MODEL 2366

UNIVERSAL PROGRAMMABLE
LOGIC MODULE

CE
Since this product is a subassembly, it is the responsibility of the end user, acting as the system integrator, to ensure that the overall system is CE compliant. This product was demonstrated to meet CE conformity using a CE compliant crate housed in an EMI/RFI shielded enclosure. It is strongly recommended that the system integrator establish these same conditions.
# TABLE OF CONTENTS

1. General Information
   - Purpose 7
   - Unpacking & Inspection 7
   - Warranty 7
   - Product Assistance 7
   - Maintenance Agreements 7
   - Documentation Discrepancies 8
   - Software Licensing Agreement 8
   - Service Procedure 8

2. Operating Instruction
   - General 9
   - Specifications 10
   - Front Panel 12

3. Applications
   - How to Program the Model 2366 16
     - Software
       - Load 2366 18
       - LDGP 2366 20
       - EPRM 2366 22
   - 59-Bit Input Module 24
   - 59-bit Output Register Module 30
   - CAMAC Dataway Test Module 38
   - 48-Input Majority Logic Unit 44
   - Long Range Multihit Time Digitizer 52
IMPORTANT NOTICE: THE 2366 ULM IS SHIPPED WITH BOTH INPUT AND OUTPUT LEVEL CONVERTERS INSTALLED. NO DAMAGE WILL OCCUR IF THE UNIT IS POWERED WITH THE STANDARD EPROM PROGRAM, WHICH DOES NOT USE THE FRONT PANEL I/O. HOWEVER, BEFORE ANY OTHER USER PROGRAM IS LOADED, THE UNIT MUST BE CONFIGURED WITH THE DESIRED I/O PATTERN. FOR EACH PAIR OF SOCKETS, FOR THE 10124 AND 10125, ONLY ONE SHOULD BE OCCUPIED. IN TOTAL, 17 ICs SHOULD BE REMOVED.
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PURPOSE</td>
<td>This manual is intended to provide instruction regarding the setup and operation of the covered instruments. In addition, it describes the theory of operation and presents other information regarding its functioning and application.</td>
</tr>
<tr>
<td>UNPACKING AND INSPECTION</td>
<td>It is recommended that the shipment be thoroughly inspected immediately upon delivery. All material in the container should be checked against the enclosed Packing List and shortages reported promptly. If the shipment is damaged in any way, please notify the Customer Service Department or the local field service office. If the damage is due to mishandling during shipment, you may be requested to assist in contacting the carrier in filing a damage claim.</td>
</tr>
<tr>
<td>WARRANTY</td>
<td>LeCroy warrants its instrument products to operate within specifications under normal use and service for a period of one year from the date of shipment. Component products, replacement parts, and repairs are warranted for 90 days. This warranty extends only to the original purchaser. Software is thoroughly tested, but is supplied &quot;as is&quot; with no warranty of any kind covering detailed performance. Accessory products not manufactured by LeCroy are covered by the original equipment manufacturers' warranty only. In exercising this warranty, LeCroy will repair or, at its option, replace any product returned to the Customer Service Department or an authorized service facility within the warranty period, provided that the warrantor's examination discloses that the product is defective due to workmanship or materials and has not been caused by misuse, neglect, accident or abnormal conditions or operations. The purchaser is responsible for the transportation and insurance charges arising from the return of products to the servicing facility. LeCroy will return all in-warranty products with transportation prepaid. This warranty is in lieu of all other warranties, express or implied, including but not limited to any implied warranty of merchantability, fitness, or adequacy for any particular purpose or use. LeCroy shall not be liable for any special, incidental, or consequential damages, whether in contract, or otherwise.</td>
</tr>
<tr>
<td>PRODUCT ASSISTANCE</td>
<td>Answers to questions concerning installation, calibration, and use of LeCroy equipment are available from the Customer Service Department, 700 Chestnut Ridge Road, Chestnut Ridge, New York, 10977-6499, (914) 578-6030.</td>
</tr>
<tr>
<td>MAINTENANCE AGREEMENTS</td>
<td>LeCroy offers a selection of customer support services. For example, Maintenance Agreements provide extended warranty that allows the customer to budget maintenance costs after the initial warranty has expired. Other services such as installation, training, on-site repair, and addition of engineering improvements are available through specific Supplemental Support Agreements. Please contact the Customer Service Department for more information.</td>
</tr>
</tbody>
</table>
DOCUMENTATION DISCREPANCIES

LeCroy is committed to providing state-of-the-art instrumentation and is continually refining and improving the performance of its products. While physical modifications can be implemented quite rapidly, the corrected documentation frequently requires more time to produce. Consequently, this manual may not agree in every detail with the accompanying product and the schematics in the Service Documentation. There may be small discrepancies in the values of components for the purposes of pulse shape, timing, offset, etc., and, occasionally, minor logic changes. Where any such inconsistencies exist, please be assured that the unit is correct and incorporates the most up-to-date circuitry.

SOFTWARE LICENSING AGREEMENT

Software products are licensed for a single machine. Under this license you may:

- Copy the software for backup or modification purposes in support of your use of the software on a single machine.

- Modify the software and/or merge it into another program for your use on a single machine.

- Transfer the software and the license to another party if the other party accepts the terms of this agreement and you relinquish all copies, whether in printed or machine readable form, including all modified or merged versions.

SERVICE PROCEDURE

Products requiring maintenance should be returned to the Customer Service Department or authorized service facility. If under warranty, LeCroy will repair or replace the product at no charge. The purchaser is only responsible for the transportation charges arising from return of the goods to the service facility. For all LeCroy products in need of repair after the warranty period, the customer must provide a Purchase Order Number before any inoperative equipment can be repaired or replaced. The customer will be billed for the parts and labor for the repair as well as for shipping. All products returned for repair should be identified by the model and serial numbers and include a description of the defect or failure, name and phone number of the user. In the case of products returned, a Return Authorization Number is required and may be obtained by contacting the Customer Service Department at (914) 578-6030.
GENERAL

The CAMAC Model 2366 is a general purpose programmable logic module, using state-of-the-art field programmable gate array technology. This CAMAC module can be used as a programmable LeCroy ECLine trigger processor module, among other uses. In addition to a full 24-bit CAMAC interface, there are 59 front panel differential ECL I/O signals, which (with some restrictions) can be independently selected to be either inputs or outputs. This module is also useful as a general purpose controller, as part of a test system or data acquisition system.

