```
title "V107 RHIC RTDL timestamp module";
%drawing 93028635
 next assembly 94028632
 by C. R. Conkling Jr.
 using EPF10K10QC208
 revision July 22, 1997 changed interrupt trigger to !SELECT
 revision July 23, 1997 changed to PROM table for 60Hz decode
 revision December 5, 1997 changed RTDL output words to overlap 6-bits
                        changed many names to avoid confusion
                        added additional 720 Hz period test
                        changed interrupt to change-of-state
                        added additional output on PROM table for 15_HZ
 revision December 16, 1997 changed UNIX time counter to hold at a
                        count of -1 at rollover of 2^31 counts.

UNIX time is a positive 32-bit signed integer which allows a 31-bit count.
2^31, 1-second intervals approximately:
        = 2.1^9 seconds
        = 596000 hours
        = 24855 days
        = 68 years
UNIX time started January 1, 1970
counting P/S 720 Hz ticks:
1-second = 720 720 Hz ticks; 10-bits

RTDL UNIX TIME DATA FORMAT
|-----------|-----------|-----------|-----------|-----------|-----------|
|23|22|21|20|19|18|17|16|15|14|13|12|11|10|09|08|07|06|05|04|03|02|01|00|
|31---------------------------MSB UNIX TIME---------------------------08|
|13------------LSB UNIX TIME------------00|09------720 Hz ticks-------00|

event encoder/input module A16 addressing
|-----------|-----------|-----------|-----------|
|15|14|13|12|11|10|09|08|07|06|05|04|03|02|01|00|
-------------------------------------------------
|-----------SWR------------|                     |
                           | 0  X  X  X  X  X  X|R/- BYTE status/ID PROM
                           | 1  0  0  0  0  0  1|R/W BYTE CSR
                           | 1  0  0  0  0  0  1|R/W BYTE interrupt vector
                           | 1  0  0  0  0  1  1|R/W BYTE interrupt level
                           | 1  0  0  0  1  0  0|R/W BYTE seconds msb id
                           | 1  0  0  0  1  0  1|R/W BYTE seconds lsb id
                           | 1  0  0  1  0  0  0|R/W LWORD time - seconds%

include "lpm_rom"; %VMEid PROM%

subdesign 94028635

(/SYSRST, %VMEbus system reset%
 SYSCLK, %VMEbus 16 MHz clock%
 SWR[15..7], %base address comparison%
 BA[15..1], %VMEbus address lines%
 BAM[5..0], %VMEbus address type%
 /DS0, /DS1, /LWORD, %VMEbus address control%
 /AS, /IACK, /WRITE, /IACKIN, %VMEbus control%
```

```
 /RTDL_ASACK, %enable RTDL output%
 RTDL_PID[7..0], %RTDL parameter selection code%
 /POWER_SUPPLY_720 : input; %external 720Hz input%

 D[31..0] : bidir; %VME data bus%

 /DTACK, %acknowledge transfer%
 /IRQ[7..1], %select interrupt request level%
 /IACKOUT, %VMEbus interrupt control%
 RTDL[23..0], %RTDL data bus%
 /RTDL_ASACK_ACK, %acknowledge ASACK%
 720_HZ, 60_HZ, 15_HZ, 1_SECOND, %time pulse outputs based on 720 Hz counter%
 4_SECOND, 64_SECOND, %time pulse outputs based on seconds counter%
 /OFFLINE, %drives OFF LINE LED%
