0E</td>
<td>27.4</td>
<td>25.5</td>
<td>28.7</td>
<td>27.1</td>
<td>26.4</td>
</tr>
<tr>
<td>0x0F</td>
<td>28.4</td>
<td>27.4</td>
<td>30.2</td>
<td>28.7</td>
<td>27.6</td>
</tr>
</tbody>
</table>

Table 3: This table lists the not-sure window widths for valid values of the not-sure window width registers. The times were taken from post place-and-route simulations done on the Kitchen Sink FPGA version 1 firmware.

Register Functionality:

0x58 – Not-Sure Window 1 Width

- **Valid Value Range:** 0x00 – 0x0F (0 – 15)
- **Functionality:** Read/Write
- **Description:** This register value determines the number of extra buffers used in delaying the leading edge of window 1, which effectively determines the width of not-sure window 1.

0x5C – Not-Sure Window 2 Width

- **Valid Value Range:** 0x00 – 0x0F (0 – 15)
- **Functionality:** Read/Write
- **Description:** This register value determines the number of extra buffers used in delaying the leading edge of window 2, which effectively determines the width of not-sure window 2.

0x60 – Not-Sure Window 3 Width

- **Valid Value Range:** 0x00 – 0x0F (0 – 15)
- **Functionality:** Read/Write
- **Description:** This register value determines the number of extra buffers used in delaying the leading edge of window 3, which effectively determines the width of not-sure window 3.
0x64 – Not-Sure Window 4 Width
  
  **Valid Value Range:** 0x00 – 0x0F (0 – 15)
  
  **Functionality:** Read/Write
  
  **Description:** This register value determines the number of extra buffers used in delaying the leading edge of window 4, which effectively determines the width of not-sure window 4.

0x68 – Not-Sure Window 5 Width
  
  **Valid Value Range:** 0x00 – 0x0F (0 – 15)
  
  **Functionality:** Read/Write
  
  **Description:** This register value determines the number of extra buffers used in delaying the leading edge of window 5, which effectively determines the width of not-sure window 5.

### 3.5 Miscellaneous Registers (0x0C, 0x6C – 0x88)

These registers perform various functions, including providing access to the board ID, the board serial number, the number of time bins in the design, enabling/disabling shift_in_bit and the not-sure windows, reprogramming the Transition board FPGA, and setting the Kitchen Sink test mode.

*Note: some of these registers (namely 0x0C, 0x6C, 0x70, 0x7C, 0x84, and 0x88) are not implemented in the prototype designs. While they are free read/write registers, to maintain as much compatibility between the prototype and production boards as possible (except register 0x0C), they should not be used for other functions.*

**Register Functionality:**

- **0x0C – Board ID (Production Boards Only)**
  
  **Functionality:** Read-Only
  
  **Description:** When read, the contents of this register alternate between 0xF1 and 0x66 for the 6-bin design.

- **0x6C – Board Serial Number (Production Boards Only)**
  
  **Functionality:** Read-Only
  
  **Description:** The low 8 bits of the 9-bit board serial number are stored in this register.

- **0x70 – Board Serial Number (Production Boards Only)**
  
  **Functionality:** Read-Only
  
  **Description:** The most significant bit of the 9-bit board serial number is stored in the least significant bit of this register. The other 7 bits of this register are set to 0.
0x74 – Number of Time Bins

*Valid Value Range:* Only 0x02 and 0x06 (0 and 6)

*Functionality:* Read-Only 0x02 and 0x06

*Description:* This register is set to either 2 or 6, indicating the number of time bins in the design.

0x78 – Transition Board FPGA Reprogram

*Valid Value Range:* 0x00 – 0xFF (0 – 255)

*Functionality:* Read/Write

*Description:* This register is used to reprogram the FPGA on the Transition Board. Write 0xFF (or any value greater than 0x00) followed by 0x00 to this register to begin the FPGA reconfiguration process. The register must be equal to 0xFF (or any value greater than 0x00) for at least 300ns. Since VME writes take significantly more time than this, 0x00 may be written immediately after 0xFF.