The desired logical operations are programmed in a Xilinx 4005 gate array chip. Any logic that can be implemented as a synchronous (clocked) state machine may be programmed, subject only to the limitations of the Xilinx 4005 gate array chip (approximately 5,000 equivalent gates). There are 3 clocks available on the board, 40, 20, and 10 MHz, or any of 3 special front panel inputs may be used as a clock. Input and output signals use standard 10124 and 10125 TTL-ECL level translators. Input signals as short as 5 nanoseconds can be latched and synchronized with the internal state machine logic.

The gate array must be programmed after power up, and can be reprogrammed at any time. An on-board EPROM socket may contain a program which is loaded on power up, or reloaded on CAMAC command. The Xilinx chip may also be programmed directly from the CAMAC dataway (the program information is stored in RAM in the Xilinx chip, so there is no limit to the number of times that it can be reprogrammed).

A few basic CAMAC function codes are implemented in hardware, and are available on power up. These are only used to program the Xilinx chip, and all but one of these functions disappear after a CAMAC clear operation (F9, C, or Z).

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F30</td>
<td>A0-A15 Enter programming mode with eprom selected, enable all other hardware function codes.</td>
</tr>
<tr>
<td>F28</td>
<td>A0-A15 Select CAMAC programming mode.</td>
</tr>
<tr>
<td>F25</td>
<td>A0-A15 Program Xilinx chip (program pulse lasts until the next S1).</td>
</tr>
<tr>
<td>F16</td>
<td>A0-A15 Write 8 bits to the Xilinx.</td>
</tr>
<tr>
<td>F12</td>
<td>A0-A15 Test Xilinx READY line (not usually required).</td>
</tr>
<tr>
<td>F13</td>
<td>A0-A15 Test Xilinx program DONE.</td>
</tr>
<tr>
<td>F14</td>
<td>A0-A15 Test Xilinx INIT line.</td>
</tr>
<tr>
<td>F9</td>
<td>A0-A15 Disable function codes except for F30 (CAMAC C, Z have the same effect).</td>
</tr>
</tbody>
</table>

The Xilinx chip automatically loads itself if an EPROM is installed on the board. An F30 command followed by an F25 will cause the Xilinx chip to be reset and reloaded from the EPROM.
To load a program from CAMAC, the F30 command is followed by an F28 and F25. After the Xilinx INIT line is true (test with F14), the data is written 8 bits at a time using F16. F12 tests the Xilinx ready line before each write operation. This is not required unless the CAMAC host is capable of CAMAC operations at a rate greater than 500 kHz. This continues until all data is written and the Xilinx DONE line is true (test with F13). The Xilinx XACT software or any of several third party gate array software packages can be used to prepare the program file. An example basic program is supplied which reads the Xilinx .BIT file and uses standard ESONE CAMAC functions to load the program file into the 2366 module.

The initial program in the EPROM is T2366E (the schematic is included in the manual). This implements a simple divide chain to flash the LEDs with the 3 internal clocks (40 MHz, 20 MHz and 10 MHz), a 24 bit read write register and a register which latches the CAMAC F,A,Z,I,N on every S2. This last register is read by F0, A1. This emulates most of the test portion of the LeCroy 2050 CAMAC dataway display and test module.

A simple basic program which exercises the 2366 is EP2366.bas. It starts by forcing the reload of the Xilinx chip from the eprom (with T2366E), and implements a simple CAMAC system test.

The example program LOAD2366.bas will load an arbitrary xxxx.bit file into the Xilinx chip.

**SPECIFICATIONS**

- Single width CAMAC module
- 1 LED indicates N line activity
- 2 programmable LEDs
- 59 I/O signals on front panel, all are differential ECL
- All front panel I/Os to and from the Xilinx chip
- Input or output is selectable in groups of 4. Inputs are terminated with 112 ohms, outputs will drive 100 ohm lines.
- Programmable gate array: Xilinx 4005-5, 196 configurable logic blocks and 112 I/O blocks fast carry logic, wide decoding approximately 5000 gates: 616 flip-flops, 6272 Ram bits. This is in a 156-pin PGA socket.
- The 4005-5 can be replaced with a 4005-4 or 4005-3, by the user, as they become available from Xilinx. The socket is also pin compatible with the Xilinx 4006.
- 40 MHz, 20 MHz, and 10 MHz crystal clocks on the board
- Programmable by optional on board (socketed) Eprom on power up, or by CAMAC command to reload from Eprom.
- Programmable via CAMAC, at any time, independent of Eprom. Xilinx XACT software system is required for programming. Uses Xilinx .BIT file for programming information.
Basic CAMAC programming software for IBM compatible included. 11875 CAMAC F16 write operations are required to program all logic must be clocked (synchronous logic).

CAMAC Interface: Programming, 8 bit write only interface. Test for successful programming. After programming, all CAMAC control and data lines (N, F, A, 24R/W, C, Z, S1, S2, Q, X, L) are available to Xilinx chip. Only 1 function code is reserved for reprogramming, all others are available for the user.

Possible Applications: Trigger logic, digital 48 input majority logic; trigger or readout controller; pipelined sequential logic; pulse sequence generator; any arbitrary state machine logic; fast memory, FIFO or LIFO, 16 bit digital adder.