 /VME_SELECT, %drives VME SELECT LED%
 /SELECT, %drives 720 source LED%
 /60_HZ, /15_HZ, /1_SECOND, /4_SECOND, /64_SECOND, %drives pulse output LEDs%
 /LW_ENA, /HI_ENA, /LO_ENA : OUTPUT;) %module VMEbus data control%

variable

SHORT_AD, SWR_COMP, LINK, A[15..0], SECONDS_MSB_SELECT, SECONDS_LSB_SELECT,
SECONDS_ADDRESS, SECONDS_TRIGGER, SELECT_EXT_720, CHANGE, PRESENT : node;

SR[2..0], AL[6..0], MODULE, VME_SELECT[1..0], DISPLAY[15..0], CR[7..0],
INT_VEC[7..0], INT_LEV[7..0], SECONDS_MSB_ID[7..0], SECONDS_LSB_ID[7..0],
SECONDS[31..0], 720HZ[9..0], SYSCLK_DIVIDE[14..0], OS[5..0], 720_HZ_CLOCK,
COUNT[5..0], DDTACK, IRQX, LOCAL, IACKOUT, STATUS, IDTACK, INT_SR[2..0],
60_HZ_ONE_SHOT[1..0], 60_HZ, 1_SEC_ONE_SHOT[1..0], 4_SEC_ONE_SHOT[1..0],
64_SEC_ONE_SHOT[1..0], EXT_720_TEST[14..0], EXT_720_TEST_COUNT[14..0],
1_SECOND, 4_SECOND, 64_SECOND, 15_HZ, 15_HZ_ONE_SHOT[1..0] : dffe;

TRI_DATA[31..0] : tri;

PROM : lpm_rom with(lpm_width = 8,
                    lpm_numwords =  32,
                    lpm_widthad = 5,
                    lpm_file = "vmeid.mif",
                    lpm_address_control = "unregistered",
                    lpm_outdata = "unregistered");

DECODE : lpm_rom with(lpm_width = 2,
                    lpm_numwords =  1024,
                    lpm_widthad = 10,
                    lpm_file = "decode.mif",
                    lpm_address_control = "unregistered",
                    lpm_outdata = "unregistered");

begin
%-------------------------------------------------------------------------------%
%3-bit shift register, SR[2..0], clocked by 16 MHz SYSCLK to develop local
 data strobes, and VMEbus response, DTACK. SR[2..0] is held reset by DS[1..0].
 When a VMEbus cycle starts, DS[1..0] enable the shift register. At the
 following times:
 SR2 LATCH MODULE & ADDRESS 0-62.5ns after start
```

```
 SR1 MODULE to WRITE = 62.5ns write setup
 SR0 MODULE to DTACK = 125ns read access%
SR2.d = VCC;
SR[1..0].d = SR[2..1];
SR[2..0].clk = global(SYSCLK);
SR[2..0].clrn = /SYSRST & (!/DS0 # !/DS1);

%latch encoder module base address
 module will respond byte between BASE_ADDRESS and + 0x7F
 module will respond LWORD at BASE_ADDRESS + 0x48%
MODULE.d = SWR_COMP & SHORT_AD & /IACK & ((/LWORD & (/DS0 $ /DS1))
                                          # !/LWORD & A[6..0] == b"1001000");
MODULE.clk = SR[2];
MODULE.clrn = /SYSRST & (!/DS0 # !/DS1);

SWR_COMP = SWR[15..7] == A[15..7]; %encoder base address%
SHORT_AD = BAM[5..0] == B"101X01"; %VMEbus short address%

%Address latch%
A[15..0] = (BA[15..1], (/DS1 & !/DS0));
AL[6..0].d = (A[6..0]);
AL[].clk = SR[2];
AL[].clrn = global(/SYSRST);

%VMESEL LED driver output%
VME_SELECT[0].d = VCC;
VME_SELECT[0].clk = MODULE;
VME_SELECT[0].clrn = !VME_SELECT[1];
VME_SELECT[1].d = VME_SELECT[0];
VME_SELECT[1].clk = DISPLAY[15];
VME_SELECT[1].clrn = global(/SYSRST);

/VME_SELECT = !VME_SELECT[1];

%Low frequency clock for LED displays%
DISPLAY[15..0].clk = global(SYSCLK);