0x7C – Shift In Bit Enable/Test Mode Disable

*Valid Value Range:* 0x00 – 0xFF (0 – 255)

*Functionality:* Read/Write

*Description:* This register is used to enable (any value but 0x00) and disable (0x00) the shift_in_bit signal that pulses once out of every 16 shift_clk (22-ns period) cycles. The shift_in_bit signal is used by the Data FPGA firmware in conjunction with shift_clk to determine when each set of multiplexed output data is sent. Disabling the shift_in_bit signal results in stopping all data transmission from the Data FPGA. In addition, the Data FPGA contains a counter that sets all data outputs to high impedance if a shift_in_bit pulse has not been observed for 64 shift_clk cycles. This is done to allow the Kitchen Sink FPGA to enter a test mode safely, without bus contention, in which it transmits alternating ones and zeros on the data lines that are inverted every wrd0 period (396 ns). This means that “101010…1010” is transmitted first, followed by “010101…0101.” So, to summarize all of this: writing 0x00 to this register will cause the XTC 2 to stop normal data transmission and enter the test mode; writing any other value to this register will cause the XTC 2 to exit the test mode and process/transmit data as normal. The default value for this register on power-up is 0xFF (normal operation).
0x80 – Not-Sure Window Enable (6-Bin Designs Only)

*Valid Value Range:* 0x00 – 0xFF (0 – 255)
*Functionality:* Read/Write
*Description:* This register is used to enable (any value but 0x00) and disable (0x00) the not-sure window logic in the 6-bin designs. It does not exist in the 2-bin designs. Note that disabling the not-sure windows does not actually change the detection mechanism. 11 flip-flops are still used—one for each window. This register controls the logic after detection: if disabled, any hits in the not-sure part of each time bin are counted as hits in that bin, regardless of whether hits were registered in the previous time bin.

0x84 – Transition Board Mode (6-Bin Designs Only)

*Valid Value Range:* 0x00 – 0x03 (0 – 3)
*Functionality:* Read/Write
*Description:* This register is used to set the Transition Board operating mode. The values of the two least-significant bits of this register are placed on the two reserved lines that go from the XTC 2 to the Transition Board (PCB signals RESERVED_1 and RESERVED_0).

0x88 – Kitchen Sink Test Mode (6-Bin Designs Only)

*Valid Value Range:* 0x00 – 0xFF (0 – 255)
*Functionality:* Read/Write
*Description:* This register is used to choose between two different Kitchen Sink test modes: storing 0x00 will cause the Kitchen Sink FPGA to transmit alternating ones and zeros (see description of register 0x7C); storing any other value will cause the Kitchen Sink FPGA to transmit ODLD signals. **Note:** Shift In Bit must be disabled (write 0x00 to register 0x7C) before any test signals will be transmitted by the Kitchen Sink FPGA.

### 3.6 Clock Delay Registers (0x8C – 0x9C)

These five registers (one of which functions as a value lock for the other four) are used to control the delay of four clock signals in the Kitchen Sink design. These delays ensure proper data alignment in the L2 buffers and proper synchronization of the control and data bits sent out by the XTC2 board. Provided that the XTC2 is functioning properly, it is not recommended that the default values for these registers be modified. As a result, one of these registers is used as a lock for the other four to reduce the chances of an accidental delay change.
Register Functionality:
0x8C – Clock Delay Register Lock (6-Bin Designs Only)

- **Valid Value Range:** 0x00 – 0xFF (0 – 255)
- **Functionality:** Read/Write

*Description:* This register is used to lock registers 0x90-0x9C. Those registers can only be written to while the value of this register is set to 0xFE.

0x90 – BRAM Clk132 Delay (6-Bin Designs Only)

- **Valid Value Range:** 0x00 – 0x1F (0 – 31)
- **Functionality:** Read/Write

*Description:* This register is used to adjust the delay of the clk132 signal used by the BRAM component (where the L2 buffers are implemented). An improper delay value would likely result in shifting all data in the L2 buffers by 2 or 4 entire windows. The delay steps are roughly 1-2ns each. **Note:** This register must be unlocked (write 0xFE to register 0x8C) before data can be written to it.