### CAMAC Interface Pin Assignments for Xilinx 4005-PGA156

<table>
<thead>
<tr>
<th>CAMAC NAME</th>
<th>CAMAC NAME</th>
<th>CAMAC NAME</th>
<th>CAMAC NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>XILINX PIN</td>
<td>XILINX PIN</td>
<td>XILINX PIN</td>
<td>XILINX PIN</td>
</tr>
<tr>
<td>C</td>
<td>R10</td>
<td>the 24 bit read-write bus</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>T9</td>
<td>RW1</td>
<td>T16</td>
</tr>
<tr>
<td>I</td>
<td>E14</td>
<td>RW2</td>
<td>T14</td>
</tr>
<tr>
<td>N</td>
<td>C5</td>
<td>RW3</td>
<td>T10</td>
</tr>
<tr>
<td>S1</td>
<td>T7</td>
<td>RW4</td>
<td>R9</td>
</tr>
<tr>
<td>S2</td>
<td>A2</td>
<td>RW5</td>
<td>T8</td>
</tr>
<tr>
<td>A1</td>
<td>B4</td>
<td>RW6</td>
<td>P7</td>
</tr>
<tr>
<td>A2</td>
<td>A3</td>
<td>RW7</td>
<td>T3</td>
</tr>
<tr>
<td>A4</td>
<td>B5</td>
<td>RW8</td>
<td>P4</td>
</tr>
<tr>
<td>A8</td>
<td>B6</td>
<td>RW9</td>
<td>R1</td>
</tr>
<tr>
<td>F1</td>
<td>A5</td>
<td>RW10</td>
<td>P2</td>
</tr>
<tr>
<td>F2</td>
<td>C7</td>
<td>RW11</td>
<td>P1</td>
</tr>
<tr>
<td>F4</td>
<td>B7</td>
<td>RW12</td>
<td>N1</td>
</tr>
<tr>
<td>F8</td>
<td>A6</td>
<td>RW13</td>
<td>K3</td>
</tr>
<tr>
<td>F16</td>
<td>A7</td>
<td>RW14</td>
<td>K2</td>
</tr>
<tr>
<td>X</td>
<td>G15</td>
<td>RW15</td>
<td>J2</td>
</tr>
<tr>
<td>Q</td>
<td>G16</td>
<td>RW16</td>
<td>J3</td>
</tr>
<tr>
<td>L</td>
<td>H16</td>
<td>RW17</td>
<td>H1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RW18</td>
<td>G1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RW19</td>
<td>F1</td>
</tr>
<tr>
<td>Re*</td>
<td>T11</td>
<td>r1-r24</td>
<td>RW22</td>
</tr>
<tr>
<td>Wr1*</td>
<td>R11</td>
<td>w1-w8</td>
<td>RW23</td>
</tr>
<tr>
<td>Wr2*</td>
<td>A8</td>
<td>w9-w24</td>
<td>RW24</td>
</tr>
</tbody>
</table>

enables for bidirectional
24 bit read write bus
Re* T11 r1-r24
Wr1* R11 w1-w8
Wr2* A8 w9-w24
Direction for R1-R24, normally +5 V, pin T1

40 MHz crystal clock pin B3 (PGCK1)
divide by 2 (20 MHz) pin B16 (PGCK2)
divide by 4 (10 MHz) pin T15 (PGCK3)
FRONT PANEL

Programmable LEDs, top = D14 (red), bottom = C16 (yellow)

Front Panel Input-Output pins:

<table>
<thead>
<tr>
<th>PAIR NUMBER</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8)</td>
<td>(17)</td>
<td>(17)</td>
<td>(17)</td>
<td>(17)</td>
<td>(pairs)</td>
</tr>
<tr>
<td>1</td>
<td>F15</td>
<td>R6</td>
<td>B2</td>
<td>J15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>E16</td>
<td>T4</td>
<td>C9</td>
<td>J16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>F16</td>
<td>R4</td>
<td>B9</td>
<td>K14</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>G14</td>
<td>R3</td>
<td>A9</td>
<td>K15</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>P9</td>
<td>N2</td>
<td>C10</td>
<td>K16</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>T6</td>
<td>M3</td>
<td>B10</td>
<td>L15</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>R7</td>
<td>L2</td>
<td>A10</td>
<td>L16</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>T5</td>
<td>L1</td>
<td>B11</td>
<td>M14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>K1</td>
<td>A11</td>
<td>M16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>J1</td>
<td>C12</td>
<td>N14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>G2</td>
<td>B12</td>
<td>N15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>G3</td>
<td>B13</td>
<td>P12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>E1</td>
<td>A13</td>
<td>P15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E2</td>
<td>A14</td>
<td>P16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>C2</td>
<td>C15</td>
<td>R13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>D3</td>
<td>D15</td>
<td>T13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>R16</td>
<td>B14</td>
<td>T2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The input (and output) polarity is such that a positive ECL edge (the odd numbered pin on the input connector is a positive-going edge) produces a negative TTL edge at the Xilinx input. This gives the best possible performance for capturing short input signals (as short as 5 nsec). The input signal should be inverted inside the Xilinx chip and used as the clock to a D Flip Flop (with a logic 1 as the D input). This does mean that a normal ECL logic 1 (odd pin +) becomes a logic 0 inside the Xilinx chip. Input and output are consistent, both are inverted. We suggest as a general rule to invert the data inside the Xilinx chip as it enters and leaves. This allows normal positive logic to be used inside the Xilinx chip. These extra inverters are not really extra, they are simply absorbed inside the logic and do not usually cause any extra propagation delay.

All front panel I/O is selectable as input or output in groups of 4, as indicated below (the 3 clocks inputs are separately selectable). This is accomplished by installing the socketed ECL-TTL or TTL-ECL level converters and the appropriate termination resistor networks. All inputs and outputs are differential ECL. The corresponding Xilinx pins must be programmed as either input or output.

For input only the 10125 ECL to TTL converters and the 56 ohm termination resistor SIPs are installed. The inputs are properly terminated for twisted pair cable.

For output, only the 10124 TTL to ECL converters and the 390 ohm pull-down resistor SIPs are installed.
**Note:** The module will work properly with short cables (but with reduced noise immunity) even if both the input termination SIPs and the output pull-down SIPs are installed.

The logical polarity of any signal is programmable in the Xilinx chip of course. There are a total of 59 I/O signals.

1 **16 pin (8 pairs) header:**
   - I/O A 1-4
   - I/O A 5-8

3 **34 pin (17 pairs) headers:**
   - I/O B 1-4
   - I/O B 5-8
   - I/O B 9-12
   - I/O B 13-16
   - I/O B 17 (can be connected to SGCK1)
   - I/O C 1-4
   - I/O C 5-8
   - I/O C 9-12
   - I/O C 13-16
   - I/O C 17 (can be connected to SGCK2)
   - I/O D 1-4
   - I/O D 5-8
   - I/O D 9-12
   - I/O D 13-16
   - I/O D 17 (can be connected to SGCK3)

SGCK1, 2 and 3 are Xilinx internal clock distribution networks.

