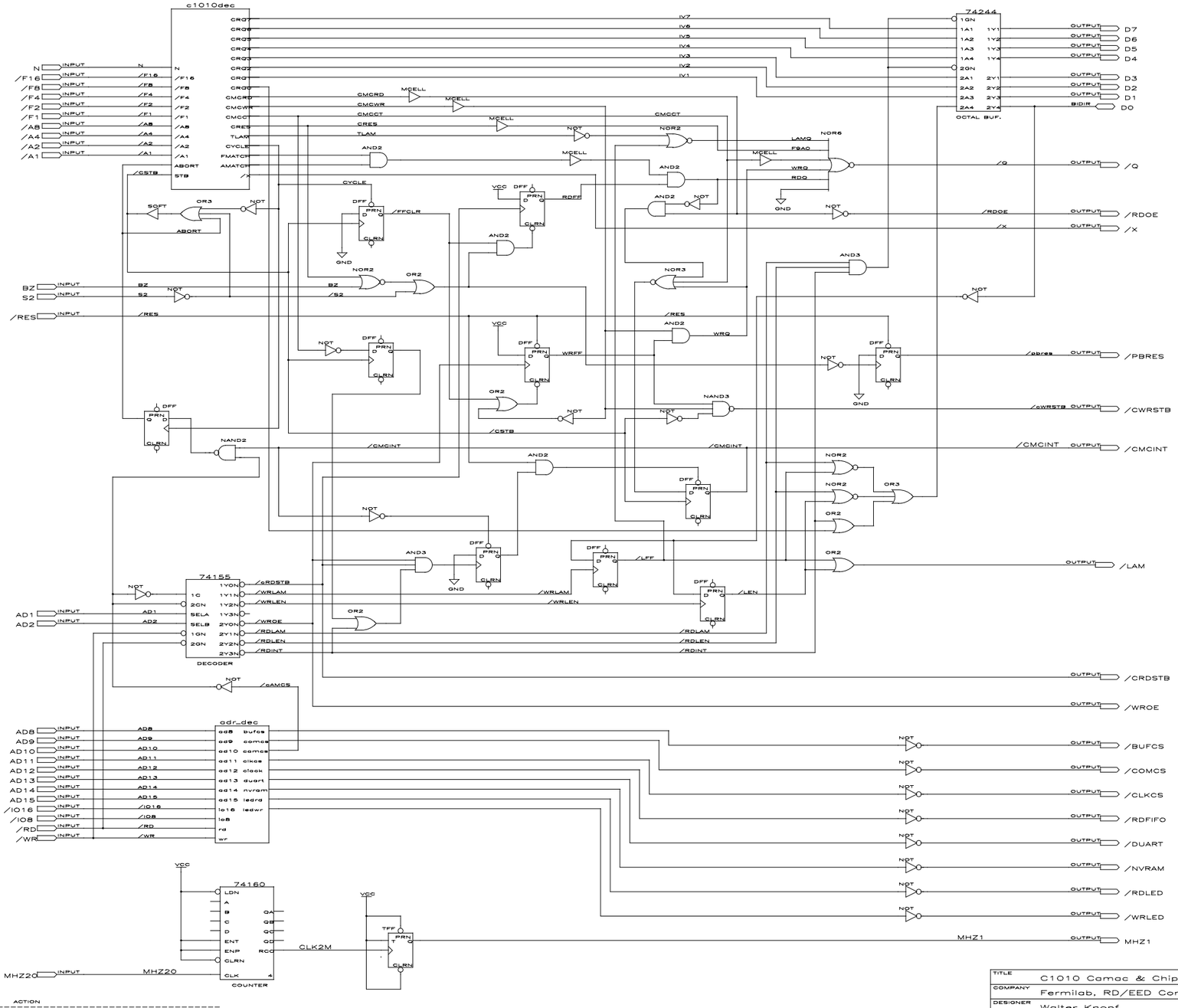
```
module _c1010u8
title 'CAMAC C1010 Viking I/O Control
mjk
02-Jul-1991
Checksum: 0000'

c1010u8 device 'p22v10';

" Inputs
dlyads,wr,rd            pin 1,2,3;
extcs                   pin 4;
ad9,ad10,ad11           pin 6,7,8;

" Wait control
xack,ready              pin 11,20;

" External Interrupt control
xint,ioint              pin 13,19;

" DMA Control
dreq0,dack0             pin 14,5;

" Outputs to Viking connector
xenb,xads,adoe,xds      pin 15,16,17,21;
xsel                    pin 22;

" Power and ground (for doc file)
VCC,GND                 pin 24,12;

" Macros
x = .x.;

address = [ad11,ad10,ad9,x,x,x,x,x,x,x,x,x];
equations

ioint   = 1;            " Don't pass through nonexistant interrupts
ready   = 0;            " We're always ready
dreq0   = 1;            " No DMA yet

!xsel   = !extcs & (address == ^h200);
!xenb   = !extcs & (address == ^h200);
!xds    = !extcs & (address == ^h200) & (!rd # !wr);

!adoe   = !extcs & (address == ^h200);
xads    = 1;            " Trans. high, latched low

end _c1010u8
```

```
module _c1010u24
title 'Tevetron Clock decode controller
mjk
09-Jul-1991
Checksum: 4AB2'

c1010u24 device 'p16v8r';

" Output clock
sclk                            pin 1;

" I/O read and write strobes
iord,iowr                       pin 7,8;

" Select lines for FIFO and mask RAM from U10
fiford,clkcs                    pin 5,9;

" Status lines related to TCLK decode
clkdav,mskout,perr              pin 3,2,4;

" Data bit 0 (mask value)
d0                              pin 13;

" FIFO Output Ready
fifoor                          pin 6;

" FIFO output enable, shift out, shift in, and shift in flip-flop
clken,rdclk,fifosi,siff         pin 12,14,15,18;

" Mask RAM Write strobe, data buffer enable, and data bit (d0)
mskwr,mskdbe,mskin              pin 19,17,16;

" VCC and GND lines for DOC file
vcc,gnd,GND                     pin 20,10,11;

equations

d0.oe    = !clkcs & !iord;              "Enable D0 when reading Mask values"
!d0      = !mskout;                     "No inversion, just buffering"
!mskin   = !d0;                         "Same here"

!clken   = !fiford & !iord & !fifoor    "FIFO OE (Data of FF means no event)"