0x94 – BRAM Clk22 Delay (6-Bin Designs Only)

- **Valid Value Range:** 0x00 – 0x1F (0 – 31)
- **Functionality:** Read/Write

*Description:* This register is used to adjust the delay of the clk22 signal used by the BRAM component (where the L2 buffers are implemented). An improper delay value would likely result in data storage errors or shifting of data within and between windows. The delay steps are roughly 1-2ns each. **Note:** This register must be unlocked (write 0xFE to register 0x8C) before data can be written to it.

0x98 – ODLD Clk22 Delay (6-Bin Designs Only)

- **Valid Value Range:** 0x00 – 0x1F (0 – 31)
- **Functionality:** Read/Write

*Description:* This register is used to adjust the delay of the clk22 signal used by the ODLD component. An improper delay value would likely result in data shifting or data errors. The delay steps are roughly 1-2ns each. **Note:** This register must be unlocked (write 0xFE to register 0x8C) before data can be written to it.
0x9C – Clk132 Shift/Shift_In_Bit Clock Count (6-Bin Designs Only)

**Valid Value Range:** 0x00 – 0x0E (0 – 14)

**Functionality:** Read/Write

**Description:** This register is used to shift the clk132 signal used by the CDFL1A Logic component (where the L1 accept logic and L2 buffers are implemented). This register value is also used to adjust when the B0, Word0, and Str (data strobe) signals are transmitted. An improper delay value would likely result in shifting all data in the L2 buffers by entire windows and/or misalignment of the B0, Word0, and Str signals with the data sent by the Data FPGA. The delay steps are 22-ns clock periods.

**Note:** This register must be unlocked (write 0xFE to register 0x8C) before data can be written to it.

### 3.7 ODLD Registers (0xA0 – 0xAC)

The Output Data Looping logic is intended to exercise the output path of the XTC 2 card, as well as provide an easy means of sending desired data patterns through the Transition cards to the Finder for testing and debugging purposes. This design, once a separate set of firmware programs, is now embedded in the 6-Bin Kitchen Sink firmware. To use this logic, the KS Test Mode register must be set to ODLD (write any non-zero value to register 0x88) and the Shift_In_Bit signal must be disabled (write 0x00 to register 0x7C).

The Output Data Looping design contains a buffer that is 216 words deep by 32 bits wide. This provides enough space to store unique data for each 22-ns output clock cycle for a set of 12 beam crossings. Data is loaded into the buffer one byte at a time using the VME bus interface. An address counter starts at 0 on a bunch 0 crossing and increments every 22 ns except during the abort gaps, and it resets after each abort gap. All necessary output synchronization signals—STROBE, WORD0, and output B0—are generated by the design and synchronized to the input B0 signal (there is some time, of course, between the input B0 pulse and the observance of the output B0 pulse).

Looping through the data in the memory buffer begins immediately after FPGA configuration, and is synchronized with the input B0 signal. The user only needs to enable the ODLD test mode, disable Shift_In_Bit, and load the desired data into the memory buffer.

The procedure for loading data into the buffer is as follows:

1. Write the 10 bits of the address to registers 0xA0 and 0xA4.
2. Write the data byte to register 0xAC.
3. Repeat steps 1 and 2 until all desired data has been written. The default value for all data bytes is 0x00.
The procedure for reading back the data from the buffer for verification is as follows:
1. Write the 10 bits of the address to registers 0xA0 and 0xA4.
2. Read the data byte from register 0xA8.
3. Compare the value read back with what was written.
4. Repeat steps 1-3 until all desired data has been verified.

Figure 5 shows a diagram of the memory buffer, along with the addressing scheme used. Bytes may be written in any order. Words 0-215 (each 32 bits in length) are transmitted in succession every 22 ns, and represent an entire train of 12 bunch crossings. These data are repeated for each train—three times for each B0. Data from word 215 is held during each abort gap.

<table>
<thead>
<tr>
<th>Bits</th>
<th>7:0</th>
<th>15:8</th>
<th>23:16</th>
<th>31:24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
</tbody>
</table>

Figure 5: This figure shows the structure of the memory buffer, the corresponding addressing scheme, and how all 216 words are repeated three times for each B0. Each word is a set of 32 bits transmitted during a 22-ns output clock cycle. The numbers in the boxes are the byte addresses written to registers 0xA0 and 0xA4. For example, to write a byte to bits 15:8 of word 215, address 861 must be stored in the address registers. Since 861_{10} = 00000011 01011101_{2}, 00000011_{2} would be written to register 0xA4, and 01011101_{2} would be written to register 0xA0.

