SEQSI

M.Morrissey

July 17, 1996

1 INTRODUCTION

The Sequencer for use in Silicon readout Investigation (SEQSI) is essentially a programmable multi–channel pulse generator, which may be used to provide control signals for driving silicon detector front–end readout chips such as those being produced by RD2 and RD20. It is, however, a general–purpose module and is not restricted to any particular chip or readout system.

1.1 FORMAT

The sequencer is a single width, 6U high, D16 A24 VME module. It uses +5V and -5.2V.
The -5.2V (not a VME standard supply voltage) can be provided either via the J2 connector (using VXI defined pins) or by using the JAUX connector as used on the CERN type VME crates. Most VME crates other than the CERN type are incompatible with a module fitted with a JAUX connector. Therefore sequencers will be supplied with *either* JAUX or J2 as specified by the user. There are 22 channels available for external use, together with 4 clock lines. The clock lines and 20 of the output channels are available at a 50–way front panel IDC header at balanced ECL levels. The other 2 output channels are available at Lemo front panel connectors at NIM levels.
A NIM trigger input is provided which allows sequences to be initiated by external random triggers.
A 16-bit input port is provided which allows status bits to be read.

2 Circuit Description

The heart of the circuit is shown in fig 2. The data memory is 32 bits wide, 64 k deep. This memory is loaded from the VME backplane with the required data. The address is provided by a counter, which is incremented by the clock. The clock can be selected to be 67, 40, 20, 10, 5 or 2.5 Mhz. The clock also latches the data from the memory. The 20 least significant bits of the latch are output after conversion to ECL. The most significant bit (B31) is used to control the length of the sequence. When this bit is asserted the address counter is loaded with an address contained in the address latch (pre-loadable from VME) and the sequence continues until B31 is again asserted.
Fig 3 illustrates a very simple sequence. If this sequence is started at address zero, and with

zero in the address latch, then the output will be as shown. Note that B31 is asserted at address 4, but B5 is seen asserted. This is because B31 is in fact pipelined and causes the counter to be loaded with the contents of the address latch 1 clock period after it becomes asserted at the data latch. The sequence therefore continues for 1 clock period after B31.
Note that for simplicity the clock control bit has not been indicated in the sequence data, this is essential and will be described later.

2.1 The output channels

fig 5 shows the output channels at the 50-pin header (PL2). The output channels correspond to sequence memory bits. B(0:19) and B(29:30) are output channels, B(27:28) and B31 are used internally. The 20 output channels are divided into two groups, B(0:3) and B(4:19).
Normally a channel output has a duration which is a multiple of the clock period. If a channel is asserted for successive periods its output is a single pulse 2 clock periods wide. See fig. 4
Some applications might require pulses during successive clock periods. B(0:3) can provide this facility. Either distinct pulses or contiguous output can be selected individually for these outputs. The selection is made by clearing or setting bits 12:15 of the **SIGNAL_CONTROL** register. 0 = contiguous output, 1 = pulsed output.
B(0:3) can also be individually delayed, for up to 7 steps of 2 nS each. The delay for B0 is set by bits 0:2 of the **SIGNAL_CONTROL** register, for B1 by bits 3:5, for B2 by 6:8 and for B3 by bits 9:11
0 = no delay, 7 = 14 nS delay.
B(4:19) are fixed in phase relative to the **internal** clock but can be individually inverted by setting a bit in the **POLARITY CONTROL** register. each of these 16 outputs is the OR of the data latch and a latch which is directly writable (the **DIRECT register**. Thus these 16 outputs can be driven directly from VME.
B29:30 are output directly (ie no delay, no polarity control) at 2 Lemo front panel sockets. These are useful for such things as scope triggering.

2.2 Clock

The clock is provided by either a 67 or 40 MHz crystal. 20, 10, 5 and 2.5 MHz, derived from the 40 MHz crystal can also be selected. Selection is made by writing to bits 0:2 of the **CLOCK_CONTROL** register.
0 = 67 MHz, 1 = 40 MHz up to 5 = 2.5 MHz
The clock is bufferred to two lines each of which can be individually delayed by 7 steps of 2 nS each. One of these is converted to ECL and output, the other is output as 3 signals, independently driven. The delay for the single clock output is set by bits 3:5 of the **CLOCK_CONTROL** register, that for the triple output by bits 6:8. 0 = no delay, 7 = 14 nS delay.
The clocks may be turned off and on – ie the sequence may be stopped and restarted under computer control and turned off by a sequence bit (B28). Asserting the bit turns the clock off. If turned off by B28 being asserted it must be restarted by computer. The *outputting* of clocks is **enabled** by a sequence bit, B27. This bit must be asserted in order to enable the clock output drivers. No other channel is affected. Note that the turning on and off of the

clocks is clean, ie the circuit is timed so as not to issue shortened pulses. This is not true of the disabling circuit. Clock on / off is controlled by bit 9 of the **CLOCK_CONTROL** register. 1 = clock off. The clock is turned off by the OR of clock_control register(9) and B28.

3 Random Triggers

The sequencer was designed to be capable of responding to random external triggers. fig 6 shows the way this feature is implemented. There are 2 register which can provide an address to be loaded into the memory address counter when parallel load is asserted to the counter. One is called the jump address register. The output of this register is normally enabled, while the output of the other register, the interrupt address register, is tri-stated. When a trigger pulse (NIM type) is fed to the trigger input front panel Lemo socket it is synchronised by the system clock and then parallel load is asserted to the counter with the interrupt register output enabled, and that of the jump address register disabled. The sequence then jumps to the location specified in the interrupt register and continues until the jump bit, B31 is asserted, when the sequence returns to the location specified in the jump address register.Only the leading edge of the trigger pulse is used, the pulse can be any width greater than 10 nS
A way in which a sequence with random triggers might be used is illustrated in fig 6a.
The sequence starts at a, which is the number loaded into the address counter before starting the sequence (ie turning the clock on). Typically the sequence would now reset the front-end chip and any peripheral circuit needing to be initialised. From b to c would typically be an idle loop, in which the sequence would stay until receipt of a trigger. After receipt of the trigger the sequence goes to d (the number loaded into the interrupt address register before starting the sequence). From d to e the first part of the sequence could be the generation of a delay until the end of the L1 trigger latency time of the chip under test. Note that the trigger signal is pipelined for 2 clocks as part of its synchronisation to the clock. This has to be allowed for in calculating trigger latency delay, as must the cable delays and transition times of the trigger generating circuit.
Also, the external trigger circuit must ensure that 2 successive triggers do not occur with less than 3 clock periods between their leading edges.
If required, the sequence can also now perform a readout sequence ie controlling a multiplexer, generating ADC strobes, etc.