         # !clken & !fiford & !iord;

rdclk    = !fiford & !iord;             "FIFO SO"

fifosi   = clkdav & mskout & siff;      "Write to FIFO on unmasked event"
!siff   := clkdav;                      " Use SCLK for SI pulse"

!mskwr   = !clkcs & !iowr;              "Write to mask RAM"

" MSKDBE selects whether CPU Address lines or the decoded clock event"
" addresses the mask RAM"
" If the processor wants to write to the mask RAM when there is a valid event"
" decoded, we have a problem. Tune in tomorrow for the dramatic conclusion!!!"

!mskdbe = !clkcs;

fuses "User signature word
        [2056..2071] = 'C1';
        [2072..2087] = '15';
        [2088..2103] = '1U';
        [2104..2119] = '22';

end _c1010u24
```

```
module _C1010U36

title 'Write Strobe Generator'

"Fermilab RD/EED, Walter Knopf, 1990

C1010U36 device 'P22V10';

"INPUT

" From 80906CA CPU
be0,be1,be2,be3         pin 8,9,10,11;  " BYTE ENABLEs
pclk                    pin 1;          " processor clock
wait                    pin 2;          " low when wait active
den                     pin 5;          " low during active data cycle
w_r                     pin 7;          " read/write status, low during READ
blast                   pin 13;         " last byte, low during last data cycle
ram                     pin 6;          " RAM select line
io8                     pin 3;          " 8-bit i/o space selected
io16                    pin 4;          " 16-bit memory or I/O space selected

"OUTPUT
wr0, wr1, wr2, wr3      pin 17,16,15,14; " active low RAM write strobes
wr,rd                   pin 18,19;      " Write and read strobes
ble, bhe                pin 22,23;      " byte low/high enable for 16-bit space

@alternate

EQUATIONS

/wr0    = /ram * w_r * /den * /wait * /be0;

/wr1    = /ram * w_r * /den * /wait * /be1;

/wr2    = /ram * w_r * /den * /wait * /be2;

/wr3    = /ram * w_r * /den * /wait * /be3;

/wr     =  w_r * /den * /wait;
/rd     = /w_r * /den * /wait;

/ble    = /io16 * (/be0 + /be2);

/bhe    = /io16 * (/be1 + /be3);

END _C1010U36
```

```
module _c1010u38
title 'C1010 address decode
mjk
24-Nov-1992
checksum: ????'