Register Functionality:
0xA0 – ODLD Memory Address (6-Bin Designs Only)
Valid Value Range: 0x00 – 0xFF (0 – 255)
Functionality: Read/Write
Description: This register contains the low 8 bits (7:0) of the memory buffer address. Changing the value of this register will result in an automatic read of the new address, the data from which can be read from register 0xA8.

0xA4 – ODLD Memory Address (6-Bin Designs Only)
Valid Value Range: 0x00 – 0x03 (0 – 3)
Functionality: Read/Write
Description: This register contains the high 2 bits (9:8) of the memory buffer address. These bits are stored in the two least-significant bits of the register (1:0). Any values written to the upper 6 bits of this register will be ignored. Changing the value of this register will result in an automatic read of the new address, the data from which can be read from register 0xA8.
0xA8 – ODLD Buffer Byte (6-Bin Designs Only)

Valid Value Range: N/A

Functionality: Read-Only

Description: When an address value is written to registers 0xA0 and 0xA4, the byte is read from the memory buffer and stored in this register.

0xAC – Write Byte to ODLD Buffer (6-Bin Designs Only)

Valid Value Range: 0x00 – 0xFF (0 – 255)

Functionality: Write-Only

Description: The byte written to this register will be written to the memory buffer at the address specified by registers 0xA0 and 0xA4.

3.8 Free Registers (0xB0 – 0xDC)

These read/write registers are currently undefined and may be used for additional functionality.

3.9 Flash Memory Registers (0xE4 – 0xFC)

These registers, implemented in the FPGA-Programmer CPLD, are used to program the flash memory with FPGA configuration files. The address used for read and write operations used to be supplied by a counter in the CPLD to which the user did not have direct access. Now, however, the user has access to all flash data, address, and control signals directly using the VME bus.

The procedure for programming the flash is as follows:

1. Give the flash the erase command (see “Erase Procedure” below).
2. Wait until erasure is complete (see “Wait for Flash Ready Procedure” below).
3. Do the following for all data bytes, one at a time: give the flash the write command (see “Write Procedure” below), and then wait until it is done writing (see “Wait for Flash Ready Procedure” below).
4. Verify successful programming by doing the following for each data byte: read the byte back from memory (see “Read Procedure” below), and then compare the read back value with the byte that should have been written.
5. If all bytes were successfully written, end the programming procedure by writing any data value to register 0xE4. This resets the CPLD and begins configuration of the FPGAs.
6. Wait until FPGA configuration is complete (see “Wait for FPGA Configuration Procedure” below).

Erase Procedure:

1. Write 0x21 to VME address 0xEC (set flash address to 0x5555)
2. Write 0x42 to VME address 0xEC (set flash control lines to write)
3. Write 0xAA to VME address 0xF0 (set flash data to 0xAA)
4. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
5. Write 0x20 to VME address 0xEC (set flash address to 0x2AAA)
6. Write 0x42 to VME address 0xEC (set flash control lines to write)
7. Write 0x55 to VME address 0xF0 (set flash data to 0x55)
8. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
9. Write 0x21 to VME address 0xEC (set flash address to 0x5555)
10. Write 0x42 to VME address 0xEC (set flash control lines to write)
11. Write 0x80 to VME address 0xF0 (set flash data to 0x80)
12. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
13. Write 0x21 to VME address 0xEC (set flash address to 0x5555)
14. Write 0x42 to VME address 0xEC (set flash control lines to write)
15. Write 0xAA to VME address 0xF0 (set flash data to 0xAA)
16. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
17. Write 0x20 to VME address 0xEC (set flash address to 0x2AAA)
18. Write 0x42 to VME address 0xEC (set flash control lines to write)
19. Write 0x55 to VME address 0xF0 (set flash data to 0x55)
20. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
21. Write 0x21 to VME address 0xEC (set flash address to 0x5555)
22. Write 0x42 to VME address 0xEC (set flash control lines to write)
23. Write 0x10 to VME address 0xF0 (set flash data to 0x10)
24. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)