3.1 Multiple Triggers

Note that the trigger circuit is recursive ie if a trigger is received while the sequence is in the trigger handling part (d to e) then the sequence will jump to the beginning of this part (d). To avoid this, and also to avoid accepting a trigger while in the "reset" part of the sequence (a to b) one of the sequencer bits may be used as a "trigger enable" bit, either as a level during the idle loop or as a pulse at the beginning of the idle loop. Fig 6b shows some features of the sequencer associated with the random trigger facility. In some cases it may be desirable to control the number of triggers accepted, for example for address fifo tests. for

this reason a 4-bit trigger counter is provided. The output of this counter is compared with the contents of the least significant 5 bits of of a register (INT_REG, address hex 1A). This register is writable from vme. When the register is written to, the counter is automatically cleared. The sequencer will respond to triggers only if the counter is less than the register. The least significant 4 bits are compared to the counter, the next bit is a control bit which determines the comparator output when the counter is equal to the ls 4 bits of the register. Writing hex 10 to the register disables external triggers (ie they are ignored), writing F to it means external triggers are always enabled. Writing n to the int_reg where n is less than F will enable n + 1 external triggers after which they are disabled (ignored). Writing 1n to the int_reg where n is between 1 and 14 inclusive will enable n external triggers after which they are disabled (ignored).
One other method for disabling external triggers exists. If bit5 of the int_reg (6th lsb) is set then external triggers will be disabled(ignored) if bit 13 (3rd msb) of the input connector (the 34-way front-panel idc header) is asserted, irrespective of the state of the counter or int_reg. Note that it is essential when loading the sequence data memory that either no external triggers are sent to the sequencer or that they are disabled within the sequencer.

3.2 Status Register

An 8-bit register can be read (address = hex 1C) which gives the state of the trigger counter (4 lsbs). the 8th bit shows whether the clock is on or off. Bits (6:4) are zero. This 8-bit register is read as a 16-bit register and the top 8 bits should be ignored.

4 VME Addressing

See fig 7. The sequencer occupies an address space of 64kBytes. It is a D16, A24 module which may be addressed as program or data, standard or privileged. The base address is selected by means of SW1, an 8-way dil switch (the only one on the board).
The left-hand switch corresponds to A16, the right-hand switch to A23.
Down = 1, up = 0.
fig n shows the memory map. All addresses given in this section are to be added to the base address. Only even byte aligned addresses are used, with 16 bit data.

4.1 The Memory

The sequence memory is readable and writable from VME. The VME address bus is not used directly. Instead, the memory address is provided by the counter. To write or read the memory the clock (which increments the memory address counter) must be turned off.
The start address must be first loaded into the address register by writing the required address as data to address 4. (this register is named JUMP_ADDRESS register in fig 7, for reasons explained in the section on random triggers. Next the contents of the address register are transferred to the memory address counter by writing to it, ie executing a write to its address which is 16 (hex), the VME data is ignored. Data can now be written to or read from the memory, in 2 16 bit words, the low order 16 bits by writing to address 0, the high

order 16 bits by writing to address 2. After reading or writing the high-order 16 bit word the memory address counter is incremented, so that to transfer a block of data the start address need only be sent once. Data should always be read or written low 16 bits first, then the high 16 bits. Note that the counter and the address latch provide word addresses, a word being 4 bytes. Therefore if the counter is set to 1 it will address the 2nd 32-bit word of the sequence memory.

4.2 Control Registers

The 4 control registers, POLARITY, DIRECT, SIGNAL, and CLOCK are all 16-bit registers, write only, with addresses as given in fig 7.

4.3 Memory Data Register

This register latches all 32 bits of the sequence memory. At power up the state of the bits is unknown and before any operation is performed on the sequencer this register must be cleared. This is done by writing to it, at address 18 (hex) the data is ignored.

4.4 INPUT

Input is 16 bits at balanced ECL level at the 34-pin header PL1. (pins 1–32 are used, positive inputs on odd pins.There is no latching, a read from address 1E (hex) simply reads the state of the input pins.

5 Software Example

The following code compiles and runs on a Mac under ThinkC. It generates a simple sequence and runs it.

```
#include <stdio.h>
#include <console.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>


#define CLOCK_DRIVERS_ENABLE 0x8000000;
#define SCOPE_TRIG 0x40000000;
#define JUMP 0x80000000;



long *big_buff_ptr;


short *CLOCK_CONTROL, *MEMORY_DATA_REG, *JUMP_ADDRESS_REG, *MEMORY_ADDRESS_CTR;
short *MEMORY_LOW_WORD, *MEMORY_HIGH_WORD;
short *POLARITY_CONTROL, *DIRECT, *SIGNAL_CONTROL;

main()
{
     short i;
     long li;
     MEMORY_LOW_WORD = (short *)0xd0010000;   /* the most significant 2 digits
                                               selects slot d in the Mac to
                                               address the Micron card, the
                                               next 2 digits are the base
                                               address of the sequencer */

     MEMORY_HIGH_WORD = (short *)0xd0010002;
     JUMP_ADDRESS_REG = (short *)0xd0010004;
     MEMORY_ADDRESS_CTR = (short *)0xd0010016;
     MEMORY_DATA_REG = (short *)0xd0010018;
     POLARITY_CONTROL  = (short *)0xd001000A;
     DIRECT   = (short *)0xd001000C;
     SIGNAL_CONTROL    = (short *)0xd001000E;
     CLOCK_CONTROL = (short *)0xd0010008;
     big_buff_ptr = (long *)calloc(4,0x7ff0);
```

```
if(big_buff_ptr == NULL)
{
     printf("big_buff_ptr 0\a\a\n");
     exit(0);
}
                               /* first generate a sequence as an array
                               of 32-bit integers */
for(i = 0;i < 8050;i++)
    big_buff_ptr[i] = CLOCK_DRIVERS_ENABLE;
                                     /* CLOCK_DRIVERS_ENABLE is defined
                                        as 0x8000000, which is bit27 of
                                          the memory. this bit must be
                                          asserted in order to enable
                                          the drivers */
for(i = 0;i < 4000;i++)
    big_buff_ptr[i] |= SCOPE_TRIG;
                                   /* 0x40000000 or bit30. The sequence
                                       will be about 8000 clocks long,
                                      so this gives an easy-to-see and
                                      easy-to-trigger-on approx square
                                            wave signal at the
                                            front-panel lemo skt */

                              /* this gives a 1 clock-period
                               pulse which progresses across
                                  the 20 output channels
                                 starting at 8000 clocks after the
                                  beginning of the sequence and at
                                      one clock intervals */
for(i = 8000,li = 1;i < 8020;i++)
{
     big_buff_ptr[i] |= li;
     li = li << 1;
}
    big_buff_ptr[8020] |= JUMP;         /* JUMP is bit 31, and is defined

                                             as 0x80000000  */

/* load this sequence into memory. first the clock must be turned off
  and the mem output reg cleared */
*CLOCK_CONTROL = 0x200;
wait(1);                      /* pause of 1 sec so that absence of clocks
                                     can be seen on scope */
*MEMORY_DATA_REG = 0;
```

```
   /* now set memory address counter to zero */
    *JUMP_ADDRESS_REG = 0;
    *MEMORY_ADDRESS_CTR = 0;
                             /* for this op the data is dummy, and contents
                              of jump_address_reg are transferred to the
                              counter */
    for(i = 0;i < 8030;i++)
    {
         *MEMORY_LOW_WORD = big_buff_ptr[i] & 0xffff;
         *MEMORY_HIGH_WORD = (big_buff_ptr[i] >> 16) & 0xffff;
                                         /* the mem address ctr is
                                         automatically incremented
                                         after writing (or reading)
                                          the mem high word */
    }
    test_memory();  /* this subroutine reads back the memory (as much as
                           was written) and compares it with whats in
                              big_buff. It tells how many errors */
    /* now set all registers to known states */
    *POLARITY_CONTROL = 0;      /* no change of polarity */
    *DIRECT = 0;
    *SIGNAL_CONTROL = 0;        /* delays set to zero and output for full
                                              clock period */
    *JUMP_ADDRESS_REG = 0;
    *MEMORY_ADDRESS_CTR = 0; /* set mem addr ctr to 0 so that sequence
                                        will start at 0 */
                           /* jump addr reg contains 0, so sequence will
                             jump back to zero */
    *MEMORY_DATA_REG = 0;
    *CLOCK_CONTROL = 1;      /* sets clock speed to 40 MHz, clock delays
                              both to zero and clock on */
    printf("sequence started\a\n");

 }
 /*****************************************************************/

         wait  (secs)
         short secs;
         {
              time_t t0;
              t0 = clock();
              while((clock() - t0) < (secs * 60));
         }
/*******************************************************************/
```

```
test_memory  ()
{
    long li;
    short i,a,b,errct = 0;

  *JUMP_ADDRESS_REG = 0;
   *MEMORY_ADDRESS_CTR = 0; /* set mem addr ctr to 0 */
   *MEMORY_DATA_REG = 0;
 for(i = 0;i < 8030;i++)
    {
          a = *MEMORY_LOW_WORD;
          b = *MEMORY_HIGH_WORD;
                                         /* the mem address ctr is
                                         automatically incremented
                                         after writing (or reading)
                                              the mem high word */
          li = ((long)b << 16) | a;
          if(li !=  big_buff_ptr[i])
          {
               errct++;
               if(errct < 20)
               {
                 printf("li:%8lX  %8lX  %4d\n",li,big_buff_ptr[i],i);
               }
          }
    }
    printf("test_memory: errors = %d\n",errct);
}
```