It is possible to convert the front panel I/O to bidirectional TTL. This user modification cannot be wholeheartedly recommended, however, since little protection is provided for the Xilinx chip. This method will only work at low speeds (less than 1 MHz) and for very short cable lengths. Deglitching will be needed at all receivers. The Xilinx pins cannot drive 100 ohm terminated lines. For best results we recommend using the terminated ECL for the cable runs, and converting to TTL at the destination circuit board. This is necessary if high speed performance is required.

To provide TTL connections construct this 16 pin dip header with 4 100 ohm resistors, for each group of 4 to be used as TTL.

Install this header in the 10124 location after removing both resistor packs, and both the 10124 and 10125 chips.

This user modification connects the Xilinx pin directly to the odd numbered pin on the front panel connector. The corresponding even numbered pin is grounded. There is no buffer, ONLY a series resistor to prevent damage to the Xilinx chip.
The Xilinx chip can be programmed for either INPUT, OUTPUT or as a TRISTATE pin.

Please Use With Care!

Pin

1  connect to 16
2  connect to 16
3  100 ohm resistor to 7
4  100 ohm resistor to 5
5  100 ohm resistor to 4
6
7  100 ohm resistor to 3
8
9
10  100 ohm resistor to 12
11  100 ohm resistor to 13
12  100 ohm resistor to 10
13  100 ohm resistor to 11
14  connect to 16
15  connect to 16
16  connect to 1,2,14,15 (ground)

Note: Only 4 100 ohm resistors are required.
INTRODUCTION

The following section includes examples of programming the Model 2366 and examples of software to load programs into the 2366 from CAMAC. These examples have all been tested and work as described in the specifications. They are offered as-is, as examples only. They may be freely used, but they are not supported products.

These examples, and others, are available for downloading from the LeCroy ftp site (currently www.lecroy.com).
HOW TO PROGRAM THE LECROY MODEL 2366 UNIVERSAL LOGIC MODULE

There are 4 steps to programming the LeCroy Model 2366 Universal Logic Module. These four steps are:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create Schematic.</td>
</tr>
<tr>
<td>2</td>
<td>Convert from CAD/CAM drawing to XILINX formatted netlist using XILINX programs ... generates an ASCII output file: a) Convert to self contained netlist b) perform format conversion to XILINX compatible format.</td>
</tr>
<tr>
<td>3</td>
<td>Copy ASCII file to PC and compile it with XILINX programs to generate a binary file capable of being loaded into the XILINX chip.</td>
</tr>
<tr>
<td>4</td>
<td>Write binary file from PC into model 2366 CAMAC module and XILINX chip.</td>
</tr>
</tbody>
</table>

*Other techniques for this step exist, such as VHDL or Verileg™.*

Step 1: Create Schematic

The very first step in programming the LeCroy Model 2366 Universal Logic Module is to produce a schematic using the XILINX parts software library *. The 2366 is based on the part identified in the XILINX library as part number 4005PLG156 which is described as a gate array. The pin assignments are described in the manual to the Model 2366 and correspond to the description given in the XILINX library. The XILINX library also includes many varieties of gates, flip-flops, counters, registers and so forth from which you are able to build up your design **.

In the following example, the file named test3377 represents the schematic design file which must be converted into a format compatible with the 2366 CAMAC module.

** Several examples of schematics are included in the Model 2366 user’s manual.

Step 2: Convert from CAD/CAM to XILINX format netlist using XILINX programs

a) Convert to self contained netlist { lca_expand command below }
b) Convert to XILINX compatible format { erel2xnf command below }

Here are the commands used to perform these operations:

lca_expand test3377
erel2xnf test3377 -p 4005pg156-5

The file output from the last command is called test3377.xnf and is in ASCII format. If you look at this file you would see something like the following:

LCANET,4
PROG,ERE2XNF,4.10,Created from //node_3ec48/disk_scsi5/local_user2/richard_b/test3377
01/25/1995 10:18
PART,4005pg156-5
SYM,I$5122/I$14,DFF,INIT=R

* We use a Mentor Graphics system, however, almost any of the popular CAD/CAM programs are capable of producing a compatibly formatted design file.

** Several examples of schematics are included in the Model 2366 user’s manual.
Step 3: Copy the *-XNF file to a PC (or compatible computer) and compile it with a XILINX program which generates a binary image file

The following command file (with the *.MAK extension) is used on the PC to convert the *.XNF file into a binary file with the extension *.BIN. This example uses the XILINX software version 5. Note that the lines beginning with a “#” are comment lines inserted to explain the contents of the *.MAK file.

```
########### Start of MAK file #####################
# test3377.mak PC MAKE file for XILINX builds

# Environment inherited macros, usually directories.
DESIGN_DIR=c:\xact\designs\test3377

# Miscellaneous files...*.XNF is input, others are outputs
SCHEM_FILE = $(DESIGN_DIR)test3377.xnf
MERGED_FILE = $(DESIGN_DIR)test3377.xff
PREP_FILE = $(DESIGN_DIR)test3377.xtf
PPR_PARAMFILE = $(DESIGN_DIR)test3377.ppr

# Compile and command macros
cmd_merge = xnfmerge $(SCHEM_FILE) $(MERGED_FILE)
cmd_prep = xnfprep $(MERGED_FILE)
cmd_ppr = ppr $(PREP_FILE) paramfile = $(PPR_PARAMFILE)
cmd_makebits = makebits test3377.lca

# Specify ultimate target of MAK.
all: $(DESIGN_DIR) test3377.bit

# Build the merged design.
$(MERGED_FILE): $(SCHEM_FILE): $(cmd_merge)

# Build final whatever.
$(DESIGN_DIR) test3377.bit: $(MERGED_FILE): $(cmd_prep) $(cmd_ppr) $(cmd_makebits)

########### End of MAK file #####################
```