Write Procedure:
1. Write 0x21 to VME address 0xEC (set flash address to 0x5555)
2. Write 0x42 to VME address 0xEC (set flash control lines to write)
3. Write 0xAA to VME address 0xF0 (set flash data to 0xAA)
4. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
5. Write 0x20 to VME address 0xEC (set flash address to 0x2AAA)
6. Write 0x42 to VME address 0xEC (set flash control lines to write)
7. Write 0x55 to VME address 0xF0 (set flash data to 0x55)
8. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
9. Write 0x21 to VME address 0xEC (set flash address to 0x5555)
10. Write 0x42 to VME address 0xEC (set flash control lines to write)
11. Write 0xA0 to VME address 0xF0 (set flash data to 0xA0)
12. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
13. Write bits 18:16 of flash address to bits 2:0 of VME address 0xFC
14. Write bits 15:8 of flash address to bits 7:0 of VME address 0xF8
15. Write bits 7:0 of flash address to bits 7:0 of VME address 0xF4
16. Write 0x42 to VME address 0xEC (set flash control lines to write)
17. Write flash data to VME address 0xF0
18. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)
Read Procedure:
1. Write bits 18:16 of flash address to bits 2:0 of VME address 0xFC
2. Write bits 15:8 of flash address to bits 7:0 of VME address 0xF8
3. Write bits 7:0 of flash address to bits 7:0 of VME address 0xF4
4. Write 0x41 to VME address 0xEC (set flash control lines to read)
5. Read flash data from VME address 0xF0
6. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)

FPGA Configuration Procedure:
1. Reset the CPLD by writing any data value to register 0xE4.
2. Wait until FPGA configuration is complete (see “Wait for FPGA Configuration Procedure” below).

Wait for Flash Ready Procedure:
1. Choose any flash address
2. Write bits 18:16 of flash address to bits 2:0 of VME address 0xFC
3. Write bits 15:8 of flash address to bits 7:0 of VME address 0xF8
4. Write bits 7:0 of flash address to bits 7:0 of VME address 0xF4
5. Repeat the following until either the second most-significant bit of the flash data byte read stops toggling on successive reads or a timeout threshold is reached:
   a. Write 0x41 to VME address 0xEC (set flash control lines to read)
   b. Read flash data from VME address 0xF0
   c. Write 0x47 to VME address 0xEC (set flash control lines to stand-by)

Wait for FPGA Configuration Procedure:
1. Keep reading VME address 0xE8 until bit 3 (in 7:0) is equal to 0 (programming = 0)
2. Read VME address 0xE8 again. If bits 0 and 1 (in 7:0) are both equal to 1, then configuration was successful (fpga_done = 1 and fpga_init_n = 1). Otherwise, configuration failed.

Register Functionality:
0xE4 – FPGA_Prog CPLD Reset
Valid Value Range: 0x00 – 0xFF (0 – 255)
Functionality: Write-Only
Description: Writing to this register (the data value does not matter) resets the FPGA-Programmer CPLD logic.
**0xE8 – CPLD/Flash Status**

*Valid Value Range:* N/A
*Functionality:* Read-Only
*Description:* This register can be read at any time to determine the current logic value of eight different signals:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Signal</th>
<th>Meaning of Logic 0</th>
<th>Meaning of Logic 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FPGA_DONE</td>
<td>FPGAs not configured</td>
<td>FPGAs configured</td>
</tr>
<tr>
<td>1</td>
<td>FPGA_INIT_N</td>
<td>FPGAs initializing</td>
<td>FPGAs not initializing</td>
</tr>
<tr>
<td>2</td>
<td>FPGA_PROG_N</td>
<td>Start FPGA configuration</td>
<td>Don’t start FPGA configuration</td>
</tr>
<tr>
<td>3</td>
<td>PROGRAMMING</td>
<td>CPLD not configuring FPGAs</td>
<td>CPLD configuring FPGAs</td>
</tr>
<tr>
<td>4</td>
<td>MAX_ADDR</td>
<td>Config address counter &lt; 0x60000</td>
<td>Config address counter = 0x60000</td>
</tr>
<tr>
<td>5</td>
<td>FLASH_CE_N</td>
<td>Flash enabled</td>
<td>Flash disabled</td>
</tr>
<tr>
<td>6</td>
<td>FLASH_OE_N</td>
<td>Flash output enabled</td>
<td>Flash output disabled</td>
</tr>
<tr>
<td>7</td>
<td>FLASH_WE_N</td>
<td>Flash write enabled</td>
<td>Flash write disabled</td>
</tr>
</tbody>
</table>