6 Preparing and Testing

Note that later manufactured sequencers are sent out with the 67 MHz oscillator (XL2) replaced by a 50 MHz one. This is to simplify the setting up of the module. All the information which follows relating to timing assumes that a 50 MHz oscillator is fitted.

6.1 Rectification

See circuit diagram, sheet 7.
Pin 8 of U9 is erroneously conected to U9.13. These must be disconnected. The simplest method is to remove pin 13 of U9 (best done by flexing the broad part of the pin which usually results in a clean break at the package).

All ICs are mounted in sockets with integral decoupling capacitors. If 2 of these sockets are butted up so that the +5V pin of one socket is next to the ground pin of another there is a (small) risk of a short circuit. This situation exists on the board at 3 places, indicated in fig n. The board should be inspected at these places (use magnifier) and, if necessary, clear some metal from the sockets. (A fine watchmaker's screwdriver works well)

6.2 Wire_Wrapped Links

Three wire wrap links have to be made. These may have to be changed later to adjust the timing. Connect Pl3.1a to Pl4.1 Pl3.1b to Pl4.3 Pl3.3a to Pl4.2

6.3 Terminating Resistors

Check that each delay line on the board has terminating resistors fitted. These should be regarded as Select On Test. The circuit diagrams indicate 2 220 Ohm resistors between Ground and +5V, but 100 Ohms to ground is also satisfactory in most cases, and this is what should initially be fitted. NOTE that the upper pair of the 4-pin socket holds the resistor to ground.

6.4 Negative Power Supply

The module requires -5.2V (approx 1A)
When the module is to be used in a CERN standard crate the 30-pin connector (J2 in the circuit diagram) should be fitted. This is usually known as JAUX.

When the module is to be used in an industry standard VME crate then the 96-pin connector J3 (J3 in the circuit diagram, normally known as J2) should be fitted. Since -5.2V is not a VME standard supply, an auxilliary supply must be connected via J3 (which connects to the bottom socket of a standard crate)
Do NOT fit BOTH the 30-pin AND the bottom 96-pin connector

Where a module intended for use in a CERN crate is to be tested in a standard crate, -5.2V may be provided for test purposes by any means possible. Connecting a supply by soldering a pair of wires to pins 16 (0V) and 8 (-5.2V) of U65 is ok, with U65 being replaced by a new one aftr the test.

6.5 Visual Inspection

Inspect module, particularly looking for mis-inserted ICs, bent VME connector pins. Make sure the bottom of the metal-can crystal oscillators are clear of the decoupling capacitors on their sockets.

6.6 Software for Testing

A simple program to exercise the main functions of the sequencer can be provided. It runs on a Mac with Micron/MacVee VME interface. The following assumes that this is used.

6.7 Powering Up and Testing

Insert module in powered-down crate and power up. There are no power indicators on the module, check that there are volts downstream of the fuses.
Run the test program. It will ask for the NUBUS slot number and the Module Address, that is the value to which the address switch is set.
The program first tests the memory and reports if there are errors.
Next it loads a test sequence which exercises all channels.

6.7.1 Sequencer Settings

Select "Sequencer" from the File menu.This menu allow the selection of the clock speed (menu heading ClockSpeed),
full-width or not for the first 4 channels (menu heading sig 0:3), clock on or off (menu heading clock),
true or complement outputs for channels (4:19), (menu heading Polarity),
delays for channels (0:3) and 2 separate delays for the clocks (menu heading Sig Delays).
Set the speed to 50 MHz. (This is the current highest speed, earlier models were fitted with 67 MHz oscillators) The following timings are optimum for maximum speed of 50 MHz.
Set all channel outputs to TRUE by means of the polarity register. This is the test program default.
Set the first four channels to full-width output (default for test prog).
Set all programmable delays to zero.

6.7.2 Memory Timing

Check the relative timing of LATCH_CLK0 and the least significant bit of the memory address counter MA(0).
The rising edge of the clock should be 18 nS (+- 2 nS) after the change of MA(0) If the timing is outside this range then it should be adjusted by changing the wire_wrap link from pin 3 of PL4 to a different pin of PL3, thus selecting a different output of delay line DL5.

6.7.3 Timing Stability

The front panel Lemo sockets B29 and B30 should both have 25 ns NIM pulses with a rep rate of 780 Hz, corresponding to the sequence length of 64040 The jitter / short term stability of the clock should be checked by triggering a scope with the output of either B30 or B29 and looking at the same signal delayed by 1.28084 mS, ie the next occurrence of the pulse. The jitter (use infinite persistence mode on scope) should be less than +- 1 nS

6.7.4 Output Channels

Each channel should be 'scoped, (both pins) at the 50-pin front panel connector. Positive going ECL pulses should appear at the odd pins, negative on the even. The channels should have signals as indicated in fig 9.