Step 4: Write the binary image file directly into the LeCroy Model 2366

At this point there is a file named test3377.bit which can be written directly into the LeCroy Model 2366 Universal Logic Module via standard CAMAC commands.
DECLARE FUNCTION init% (file$, var$, value!) ' LOAD2366.BAS ' this program is written for Microsoft QuickBasic ' NOTE! ' a single apostrophe indicates that the rest of the line is a comment ' a variable name ending in % is a 16 bit signed integer ' a variable name ending in & is a 32 bit signed integer ' a variable name ending in $ is a string ' PRINT n1$ = "load2366.ini" jj$ = "camslot%" isok% = init%(fff$, jj$, value) IF isok% = 0 THEN INPUT "Where is the 2366? Camac slot "; value END IF modl% = value ON ERROR GOTO fault IF COMMANDS = " " THEN INPUT "Please enter the file name"; fil$ ELSE fil$ = COMMAND$ END IF start: crate% = 1 CALLS cdreg(tmod0%, 0, crate%, modl%, 0) CALLS cdreg(modl%, 0, crate%, modl%, 1) CALLS cdreg(mtslot%, 0, crate%, 2, 0) CALLS ccinit(0) CALLS cssa(9, tmod0%, dd%, qq%) PRINT : PRINT "RELOAD VIA CAMAC" OPEN n1$ FOR BINARY ACCESS READ AS #1 jk$ = "" '1 byte long CALLS ccinit(0) ' enter programming mode CALLS cssa(30, tmod0%, dd%, qq%) ' select camac mode CALLS cssa(28, tmod0%, dd, qq%) ' program pulse CALLS cssa(25, tmod0%, dd%, qq%) ' wait for init line ttt% = 0: PRINT "INIT "; DO CALLS cssa(14, tmod0%, dd%, qq%) PRINT qq%; ttt% = ttt% + 1 IF ttt% > 10000 THEN EXIT DO LOOP WHILE qq% < > 1 PRINT ' skip the first 16 bytes in the file FOR j% = 0 TO 15 ' GET #1, jk$ NEXT np% = 0 DO GET #1, jk$ num% = ASC(jk$) IF num% = 255 THEN EXIT DO IF (num% > 31) AND (num% < 123) THEN PRINT jk$; np% = 0 ELSE IF np% = 0 THEN PRINT "; np% = 1 END IF LOOP GOSUB sendit GET #1, jk$: num% = ASC(jk$): GOSUB sendit PRINT num%; lengt& = num% AND 15: lengt& = lengt& + 256 + num% GET #1, jk$: num% = ASC(jk$): GOSUB sendit PRINT num%; lengt& = lengt& + 256 + num% GET #1, jk$: num% = ASC(jk$): GOSUB sendit PRINT num%; num% = (num% AND 240) / 16 lengt& = lengt& + 16 + num% nbyte = lengt& / 8 hide% = INT(nbyte) PRINT: PRINT lengt&; "(bits) "; nbyte; hide%; "bytes" FOR j% = 1 TO hide% GET #1, jk$: num% = ASC(jk$): GOSUB sendit CALLS cssa(13, tmod0%, dd%, qq%) IF qq% = 1 THEN PRINT "done at j%="; j%: EXIT FOR NEXT num% = ASC(jk$): GOSUB sendit ' one last write CLOSE 1 PRINT "finished "; CALLS cssa(13, tmod0%, dd%, qq%) IF qq% = 1 THEN PRINT "!!!success!!!" ELSE PRINT "***FAILURE***" END IF PRINT CALLS cssa(9, tmod0%, dd%, qq%) errnum% = 0 SYSTEM sendit: CALLS cssa(16, tmod0%, num%, qq%) RETURN fault: PRINT "error" SYSTEM END FUNCTION init% (file$, var$, value!) ' read in an initialization file of the form: ' varname=value ' varname=value ' etc ' case is ignored ' one variable per line ' variables can be in any order ' init% returns as 0 for variables not found
init% = 0
ON LOCAL ERROR GOTO lerr
OPEN file$ FOR INPUT AS #1
WHILE NOT EOF(1)
    LINE INPUT #1, line$
    line$ = UCASE$(line$)
    IF (INSTR(line$, UCASE$(var$)) <> 0) AND (INSTR(line$, "=") <> 0) THEN
        jnk$ = RIGHT$(line$, LEN(line$) - INSTR(line$, "="))
        value = VAL(jnk$): init% = 1
    END IF
WEND
lerr:
CLOSE #1
EXIT FUNCTION
END FUNCTION
DECLARE FUNCTION init% (file$, var$, value!) 
  
  LDGP2366.BAS 
  
  this program is written for Microsoft QuickBasic 
  
  NOTE! 
  ' a single apostrophe indicates that the rest of the line is a comment 
  ' a variable name ending in % is a 16 bit signed integer 
  ' a variable name ending in & is a 32 bit signed integer 
  ' a variable name ending in $ is a string 
  
restart: 
  
PRINT 
fff$ = "load2366.ini"
jj$ = "camslot%"
isok% = init%(fff$, jj$, value)IF isok% = 0 THEN 
  INPUT "Where is the 2366 to load? Camac slot "; value 
END IFmodl% = value 
IF COMMANDS = "" THEN 
  INPUT " Please enter the XXX.BIT file name to load"; fil$ 
ELSE 
  fil$ = COMMANDS 
END IF 
start: 
  
CALLS gpinit 
CALLS gpsec 
CALLS gpron 
crate% = 1 
CALLS gpcini(crate%) 
CALLS gpcccz(crate%) 
  
' this is the syntax for the CAMAC subroutines 
' LeCroy 8901 and National Instruments GPIB card 
' CALLS GPCFSA(crate%, f%, a%, n%, int4&, xq%) 
' x=1, q=2, so x&q=3 
ulm% = modl% 
CALLS GPCFSA(crate%, 9, 0, modl%, dd&, qq%) 
GOSUB reload 
CALLS GPCFSA(crate%, 9, 0, modl%, dd&, qq%) 
  
INPUT "Load another? (y/n)"; yn$ 
IF UCASE$(yn$) = "Y" THEN GOTO restart 
  
SYSTEM 
reload: 
  