**0xEC – Command Register**

*Valid Value Range:* 0x20, 0x21, 0x41, 0x42, 0x47 (32, 33, 65, 66, 71)
*Functionality:* Write-Only
*Description:* Writing to this register executes the command specified by the data value written. The following are the valid commands:

<table>
<thead>
<tr>
<th>Data Value</th>
<th>Command/Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20</td>
<td>Set flash address to 0x2AAA</td>
</tr>
<tr>
<td>0x21</td>
<td>Set flash address to 0x5555</td>
</tr>
<tr>
<td>0x41</td>
<td>Set flash control to read (ce_n = 0, oe_n = 0, we_n = 1)</td>
</tr>
<tr>
<td>0x42</td>
<td>Set flash control to write (ce_n = 0, oe_n = 1, we_n = 0)</td>
</tr>
<tr>
<td>0x47</td>
<td>Set flash control to stand-by (ce_n = 1, oe_n = 1, we_n = 1)</td>
</tr>
</tbody>
</table>

**0xF0 – Flash Data Byte**

*Valid Value Range:* 0x00 – 0xFF (0 – 255)
*Functionality:* Read/Write
*Description:* Writing to this register places the value written on the eight flash data lines. Reading from this register gives the value on the flash data lines.

**0xF4 – Addr0 (Address_0)**

*Valid Value Range:* 0x00 – 0xFF (0 – 255)
*Functionality:* Read/Write
*Description:* Bits 7:0 of this register correspond to bits 7:0 of the flash address.

**0xF8 – Addr1 (Address_1)**

*Valid Value Range:* 0x00 – 0xFF (0 – 255)
*Functionality:* Write-Only
*Description:* Bits 7:0 of this register correspond to bits 15:8 of the flash address.
0xFC – Addr2 (Address_2)

Valid Value Range: 0x00 – 0x07 (0 – 7)
Functionality: Read/Write

Description: Bits 2:0 of this register correspond to bits 18:16 of the flash address. The upper five bits are not used and will return 0 when read.
## Revisions