6.7.5 Polarity

Check that the polarity register works by setting all bits of the polarity register to give inverted signals. Channels 4-19 should now be inverted, ie negative going pulses on odd pins.

6.7.6 RZ / NRZ

Check the RZ feature of channels 0-3. the width of the output pulses can be varied by changing the wire-wrap link from PL4 pin 2 to select another output of DL5. If the pulse width is 12 nS +- 4 then it is ok

6.7.7 programmable Delays

Check the operation of the programmable delay lines, 2 for the clocks and one each for channels 0-3. Select the menu “Sig Delays”. Each time one of the entries is selected its delay is increased by 2 nS, after 14 nS it wraps round to zero.

6.8 Triggered Mode

Exit from the Sequencer menu, by selecting “Main” from the “File” menu. Now select “File” and release on “free_run seq”. This loads a sequence for checking the interrupt, or triggered, mode of operation.
Connect a free runing oscillator, approx 1 KHz, with nim output, to the ”trigger” input of the sequencer.
Use B30 to trigger the scope, look at B29 and the oscillator output, with scope in infinite persistence mode. The oscillator trace should have 20 nS jitter, with no trace outside the 20 nS window.

6.9 Conclusion

The sequencer may now be considered “tested”.

SEQUENCER FRONT PANEL

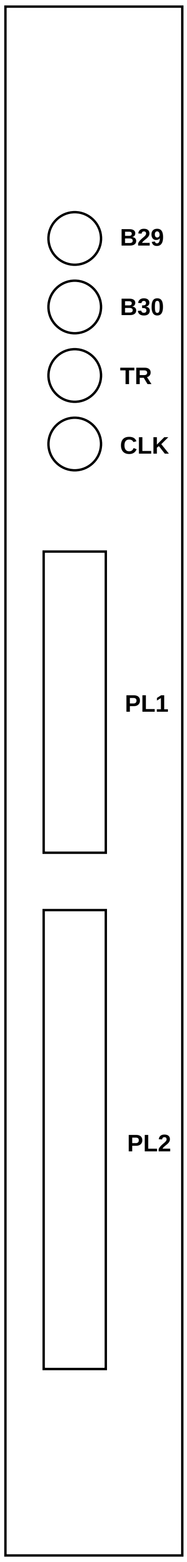


fig 1

seq_fig1