PRINT " RELOAD VIA CAMAC "; 
OPEN fil$ FOR BINARY ACCESS READ AS #1 
jk$ = " "; 1 byte long 
rel2: 
  ' enter programming mode 
CALLS GPCFSA(crate%, 30, 0, ulm%, dd&, qq%) 
  ' select camac mode 
CALLS GPCFSA(crate%, 28, 0, ulm%, dd&, qq%) 
  ' program pulse 
CALLS GPCFSA(crate%, 25, 0, ulm%, dd&, qq%) 
  ' wait for init line 
  ttt% = 0: PRINT " INIT "; 
CALLS GPCFSA(crate%, 14, 0, ulm%, dd&, qq%) 
IF qq% = 3 THEN GOTO rel2 
  
DO 
CALLS GPCFSA(crate%, 14, 0, ulm%, dd&, qq%)
' varname=value
' varname=value
' etc
'
' case is ignored
' one variable per line
' variables can be in any order
' init% returns as 0 for variables not found
'
init% = 0
ON LOCAL ERROR GOTO lerr
OPEN file$ FOR INPUT AS #1
WHILE NOT EOF(1)
    LINE INPUT #1, line$
    line$ = UCASE$(line$)
    IF (INSTR(line$, UCASE$(var$)) <> 0) AND (INSTR(line$, "=") <> 0) THEN
        jnk$ = RIGHT$(line$, LEN(line$) - INSTR(line$, "="))
        value = VAL(jnk$): init% = 1
    END IF
WEND
lerr:
CLOSE #1
EXIT FUNCTION
END FUNCTION
DECLARE FUNCTION init%(file$, var$, value!) 
' 
' EPRM2366.BAS 
' 
' this program is written for Microsoft QuickBasic 
' 
' NOTE! 
' a single apostrophe indicates that the rest of the line is a comment 
' a variable name ending in % is a 16 bit signed integer 
' a variable name ending in & is a 32 bit signed integer 
' a variable name ending in $ is a string 

PRINT 
fff$ = "load2366.ini"
jj$ = "camslot%"
isok% = init%(fff$, jj$, value)
IF isok% = 0 THEN 
END IF
modl% = value

start:
crate% = 1
CALLS cdreg(tmod0%, 0, crate%, modl%, 0)
CALLS cdreg(tmod1%, 0, crate%, modl%, 1)
CALLS cdreg(mtslot%, 0, crate%, 2, 0)
CALLS ccinit(0)
CALLS cssa(9, tmod0%, dd%, qq%)
' REprogram using EPROM
' enter programming mode
CALLS cssa(30, tmod0%, dd%, qq%)
' program pulse
CALLS cssa(25, tmod0%, dd%, qq%)
' wait for init line
ttt% = 0: PRINT " INIT ";
DO
CALLS cssa(14, tmod0%, dd%, qq%)
PRINT qq%;
ttt% = ttt% + 1
IF ttt% > 1000 THEN EXIT DO LOOP WHILE qq% <> 1
PRINT
PRINT "RELOADED FROM THE EPROM ";
tt = TIMER
DO
IF (TIMER - tt) > 1 THEN EXIT DO
' test DONE
CALLS cssa(13, tmod0%, dd%, qq%)
IF qq% = 1 THEN EXIT DO LOOP
IF qq% = 1 THEN PRINT "!!!success!!"
ELSE
PRINT "****FAILURE***"
END IF
PRINT "enter Q to quit or R to restart" dwrt& = 0
RANDOMIZE TIMER
CALLS cssa(9, tmod0%, dd%, qq%)
errnum% = 0
PRINT "enter Q to quit, or R to reprogram"
RANDOMIZE TIMER
FOR fff% = 0 TO 31
FOR aaa% = 0 TO 15
CALLS cdreg(mtslot%, 0, crate%, 2, aaa%)
CALLS cssa(fff%, mtslot%, dat&, qf%)
CALLS cssa(1, tmod0%, dat&, qr%)
df& = dat& AND (256 + 128 + 64 + 32 + 16)
df &= df& / 16
IF df& <> fff% THEN PRINT "ftest "; fff%; dat&; df&; qf%; qr%;
da& = dat& AND 15
IF df& <> aaa% THEN PRINT "atest "; aaa%; dat&; da&; qf%; qr%;
NEXT
PRINT "Address and Function code test complete"
CALLS cccc(mtslot%)
CALLS cssa(1, tmod0%, dat&, qr%)
PRINT dat& AND 3584;
IF dat& AND 2048 = 0 THEN PRINT "C failed ";
CALLS cccc(mtslot%)
CALLS cssa(1, tmod0%, dat&, qr%)
PRINT dat& AND 3584;
IF dat& AND 1024 = 0 THEN PRINT "Z failed ";
CALLS cccc(mtslot%, 255)
CALLS cssa(1, tmod0%, dat&, qr%)
PRINT dat& AND 3584;
IF dat& AND 512 = 0 THEN PRINT "I failed ";
CALLS cccc(mtslot%, 0)
PRINT " C,Z,I test complete"
PRINT "Begin 24 bit random write/read test"
npass = 0
DO
FOR jklmn% = 0 TO 10000
r1% = INT(RND * 4095.999)
r2% = INT(RND * 4095.999)
dwrt& = r1%
dwrt& = dwrt& * 4096 + r2%
CALLS cssa(16, tmod0%, dwrt&, qw%)
CALLS cssa(0, tmod0%, drd&, qr%)
IF dwrt& <> drd& THEN
LOCATE 24, 1: errnum% = errnum% + 1
PRINT "read write error "; errnum%; dwrt&; qw%; drd&; qr%;
END IF
kkk$ = UCASE$(INKEY$)
IF kkk$ = "R" THEN GOTO start
IF kkk$ = "Q" THEN SYSTEM
NEXT
npass = npass + 1
LOCATE 1; PRINT " pass complete, # "; npass;
SYSTEM
END
FUNCTION init%(file$, var$, value)
' read in an initialization file of the form:
' varname=value
' varname=value
' etc
' case is ignored
' one variable per line
' variables can be in any order
' init% returns as 0 for variables not found
' init% = 0
ON LOCAL ERROR GOTO lerr
OPEN file$ FOR INPUT AS #1
WHILE NOT EOF(1)
LINE INPUT #1, line$
line$ = UCASE$(line$
IF (INSTR(line$, UCASE$(var$)) <> 0) AND (INSTR(line$, ",") <> 0) THEN
    jnk$ = RIGHT$(line$, LEN(line$) - INSTR(line$, ",")
    value = VAL(jnk$); init% = 1
END IF
WEND
lerr:
CLOSE #1
EXIT FUNCTION
END FUNCTION
59 BIT INPUT MODULE

Xilinx bit file is T2366IN.BIT

The I/O connectors should be configured as all inputs. The 10125 ECL to TTL converters and the red termination SIPs should be installed. All 10124 ICs and all yellow pull-down SIPs should be removed. No damage will result if the configuration is incorrect, but the module will, of course, not function correctly.