DISPLAY[].clrn = global(/SYSRST);
DISPLAY[] = DISPLAY[] + 1;

%-----------------------------------------------------------------------------%
%Acknowledge a VMEbus read/write cycle%
DDTACK.d = MODULE;
DDTACK.clk = SR[0];
DDTACK.clrn = /SYSRST & (!/DS0 # !/DS1);

%VMEbus DTACK is a logic OR of read/write and interrupt acknowledgements%
/DTACK = !(DDTACK # IDTACK);

%-----------------------------------------------------------------------------%
%Module registers%
%VME ID prom contained in EPLD%
PROM.address[4..0] = AL[5..1];

%Command register. Status bit CR[7..6], read only,
 generate interrupt on change-of-state%
```

```
CR[7].d = SELECT_EXT_720;
CR[7].clk = CHANGE;
CR[6].d = PRESENT;
CR[6].clk = CHANGE;

CHANGE = /SYSRST & (SELECT_EXT_720, PRESENT) != CR[7..6];

CR[5..0].d = D[5..0];
CR[5..0].clk = MODULE & (AL[6..0] == h"41") & SR[1] & !/WRITE;
CR[7..0].clrn = global(/SYSRST);

%drives OFF LINE LED%
/OFFLINE = CR[0];

%Module interrupt status/ID register. The interrupt status/ID register
 contains the interrupt vector which is returned during an interrupt cycle.%
INT_VEC[7..0].d = D[15..8];
INT_VEC[7..0].clk = MODULE & (AL[6..0] == h"42") & SR[1] & !/WRITE;
INT_VEC[7..0].clrn = global(/SYSRST);

%Module interrupt request level register. The interrupt request is made
 on the level selected in this register%
INT_LEV[7..0].d = D[7..0];
INT_LEV[7..0].clk = MODULE & (AL[6..0] == h"43") & SR[1] & !/WRITE;
INT_LEV[7..0].clrn = global(/SYSRST);

%generate seven interrupt request level outputs%
for LEVEL in 7 to 1 generate
/IRQ[LEVEL] = !(INT_LEV[2..0] == LEVEL & IRQX);
end generate;

%SECONDS MSB RTDL parameter id%
SECONDS_MSB_ID[7..0].d = D[15..8];
SECONDS_MSB_ID[7..0].clk = MODULE & (AL[6..0] == h"44") & SR[1] & !/WRITE;
SECONDS_MSB_ID[7..0].clrn = global(/SYSRST);

if !CR[0] then SECONDS_MSB_SELECT = GND;
   elsif SECONDS_MSB_ID[] == RTDL_PID[7..0] then SECONDS_MSB_SELECT = VCC;
   else SECONDS_MSB_SELECT = GND;
end if;

%SECONDS LSB RTDL parameter id%
SECONDS_LSB_ID[7..0].d = D[7..0];
SECONDS_LSB_ID[7..0].clk = MODULE & (AL[6..0] == h"45") & SR[1] & !/WRITE;
SECONDS_LSB_ID[7..0].clrn = global(/SYSRST);

if !CR[0] then SECONDS_LSB_SELECT = GND;
   elsif SECONDS_LSB_ID[] == RTDL_PID[7..0] then SECONDS_LSB_SELECT = VCC;
   else SECONDS_LSB_SELECT = GND;
end if;

%-----------------------------------------------------------------------------%
%24-bit T/S RTDL data bus. RTDL words overlap 6-bits for uncertancy%
case (SECONDS_LSB_SELECT) is
when b"0" => RTDL[23..0] = SECONDS[31..8];
```

```
when b"1" => RTDL[23..0] = (SECONDS[13..0], 720HZ[9..0]);
end case;

%Acknowledge RTDL transfer%
/RTDL_ASACK_ACK = !(!/RTDL_ASACK & (SECONDS_MSB_SELECT # SECONDS_LSB_SELECT));

%-----------------------------------------------------------------------------%
%VMEbus interrupt logic%

%IRQX is set to request an interrupt%
IRQX.d = VCC;
IRQX.clk = CHANGE;
IRQX.clrn = !IDTACK;

%LOCAL is set by any following DS0%