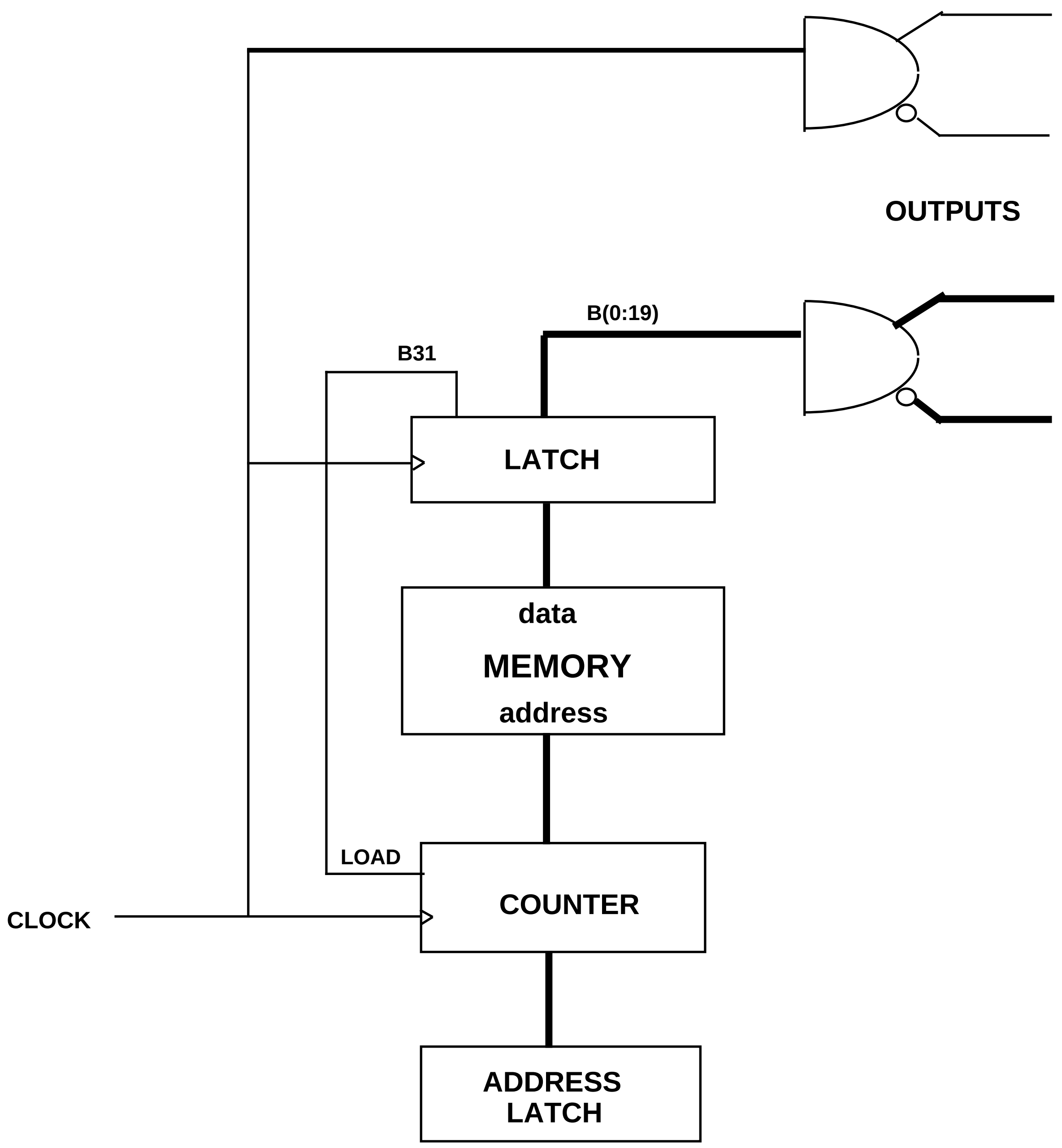


fig 2

seq_fig2

ADDRESS	DATA (HEX)
0	1
1	2
2	4
3	8
4	80000010
5	20
6	40
7	80

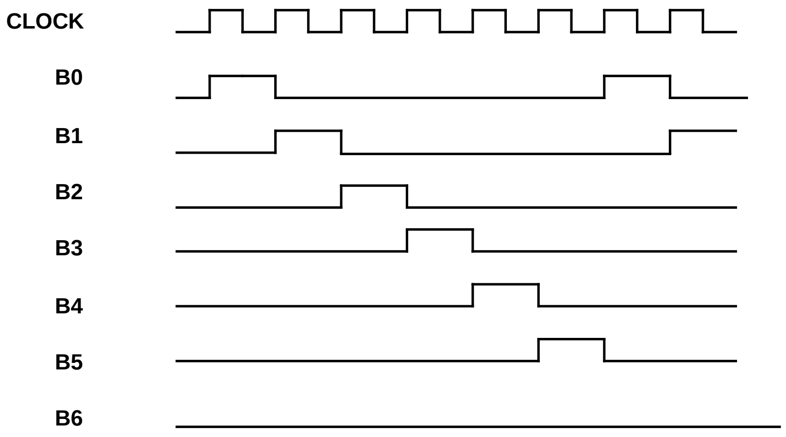


fig 3

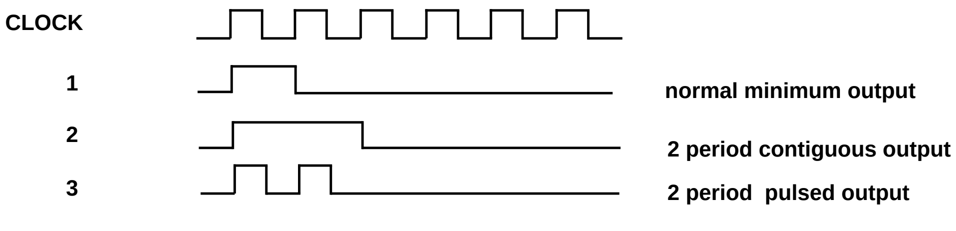


fig 4

seq_fig3_4

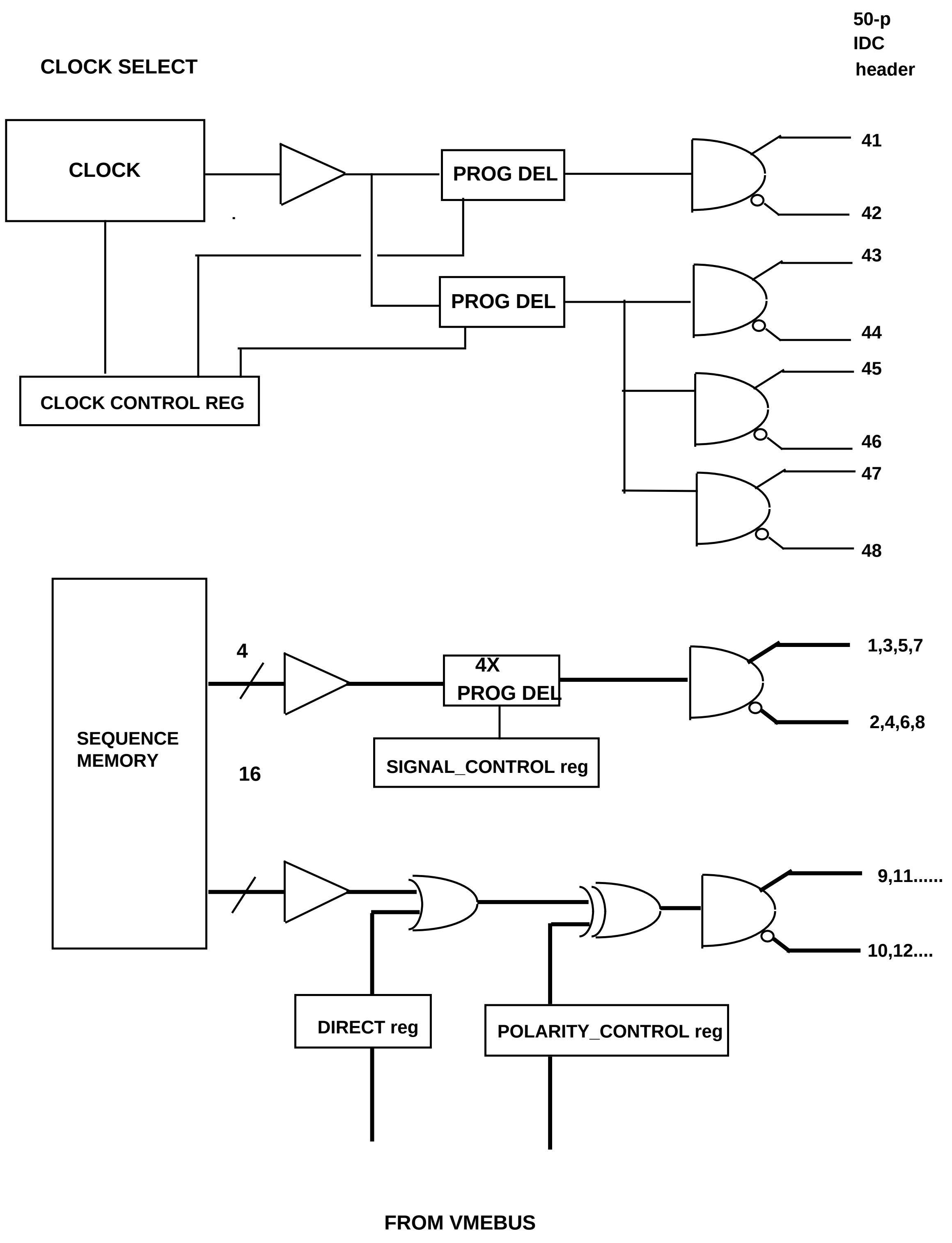


fig 5

seq_fig5

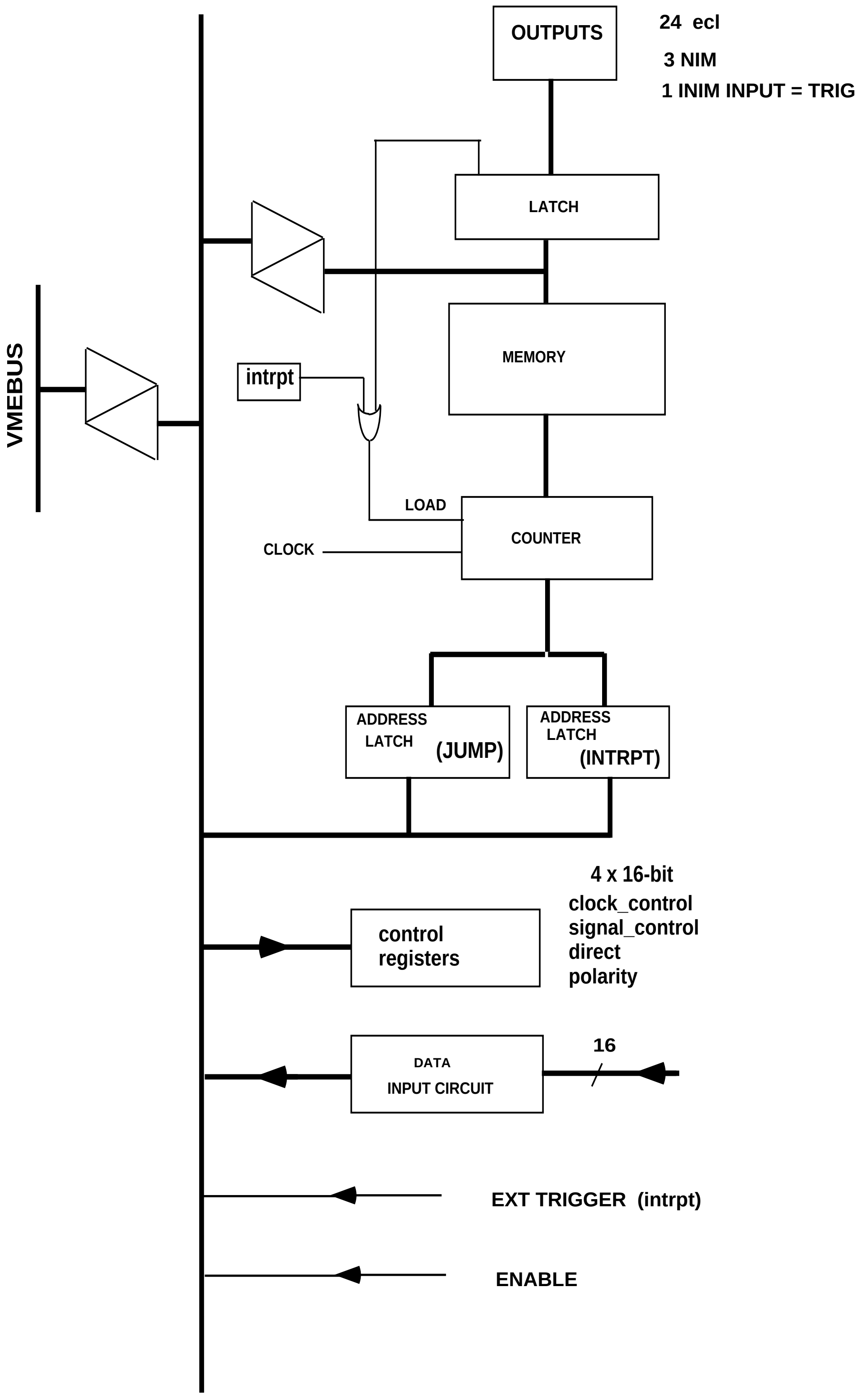


fig 6

seq_fig6

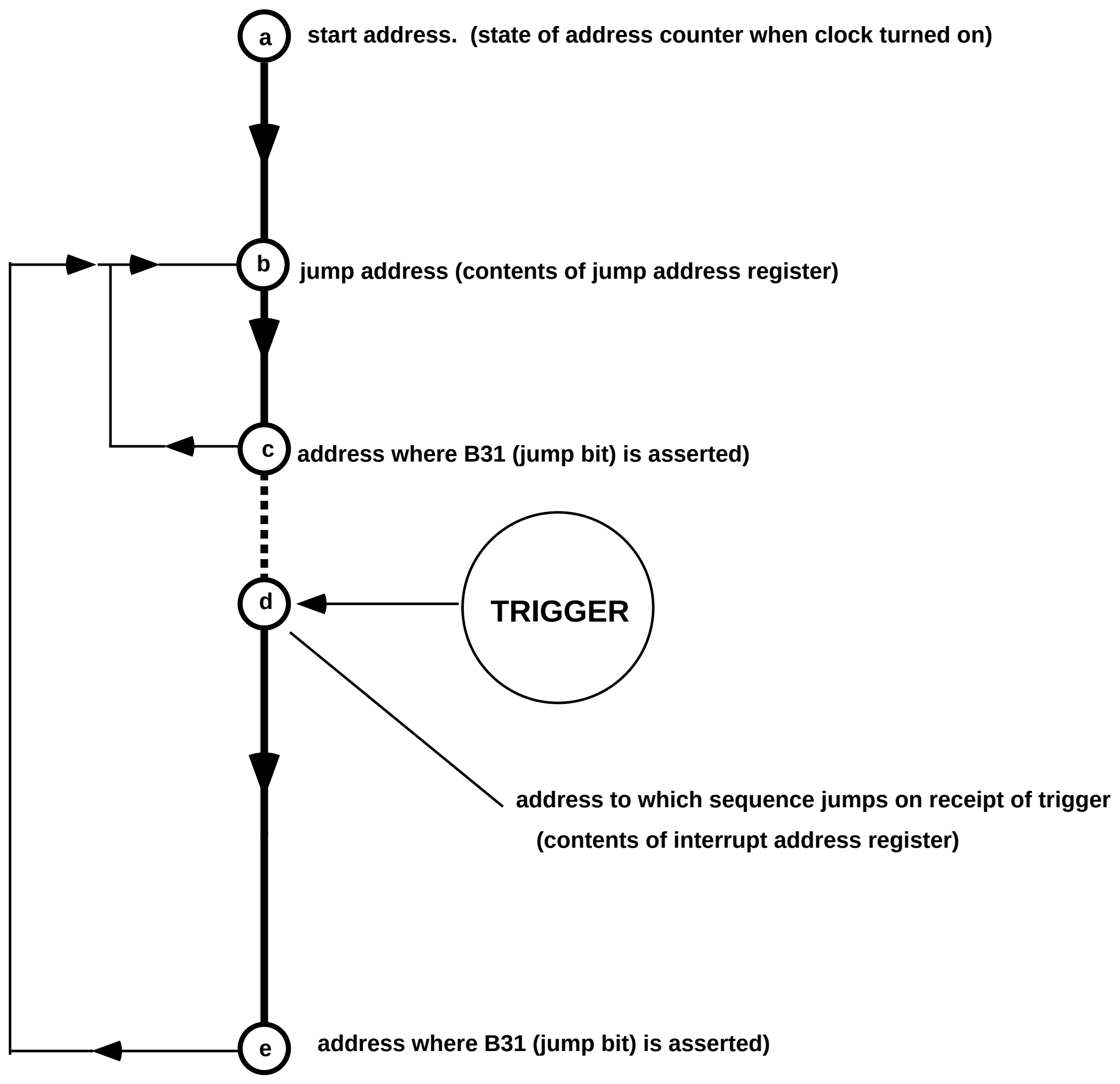

a
start address. (state of address counter when clock turned on)
b
jump address (contents of jump address register)
c
address where B31 (jump bit) is asserted)
d
TRIGGER
address to which sequence jumps on receipt of trigger
(contents of interrupt address register)
e
address where B31 (jump bit) is asserted)