CAMAC Function Codes

F0, A0  Read 8 bits from the A input connector.
F0, A1  Read 17 bits from the B input connector.
F0, A2  Read 17 bits from the C input connector.
F0, A3  Read 17 bits from the D input connector.

In all cases, the numbered inputs correspond to the CAMAC R lines.

This module will never return X or Q.
Figure 2
Figure 4
Figure 5
The I/O connectors should be configured as all outputs. All 10124 TTL to ECL converters and all yellow pull-down SIPs should be installed. The 10125 ECL to TTL converters and the red termination SIPs should be removed. This should be carefully checked, it is possible to damage the Xilinx chip if any channels are configured as inputs.

This module also contains a 24 bit read write register, and a 13 bit last command register. On every S2, the A line, F lines, I, Z, C, and N are latched (A1 is R1, F1 is R5, I is R10, etc). These can be read on the next CAMAC cycle.

**CAMAC Function Codes**

- **F0, A0** Read from the internal 24 bit register.
- **F1, A0** Read from the 13 bit last command register.
- **F16, A0** Write to the internal 24 bit register.
- **F17, A0** Write 8 bits to the A input connector.
- **F17, A1** Write 17 bits to the B input connector.
- **F17, A2** Write 17 bits to the C input connector.
- **F17, A3** Write 17 bits to the D input connector.

In all cases, the numbered inputs correspond to the CAMAC R lines.

The module returns Q and X for the F0, F1, and F16 commands. Neither Q nor X is returned for the F17 commands.
Figure 7
Figure 8
Figure 9
CAMAC DATAWAY
TEST MODULE

Xilinx bit file is T2366E.BIT

This module is loaded in the eprom which is installed in the 2366 when shipped.

This module contains a 24 bit read write register, and a 13 bit last command register. On every S2, the A line, F lines, I, Z, C, and N are latched (A1 is R1, F1 is R5, I is R10, etc). These can be read on the next CAMAC cycle.

The I/O connectors can have any configuration, including all ICs and SIPs installed (the shipping configuration). No damage will result, but the power dissipation will be higher than normal. None of the I/O lines are driven by the Xilinx chip.

CAMAC Function Codes

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0, A0</td>
<td>Read from the internal 24 bit register.</td>
</tr>
<tr>
<td>F1, A0</td>
<td>Read from the 13 bit last command register.</td>
</tr>
<tr>
<td>F16, A0</td>
<td>Write to the internal 24 bit register.</td>
</tr>
</tbody>
</table>

The module returns Q and X for the F0, F1, and F16 commands.
Figure 14
Figure 15
### Specifications

- **48 inputs**, differential ECL. Pulse width greater than 5 nsec. Synchronized internally with the clock.
- 48 bit input mask register to enable/disable individual inputs.
- Output: 6 bit digital word equal to the number of inputs true within the coincidence resolving time.
- Selectable clock rate, internal 10, 20 or 40 MHz or a choice of 3 external clock inputs (40 MHz maximum).
- 40 MHz output rate (40 MHz clock)
- Output pipeline delay: 175 nsec (40 MHz clock)
- Coincidence resolution: minimum 25 nsec, maximum 50 nsec (40 MHz clock)
- Configuration: connector A, all output, Connectors B, C, and D are all input.

Complete schematic and Xilinx programming files are available on request.

### Description

The LeCroy CAMAC Model 2366 Universal Logic Module is a general purpose logic module using a Xilinx 4005 field programmable gate array. The module has 59 front panel i/o signals, which can be configured as either input or output in 7 groups of 4 channels, and 3 individual channels. In addition there is a full 24 bit CAMAC interface. The Xilinx gate array can be programmed with an on board EPROM, or from CAMAC.

For use as the 48 input majority logic module, the 2366 is configured as 48 inputs and 8 outputs. Three of the 59 possible i/o are used for an external clock. The 48 inputs are synchronized with the clock and trigger a 2 clock period one-shot. The inputs are edge sensitive, triggering on the leading edge of pulses as short as 5 nsec. Seven pipelined adder stages sum the 48 input signals to produce the binary output word. The output word is updated at the clock rate, with a 7 clock period pipeline delay. A 48 bit mask register (CAMAC F16, A0 and A1, read with F0) is used to enable (=1) or disable (=0) the individual inputs. The coincidence resolution for determining the accidental triggers is 1.5 clock periods. Real events must have all inputs within a span of one clock period or less to trigger with 100% efficiency.

This program uses most of the resources of the Xilinx 4005 in the 2366 module. It uses 243 function generators (62%) and 352 CLB flip-flops (90%). In order to operate at 40 MHz, adders with more than 3 input bits are split into 2 pipeline stages.

Several variations of this circuit are possible. By slowing the maximum clock rate to 20 MHz, the circuit can be simplified to four pipeline stages. The coincidence resolution for a 20 MHz clock will be 75 nsec for
accidentals, and 50 nsec for real events. The pipeline delay would be 200 nsec, with a 20 MHz output word rate.

If the input signals are wide pulses, a different synchronizing circuit is possible. The input can simply be clocked with the selected clock using the i/o flip flop, then differentiated to produce a short pulse. This method frees up two extra flip flops per input, which can be used to lengthen the pulse from 2 to 3 clock periods. This changes the coincidence resolution to 2.5 clock periods for accidentals and 2 clock periods for real events.