LOCAL.d = IRQX;
LOCAL.clk = !/DS0;
LOCAL.clrn = global(/SYSRST);

%IACKOUT is set if the interrupt is not local or not level%
IACKOUT.d = !LOCAL # (AL[3..1] != INT_LEV[2..0]);
IACKOUT.clk = INT_SR0;
IACKOUT.clrn = !/AS;
/IACKOUT = !IACKOUT;

%Enable interrupt status/ID byte%
STATUS.d = LOCAL & (AL[3..1] == INT_LEV[2..0]);
STATUS.clk = INT_SR1;
STATUS.clrn = !/DS0;

%Interrupt cycle acknowledgement%
IDTACK.d = LOCAL & (AL[3..1] == INT_LEV[2..0]);
IDTACK.clk = INT_SR0;
IDTACK.clrn = !/DS0;

%3-bit shift register, clocked by 18_MHz to develop VMEbus interrupt delays%
INT_SR2.d = !/IACKIN;
INT_SR[1..0].d = INT_SR[2..1].q;
INT_SR[2..0].clk = global(SYSCLK);
INT_SR[2..0].clrn = !/DS0;

%-----------------------------------------------------------------------------%
%32-bit T/S local VMEbus data bus for data, VME status/ID message, or
 interrupt vector. VME status/ID odd byte read internal PROM, even byte = ".",
 0x2E tri-state local VME data bus; odd data bus output is also controlled by
 the interrupt STATUS flip/flop%

TRI_DATA[31..16].in = (SECONDS[31..16]);

case (AL[6..0]) is %even byte addresses and LWORD%
 when b"0xxxxxx" => TRI_DATA[15..8].in =
                        (GND, GND, VCC, GND, VCC, VCC, VCC, GND);
 when b"100001x" => TRI_DATA[15..8].in = INT_VEC[7..0];
 when b"100010x" => TRI_DATA[15..8].in = SECONDS_LSB_ID[7..0];
 when b"100100x" => TRI_DATA[15..8].in = SECONDS[15..8];
```

```
 when others => TRI_DATA[15..8].in = (GND, GND, GND, GND, GND, GND, GND, GND);
end case;

case (STATUS, AL[6..0]) is %odd byte addresses%
 when b"00xxxxxx" => TRI_DATA[7..0].in = PROM.q[7..0];
 when b"0100000x" => TRI_DATA[7..0].in =
                  (SELECT_EXT_720, GND, GND, GND, GND, GND, CR[1..0]);
 when b"0100001x" => TRI_DATA[7..0].in =
                  (GND, GND, GND, GND, GND, INT_LEV[2..0]);
 when b"0100010x" => TRI_DATA[7..0].in = SECONDS_LSB_ID[7..0];
 when b"0100100x" => TRI_DATA[7..0].in = SECONDS[7..0];
 when b"1xxxxxxx" => TRI_DATA[7..0].in = INT_VEC[7..0];
 when others => TRI_DATA[7..0].in = (GND, GND, GND, GND, GND, GND, GND, GND);
end case;

%internal register connections to tri-state VMEbus data%
TRI_DATA[31..0].oe = STATUS # MODULE & /WRITE;
D[31..0] = TRI_DATA[31..0].out;

%VMEbus data bus buffer enables%
/LW_ENA = !(MODULE & !/LWORD);
/HI_ENA = !(MODULE & (!/DS1 & /DS0 # !/LWORD));
/LO_ENA = !(MODULE & (/DS1 & !/DS0 # !/LWORD # STATUS));

%-----------------------------------------------------------------------------%
%Select clock source local vs external 720 Hz input.%
%synchronize external 720 Hz to SYSCLK.%
COUNT[0].d = VCC;
COUNT[0].clk = !/POWER_SUPPLY_720;
COUNT[0].clrn = !COUNT[1];
COUNT[1].d = COUNT[0];
COUNT[1].clk = global(SYSCLK);
COUNT[1].clrn = global(/SYSRST);

%gross test external 720 Hz clock presence%
COUNT[5..2].clk = SYSCLK_DIVIDE[14];
COUNT[5..2].clrn = /SYSRST & !COUNT[1];

if COUNT[5..2] == 15 then COUNT[5..2] = 15;
else COUNT[5..2] = COUNT[5..2] + 1;