fig 6a

seq_fig6a

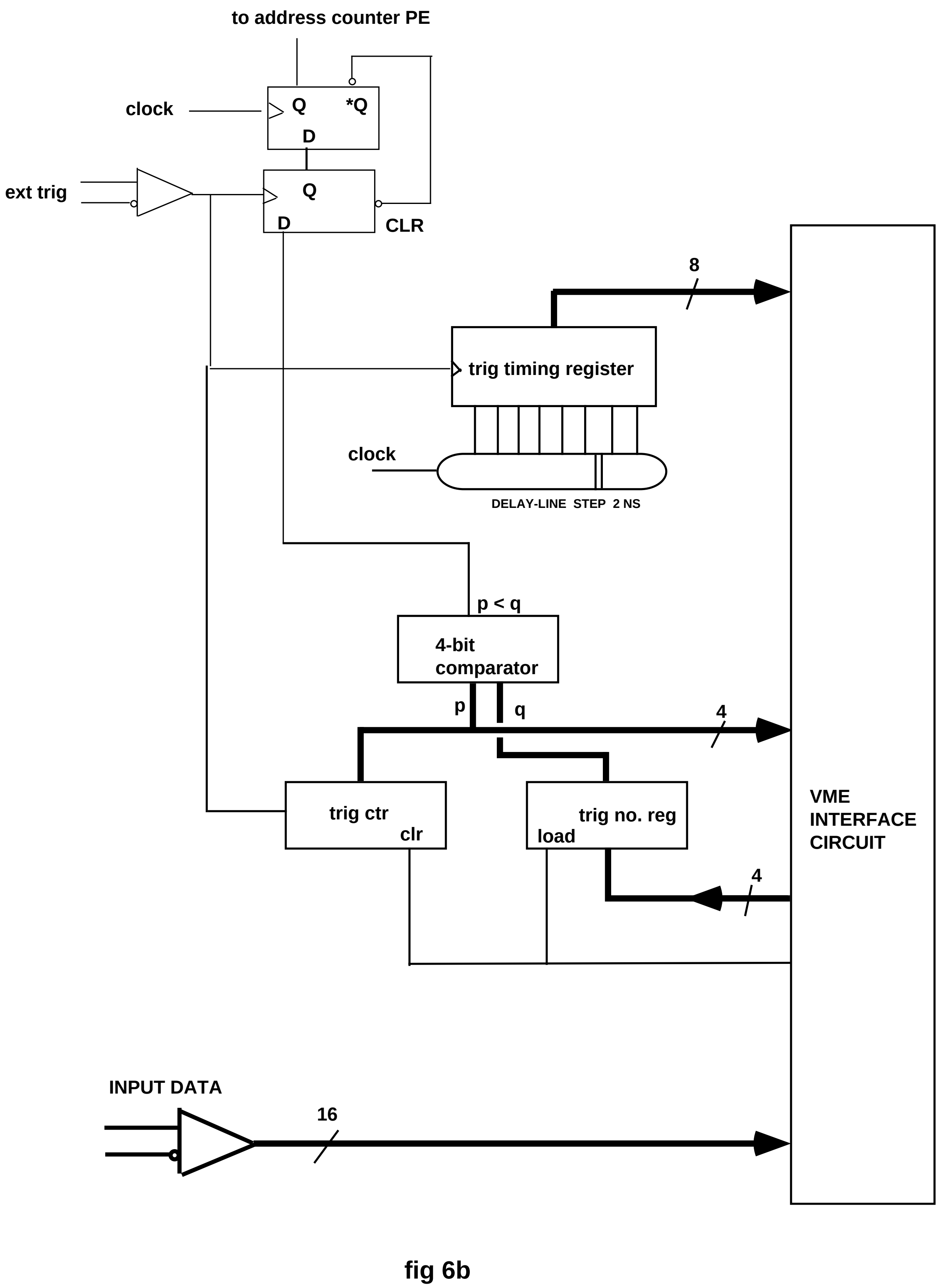
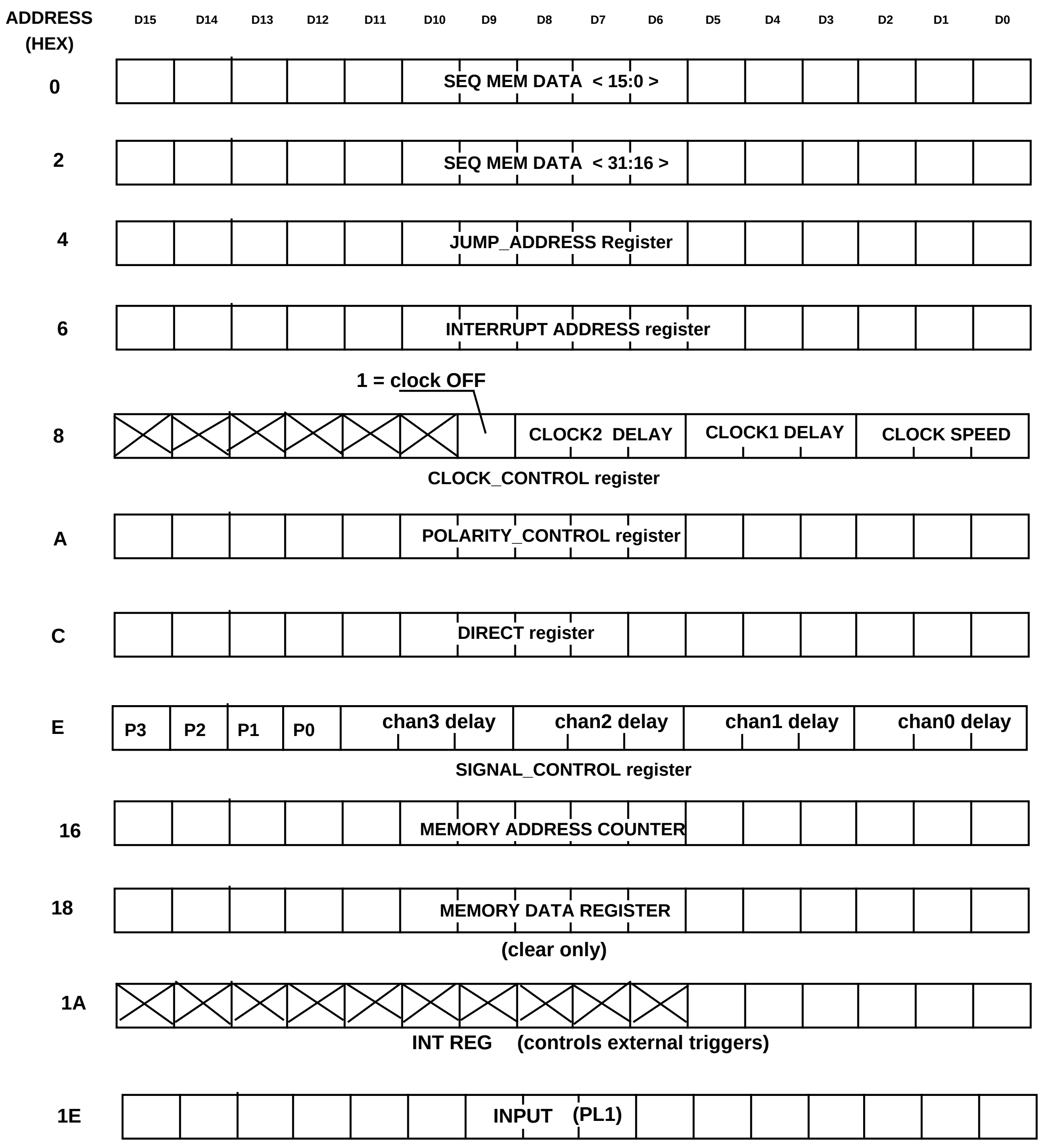