The output word is a 6 bit binary number which is the number of inputs that were true during the clock period 7 clocks earlier. If 48 inputs is sufficient, the 6 bit output word of the majority logic can be digitally compared with a CAMAC loaded register and produce a trigger signal directly.

For systems which require a larger majority logic unit, multiple 48 input majority logic units can be combined. The 6 bit output word can be time aligned (in clock steps) and added to the outputs of other majority logic modules, using another 2366 module. A single 2366 module configured as an adder tree can add eight 6 bit input words in 7 pipeline stages. The output word is limited to 8 bits in this example, so the sum will limit at 255 (there can be as many as 384 inputs). If counting to beyond 255 is required, The 2366 can be configured to have fewer inputs and a wider output word (in units of 4 bits).

Extending this to one more level (adding another 2366 to sum the outputs of the adders) allows six 8 bit inputs to be counted in 9 pipeline stages (again limiting at 255). This produces a 2304 input majority logic in a total of 25 pipeline stages (allowing one stage for going from module to module). When using multiple 2366 modules, a common clock should be used to synchronize the modules. Each 2366 has 3 inputs which can be used as the external clock source.

This majority logic program consumes most of the resources of the Xilinx chip. There is very little left with which to implement test features. A trigger system for a large high energy physics experiment is usually very complex, with many modules and very, very many cables and connectors which combine the logic modules to produce the desired trigger algorithm. Testing a system like this usually requires looking at signals with oscilloscopes and logic analyzers. It also usually requires disconnecting cables. It is usually not possible to test the system completely in situ. This allows many possibilities for errors, and for undetected (until long after the run) problems such as intermittent or missing connections.

The 2366 universal logic module solves this problem by allowing COMPLETE reprogrammability of the trigger logic. This is dynamic reprogramming, on the fly, without moving a single cable of jumper anywhere in the system. Test logic can be downloaded over CAMAC in a second or two per module, completely replacing the trigger logic. This test logic can be as simple as read write registers which drive the cable connections (testing both the integrity of the connections AND verifying the topology) to complex pattern generators which test the trigger algorithms of other modules. When testing is finished (which could require several different downloads of test logic) the trigger logic is simply reprogrammed with the
trigger algorithms and the experiment is ready to run, with a much higher confidence level than normally found in trigger systems.

This reprogrammability also allows the trigger to be easily and quickly changed for new requirements. If the topology is the same, or a superset of the previous topology, then switching back and forth between the old and new triggers can be done quickly AND safely, without touching a cable.

**CAMAC Function Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0, A0</td>
<td>Read enable mask for bits 1-24</td>
</tr>
<tr>
<td>F0, A1</td>
<td>Read enable mask for bits 25-48</td>
</tr>
<tr>
<td>F0, A2</td>
<td>Read CLOCK select register, bits 1-8</td>
</tr>
<tr>
<td>F16, A0</td>
<td>Write enable mask for bits 1-24</td>
</tr>
<tr>
<td>F16, A1</td>
<td>Write enable mask for bits 25-48</td>
</tr>
<tr>
<td>F16, A2</td>
<td>Write CLOCK select register, bits 1-8</td>
</tr>
</tbody>
</table>

**CLOCK select register**

- 0 = internal 10 MHz clock
- 1 = internal 20 MHz clock
- 2 = internal 40 MHz clock
- 3 = external clock, pair B17
- 4 = external clock, pair C17
- 5 = external clock, pair D17
Figure 20
A LONG RANGE MULTIHIT
TIME DIGITIZER WITH
100 NS RESOLUTION

The Xilinx bit file is SLOWTDC.BIT

The top connector on the 2366 ULM, I/O A, must be configured for input. The other connectors are not used and can be configured either as in or out without damage to the module.

The START pulse is the OR of I/O A pins 1, 2, 3 and 4. The HIT pulses are the OR of I/O A pins 5, 6, 7 and 8. The minimum width for the START and HIT inputs is 10 nsec. The inputs are edge triggered and inverted, a negative edge (at the left hand pin) is the time to be measured.

The module should be cleared with F9, and enabled with F26. The first START pulse is synchronized with the internal 40 MHz clock. Then the HIT inputs, the 24 bit counter and a divide by 4 counter (with 40 MHz input) are enabled. The divide by 4 provides a 10 MHz clock for the 24 bit counter and the data storage FIFO. The clock remains running until the next F9. Note that the START time is quantized with the 40 MHz clock, and resets the phase of the 10 MHz clock.

When a HIT arrives, the pulse is synchronized with the 40 MHz clock and triggers an updating one shot with 100 nsec output width. The one shot can retrigger after 50 nsec.

The one shot output is synchronized with the 10 MHz clock and causes the contents of the 24 bit counter to be stored in the FIFO. The FIFO can hold up to 31 HIT times.

Note that the HIT time is quantized with the 10 MHz clock. The RMS quantizing error for the time difference, HIT - START, is 30 nsec, the sum in quadrature of the START error and the HIT error. The time zero must be calibrated, of course.

The module must be disabled with F24,A0 before readout. The data is read from the FIFO with F0,A0. This command returns Q true if there was a valid data word in the FIFO.

<table>
<thead>
<tr>
<th>CAMAC Function Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9,A0</td>
<td>Clear data, reset clocks, disable inputs.</td>
</tr>
<tr>
<td>F26,A0</td>
<td>Enable module (arm the start logic).</td>
</tr>
<tr>
<td>F27,A0</td>
<td>Test Enabled state.</td>
</tr>
<tr>
<td>F27,A1</td>
<td>Test FIFO not empty.</td>
</tr>
<tr>
<td>F27,A2</td>
<td>Test Start pulse received.</td>
</tr>
<tr>
<td>F24,A0</td>
<td>Disable Inputs (stop acquisition, put end mark in FIFO).</td>
</tr>
<tr>
<td>F0,A0</td>
<td>Read one data word from FIFO, Advance FIFO.</td>
</tr>
</tbody>
</table>

The proper sequence is:
F9, F26
a start pulse
up to 31 timing pulses within 1.6 seconds after the start
F24
F0 until no Q
The LSB of the 24 bit data is 100 nsec
Figure 25
Figure 26
Figure 27