end if;

PRESENT = COUNT[5..2] < 15;

%fine test external 720 Hz clock frequency%
EXT_720_TEST[14..0] = EXT_720_TEST[14..0] + 1;
EXT_720_TEST[14..0].clk = global(SYSCLK);
EXT_720_TEST[14..0].clrn = !COUNT[1] & /SYSRST;

EXT_720_TEST_COUNT[14..0].d = EXT_720_TEST[14..0];
EXT_720_TEST_COUNT[14..0].clk = COUNT[1];
EXT_720_TEST_COUNT[14..0].clrn = global(/SYSRST);

%Select external 720_HZ clock if present and within 1 percent of 720 Hz%
if PRESENT then
```

```
   if (EXT_720_TEST_COUNT[14..0] < 22000
     # EXT_720_TEST_COUNT[14..0] > 22444)
     then SELECT_EXT_720 = GND;
   else SELECT_EXT_720 = VCC;
   end if;
else SELECT_EXT_720 = GND;
end if;

%/SELECT_EXT_720 drives front panel external 720 input LED%
/SELECT = SELECT_EXT_720;

%divide 16 MHz SYSCLK to 720 Hz in case POWER_SUPPLY_720 fails%
SYSCLK_DIVIDE[14..0].clk = global(SYSCLK);
SYSCLK_DIVIDE[14..0].clrn = global(/SYSRST);

if SYSCLK_DIVIDE[] == 22222 then SYSCLK_DIVIDE[]= 0;
   else SYSCLK_DIVIDE[] = SYSCLK_DIVIDE[] + 1;
end if;

%-----------------------------------------------------------------------------%
%Stretch leading edge of external 720_Hz or divider to 1 usec%
OS[0].d = VCC;
OS[0].clk = (!/POWER_SUPPLY_720 & SELECT_EXT_720) # (SYSCLK_DIVIDE[14] &
!SELECT_EXT_720);
OS[0].clrn = !OS[1];
OS[1].d = OS[0];
OS[1].clk = global(SYSCLK);
OS[1].clrn = /SYSRST;

%1 usec one-shot%
OS[5..2].clk = global(SYSCLK);
OS[5..2].clrn = global(/SYSRST);

if OS[1] then OS[5..2] = OS[5..2] + 1;
   elsif OS[5..2] != 0 then OS[5..2] = OS[5..2] + 1;
   else OS[5..2] = 0;
end if;

720_HZ_CLOCK.d = OS[5..2] != 0;
720_HZ_CLOCK.clk = global(SYSCLK);
720_HZ_CLOCK.clrn = global(/SYSRST);

%-----------------------------------------------------------------------------%
%720HZ[9..0] counts 720 Hz clock. 720_HZ_CLOCK normally be derived from the
 external POWER_SUPPLY_720. If the external input fails the 720Hz is derived
 from SYSCLK. The register is reset by a VMEbus write to SECONDS[31..0].%
720HZ[9..0].clk = 720_HZ_CLOCK;
720HZ[9..0].clrn= /SYSRST & !SECONDS_TRIGGER;

if 720HZ[] == 719 then 720HZ[] = 0;
   else 720HZ[] = 720HZ[] + 1;
end if;

%SECONDS[31..0] tabulates UNIX time in seconds. The register will normally be
 preset to UNIX time. SECONDS[31..0] counts 1 second ticks derived from the
```

```
 720HZ[9..0] counter%
SECONDS[31..0].clk = 720_HZ_CLOCK # SECONDS_TRIGGER;
SECONDS[31..0].clrn = global(/SYSRST);

if SECONDS_ADDRESS then SECONDS[31..0] = D[31..0];
   elsif SECONDS[31] then SECONDS[] = h"ffffffff";
   elsif 720HZ[] == 719 then SECONDS[] = SECONDS[] + 1;
   else SECONDS[] = SECONDS[];
end if;

SECONDS_ADDRESS = MODULE & (AL[6..0] == h"48") & !/LWORD & !/WRITE;
SECONDS_TRIGGER = MODULE & (AL[6..0] == h"48") & SR[1] & !/LWORD & !/WRITE;

%-----------------------------------------------------------------------------%
%Front panel pulse outputs%
720_HZ = 720_HZ_CLOCK;

%SRAM section used as a 1024x2 PROM. The programming file is decode.mif.