fig 6b

seq_fig6b

SEQUENCER MEMORY MAP



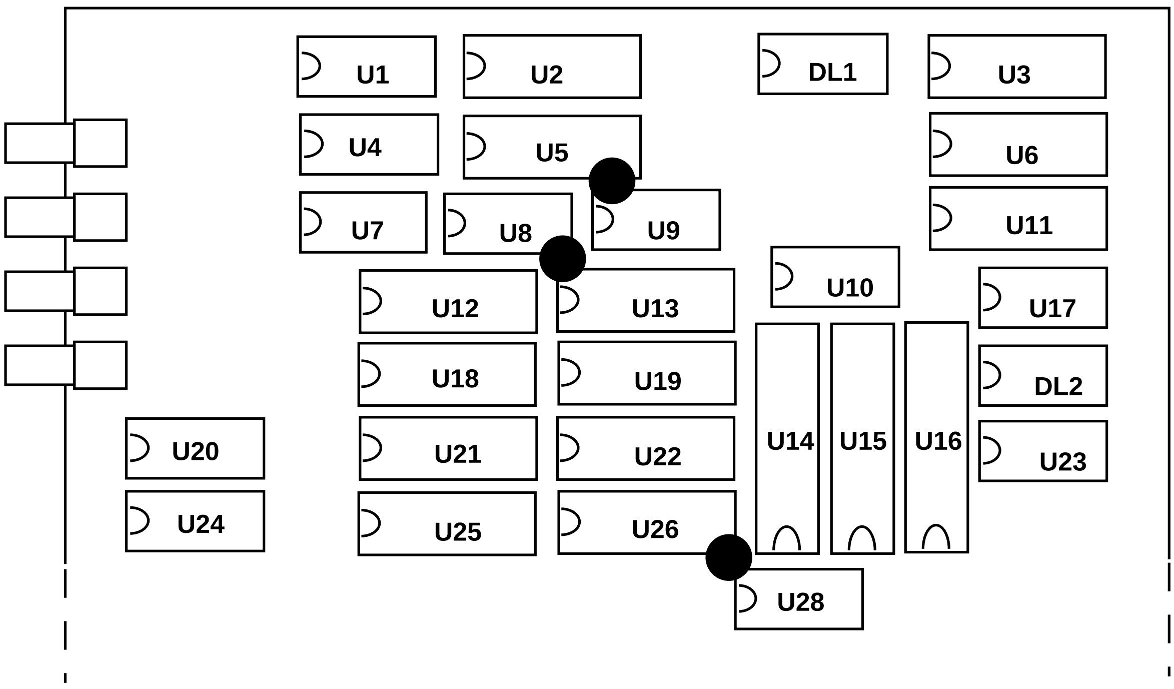
SEQUENCE MEMORY DATA <0:15> AND <16:31> are read / write.

INPUT is read only.

ALL OTHER registers are write only.

fig 7

seq_fig7



places to check for shorts between adjacent corner pins

fig 8

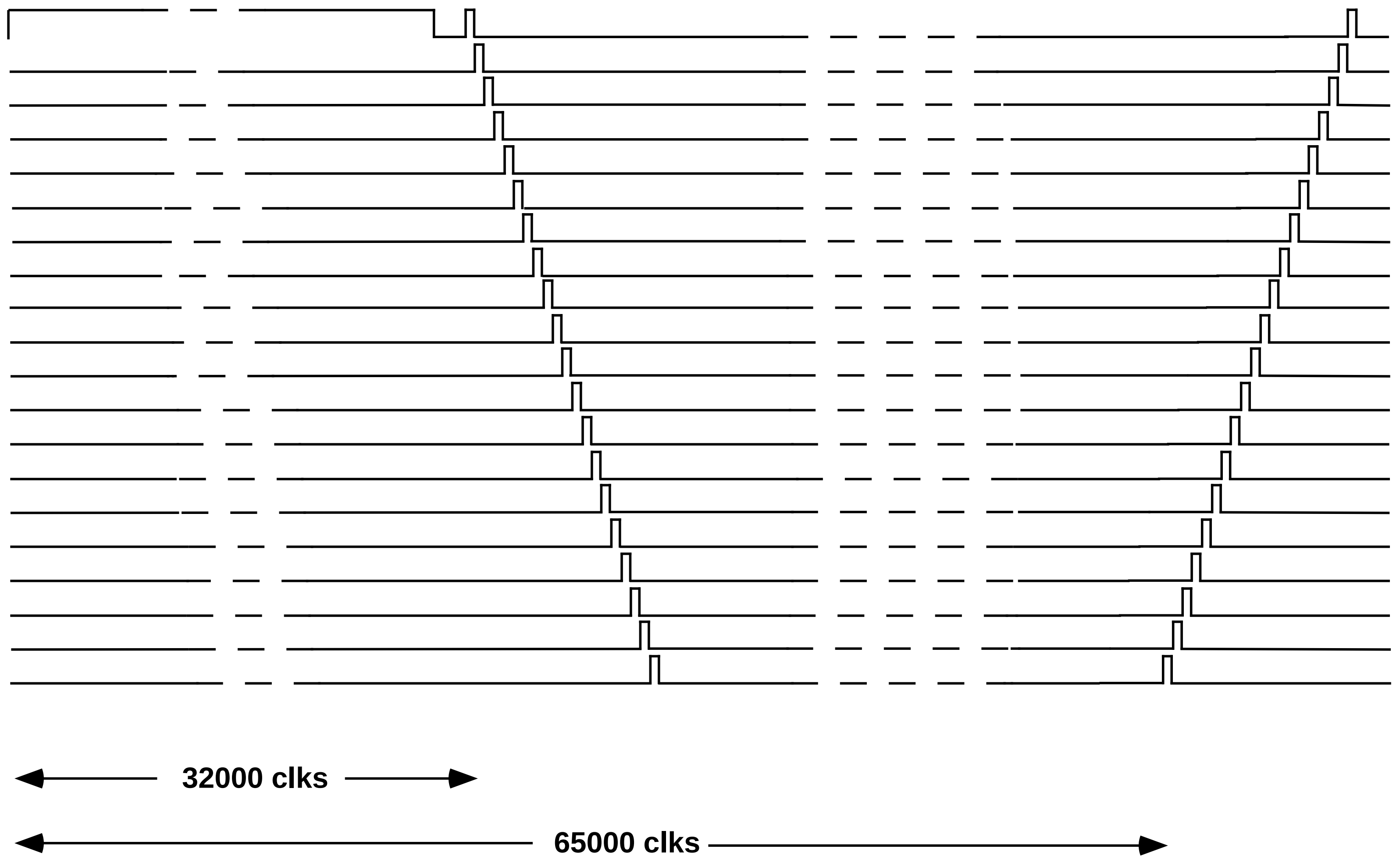


fig 9

seq_fig9