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/27/2004</td>
<td>1.0</td>
<td>-Original document</td>
</tr>
<tr>
<td>11/5/2004</td>
<td>1.1</td>
<td>-Updated section 3.3: added more design number definitions</td>
</tr>
</tbody>
</table>
| 3/16/2005  | 1.2     | -Updated entire document: replaced all instances of “FPGA-Programming CPLD” with “FPGA-Programmer CPLD”  
-Updated Table 1: added registers; expanded the “First Version” column; added the “Initial Value” columns; changed the buffer status registers to read-only  
-Added Table 2 for the production design  
-Updated section 3: the KS FPGA only reprograms the delay line chips and other registers when certain registers are written to; added note on firmware versions for which the register definitions are valid  
-Updated section 3.1: added reference to the initial register values in Tables 1 and 2; expanded “step size” definitions  
-Updated section 3.2: modified registers 0x40-0x4C; added note about the first byte added to the beginning of each level 2 buffer; updated the valid value range of register 0x38  
-Updated section 3.3: added register 0xE0; updated and added more design number definitions  
-Updated section 3.4: renumbered Table 2 to Table 3; marked the values in Table 3 as old  
-Updated register ranges in the title of sections 3.5 and 3.6  
-Updated section 3.6: removed register 0xE0 from this section; modified registers 0xE4 and 0xE8; modified the flash programming procedure to account for the update to 0xE8  
-Renumbered sections 3.5 and 3.6 to 3.6 and 3.7  
-Added section 3.5: Miscellaneous Registers                                                                                                                                 |
| 4/7/2005   | 1.3     | -Updated Table 1: changed register 0xE0  
-Updated Table 2: changed registers 0x04, 0x10, 0x1C, 0x20, 0x3C, 0x7C, and 0xE0; changed the ** note for Table 2  
-Updated Section 3: updated the list of registers for which the KS FPGA reprograms the delay line chips; updated firmware version numbers  
-Updated Section 3.1: updated the “Standard Production” delay line chip step values  
-Updated Section 3.2: modified the buffer read procedure listing; updated the descriptions of registers 0x38 and 0x3C  
-Updated Section 3.3: updated and added more design number definitions  
-Updated Section 3.5: updated the register range in the title; added register 0x7C  
-Updated Section 3.6: updated the register range in the title |
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 5/9/2005 | 1.4     | - Updated Table 2: grayed out register 0x0C because chip U10 was removed from all production boards  
- Updated Section 3: updated firmware version numbers  
- Updated Section 3.1: added note about the removal of U10; updated the description of register 0x0C  
- Updated Section 3.3: added more design number definitions; added a table for the Output Buffers CPLD versions |
| 6/3/2005 | 1.5     | - Updated Section 3: updated firmware version numbers  
- Updated Section 3.3: added more design number definitions |
| 6/24/05  | 1.6     | - Updated Table 2: redefined VME register 0x40 – 0x4C  
- Updated Section 3: updated firmware version numbers  
- Updated Section 3.2: added a note about the change of the Buffer Status Registers to the Buffer Content Registers; removed the Buffer Status Register reads from the buffer read procedure; updated the register definitions of register 0x40 – 0x4C  
- Updated Section 3.3: added more design number definitions |
| 9/9/05   | 1.7     | - Updated Table 2: changed the 6-bin initial values  
- Updated Section 3: updated firmware version numbers  
- Updated Section 3.3: added more design number definitions |
| 9/20/05  | 1.8     | - Updated Table 1: changed the R/W field for register 0x78 to R/W  
- Updated Table 2: added register 0x84; updated the unused register range; changed the R/W field for register 0x78 to R/W  
- Updated Section 3: updated the list of registers for which the KS FPGA reprograms the delay line chips; updated firmware version numbers  
- Updated Section 3.3: added more design number definitions  
- Updated Section 3.5: changed the Functionality of register 0x78 to “Read/Write”; added register 0x84 to the list of registers only found in the 6-bin design (see the Note in that section); added a register definition for register 0x84  
- Updated Section 3.6: updated the register range in the title |
| 11/29/05 | 1.9     | - Updated Table 1: completely redefined registers 0xE8-0xFC  
- Updated Table 2: completely redefined registers 0xE8-0xFC  
- Updated Section 3: updated firmware version numbers  
- Updated Section 3.2: modified the description of register 0x38 to include the revised automatic counting sequence; updated the buffer read procedure in the beginning of this section to reflect this sequence change  
- Updated Section 3.3: added more design number definitions  
- Updated Section 3.7: modified the section description; updated the procedures for programming the flash; updated registers 0xE8-0xFC (completely different functionality for all of them) |
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 12/2/05   | 1.10    | -Updated Table 2: redefined register 0x0C  
-Updated Section 3: updated firmware version numbers  
-Updated Section 3.1: noted that the register definition of 0x0C in this section only applies to prototype boards; updated the text to reflect this change  
-Updated Section 3.3: added more design number definitions  
-Updated Section 3.5: updated the register range in the title; added register 0x0C to this section; updated the text to reflect this addition |
| 2/22/06   | 1.11    | -Updated Table 2: added registers 0x88 – 0xAC  
-Updated Section 3: updated the list of registers for which the KS FPGA reprograms the delay line chips; updated firmware version numbers  
-Updated Section 3.3: added more design number definitions  
-Updated Section 3.5: updated the register range in the title; added register 0x88 to this section; updated the text to reflect this addition  
-Renumbered Sections 3.6 and 3.7 to 3.8 and 3.9  
-Added Section 3.6: Clock Delay Registers  
-Added Section 3.7: ODLD Registers  
-Updated Section 3.9: updated the register range in the title’ |
| 3/13/06   | 1.12    | -Updated Section 3: updated firmware version numbers  
-Updated Section 3.3: added more design number definitions |