 decode.mif decodes the outputs of the 720HZ[] counter%
DECODE.address[9..0] = 720HZ[9..0];

60_HZ.d = DECODE.q[0];
60_HZ.clk = 720_HZ_CLOCK;
60_HZ.clrn = global(/SYSRST);

15_HZ.d = DECODE.q[1];
15_HZ.clk = 720_HZ_CLOCK;
15_HZ.clrn = global(/SYSRST);

%Streatch 60_HZ to drive both 720 & 60 front panel LEDs%
60_HZ_ONE_SHOT[0].d = VCC;
60_HZ_ONE_SHOT[0].clk = 60_HZ;
60_HZ_ONE_SHOT[0].clrn = !60_HZ_ONE_SHOT[1];
60_HZ_ONE_SHOT[1].d = 60_HZ_ONE_SHOT[0];
60_HZ_ONE_SHOT[1].clk =  DISPLAY[15];
60_HZ_ONE_SHOT[1].clrn = global(/SYSRST);
/60_HZ = !60_HZ_ONE_SHOT[1];

%Streatch 15_HZ to drive front panel LED%
15_HZ_ONE_SHOT[0].d = VCC;
15_HZ_ONE_SHOT[0].clk = 15_HZ;
15_HZ_ONE_SHOT[0].clrn = !15_HZ_ONE_SHOT[1];
15_HZ_ONE_SHOT[1].d = 15_HZ_ONE_SHOT[0];
15_HZ_ONE_SHOT[1].clk =  DISPLAY[15];
15_HZ_ONE_SHOT[1].clrn = global(/SYSRST);
/15_HZ = !15_HZ_ONE_SHOT[1];

%Generate 1, 4, and 64 second output = 720 clock.
 Streatch to 4 ms for display LEDs%
1_SECOND.d = 720HZ[9..0] == 719;
1_SECOND.clk = 720_HZ_CLOCK;
1_SECOND.clrn = global(/SYSRST);

1_SEC_ONE_SHOT[0].d = VCC;
1_SEC_ONE_SHOT[0].ena = 720HZ[9..0] == 719;
```

```
1_SEC_ONE_SHOT[0].clk = 720_HZ_CLOCK;
1_SEC_ONE_SHOT[0].clrn = !1_SEC_ONE_SHOT[1];
1_SEC_ONE_SHOT[1].d = 1_SEC_ONE_SHOT[0];
1_SEC_ONE_SHOT[1].clk = DISPLAY[15];
1_SEC_ONE_SHOT[1].clrn = global(/SYSRST);

4_SECOND.ena = 720HZ[9..0] == 719;
4_SECOND.d = SECONDS[1..0] == 3;
4_SECOND.clk = 720_HZ_CLOCK;
4_SECOND.clrn = global(/SYSRST);

4_SEC_ONE_SHOT[0].d = SECONDS[1..0] == 3;
4_SEC_ONE_SHOT[0].ena = 720HZ[9..0] == 719;
4_SEC_ONE_SHOT[0].clk = 720_HZ_CLOCK;
4_SEC_ONE_SHOT[0].clrn = !4_SEC_ONE_SHOT[1];
4_SEC_ONE_SHOT[1].d = 4_SEC_ONE_SHOT[0];
4_SEC_ONE_SHOT[1].clk = DISPLAY[15];
4_SEC_ONE_SHOT[1].clrn = global(/SYSRST);

64_SECOND.ena = 720HZ[9..0] == 719;
64_SECOND.d = SECONDS[5..0] == 63;
64_SECOND.clk = 720_HZ_CLOCK;
64_SECOND.clrn = global(/SYSRST);

64_SEC_ONE_SHOT[0].d = SECONDS[5..0] == 63;
64_SEC_ONE_SHOT[0].ena = 720HZ[9..0] == 719;
64_SEC_ONE_SHOT[0].clk = 720_HZ_CLOCK;
64_SEC_ONE_SHOT[0].clrn = !64_SEC_ONE_SHOT[1];
64_SEC_ONE_SHOT[1].d = 64_SEC_ONE_SHOT[0];
64_SEC_ONE_SHOT[1].clk = DISPLAY[15];
64_SEC_ONE_SHOT[1].clrn = global(/SYSRST);

%drives front panel 1, 4, 64  LED%
/1_SECOND = !1_SEC_ONE_SHOT[1];
/4_SECOND = !4_SEC_ONE_SHOT[1];
/64_SECOND = !64_SEC_ONE_SHOT[1];

end;
```