" We are making a transparent latch out of this chip
" Transparent when ads is low, latch with ads high

c1010u38        device 'p22v10';

" Inputs
a31,a30,a29,a28         pin     11,10,9,8;      " Address lines
a23,a22,a21,a20         pin     7,6,5,4;
a19,a17                 pin     3,2;
ads,w_r                 pin     1,23;   " Address Strobe & Write/Read line

" Outputs
io8,io16                pin     22,21;  " 8 and 16 bit address space
ram                     pin     20;     " RAM bank A or B selected
extcs                   pin     19;     " Viking I/O select
rama,ramb               pin     18,16;  " RAM bank A & B selects
ramoe                   pin     17;     " RAM output enable
roma,romb               pin     15,14;  " ROM bank A & B selects

" define pin configuration
a31,a30,a29,a28         istype  'com';
a23,a22,a21,a20         istype  'com';
a19,a17                 istype  'com';
ads,w_r                 istype  'com';
ram,rama,ramb,ramoe     istype  'com';
io8,io16,roma,romb      istype  'com';

x = .x.;
address = [a31,a30,a29,a28,x,x,x,x,a23,a22,a21,a20,a19,x,a17,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x];

equations

" Transparent when ads is low, latch with ads high

" 27C010 EPROM bank A                    address: FFFE00000 - FFFFFFFF
!roma   = (address == ^hFFFE0000)
        # (address >= ^hFFFE0000) & (address <= ^hFFFFFFFF) & !ads
        # (!roma & ads);

" 27C010 EPROM bank B                    address: FFFC0000 - FFFDFFFF
!romb   = (address == ^hFFFC0000)
        # (address >= ^hFFFC0000) & (address <= ^hFFFDFFFF) & !ads
        # (!romb & ads);

" RAM select (Bank A or B)
!ram    = (address >= ^hE0000000) & (address <= ^hE00FFFFF) & !ads
        # (!ram & ads);

" STATIC RAM bank A                     address: E0000000 - E017FFFF
!rama   = (address >= ^hE0000000) & (address <= ^hE001FFFF) & !ads
        # (!rama & ads);

" STATIC RAM bank B                     address: E0020000 - E003FFFF
!ramb   = (address >= ^hE0020000) & (address <= ^hE003FFFF) & !ads
        # (!ramb & ads);

!ramoe  = (address >= ^hE0000000) & (address <= ^hE00FFFFF) & !w_r & !ads
        # (!ramoe & ads);

" 16 bit I/O                            address: D0000000 - D00FFFFF
!io16   = (address >= ^hD0000000) & (address <= ^hD00FFFFF) & !ads
        # (!io16 & ads);

" 8 bit I/O                             address: C0000000 - C00FFFFF
!io8    = (address >= ^hC0000000) & (address <= ^hC00FFFFF) & !ads
        # (!io8 & ads);

" SLOW 16 bit I/O                       address: B0000000 - B00FFFFF
!extcs  = (address >= ^hB0000000) & (address <= ^hB00FFFFF) & !ads
        # (!extcs & ads);

test_vectors
([address,w_r,ads] -> [io8,io16,ram,rama,ramb,ramoe,roma,romb]);
[^hC0000000,0,0] -> [0,1,1,1,1,1,1,1];
[^hC0000000,0,1] -> [0,1,1,1,1,1,1,1];
[^hD0000000,0,0] -> [1,0,1,1,1,1,1,1];
[^hD0000000,0,1] -> [1,0,1,1,1,1,1,1];
[^hE0000000,0,0] -> [1,1,0,0,1,0,1,1];
[^hE0000000,0,1] -> [1,1,0,0,1,0,1,1];
[^hE0000000,1,0] -> [1,1,0,0,1,1,1,1];
[^hE0000000,1,1] -> [1,1,0,0,1,1,1,1];
[^hE0020000,0,0] -> [1,1,0,1,0,0,1,1];
[^hE0020000,0,1] -> [1,1,0,1,0,0,1,1];
[^hE0020000,1,0] -> [1,1,0,1,0,1,1,1];
[^hE0020000,1,1] -> [1,1,0,1,0,1,1,1];

[^hFFFC0000,0,0] -> [1,1,1,1,1,1,1,0];
[^hFFFC0000,0,1] -> [1,1,1,1,1,1,1,0];
[^hFFFC0000,1,0] -> [1,1,1,1,1,1,1,0];
[^hFFFC0000,1,1] -> [1,1,1,1,1,1,1,0];

[^hFFFE0000,0,0] -> [1,1,1,1,1,1,0,1];
[^hFFFE0000,0,1] -> [1,1,1,1,1,1,0,1];
[^hFFFE0000,1,0] -> [1,1,1,1,1,1,0,1];
```

```
[^hFFFE0000,1,1] -> [1,1,1,1,1,1,0,1];

[^hFFFFFF00,0,0] -> [1,1,1,1,1,1,0,1];
[^hFFFFFF00,0,1] -> [1,1,1,1,1,1,0,1];
[^hFFFFFF00,1,0] -> [1,1,1,1,1,1,0,1];
[^hFFFFFF00,1,1] -> [1,1,1,1,1,1,0,1];

[^h00000000,0,0] -> [1,1,1,1,1,1,1,1];
[^h00000000,0,1] -> [1,1,1,1,1,1,1,1];
end _c1010u38
```



AD2	AD1	/IOWR	/IORD	ACTION
0	0	0	1	WRITE CAMAC READ BUFFER
0	0	1	0	READ CAMAC WRITE BUFFER
0	1	0	1	WRITE LAM REGISTER USING D0
0	1	1	0	READ LAM REGISTER INTO D0
1	0	0	1	WRITE LAM ENABLE USING D0
1	0	1	0	READ LAM ENABLE INTO D0
1	1	0	1	NO EFFECT
1	1	1	0	READ VECTOR

NOTE: INTERRUPT IS CLEARED BY READ OR WRITE CAMAC BUFFER, AND READ VECTOR IF LAST FUNCTION WAS A CONTROL FUNCTION.

TITLE	C1010 Camac & Chip Cont
COMPANY	Fermilab, RD/EED Control
DESIGNER	Walter Knopf
SIZE	E
NUMBER	1.00
REV	A
DATE	4:24p 2-01-1994
SHEET	1 OF 1

```
%***********************************
*   Chip Select decoder for         *
*     C1080 Camac Interface.        *
*   'ADR_DEC' in 'C1010.GDF'        *
*   (MULTIBUS I Interface)          *
*   W. Knopf, April 9, 1992         *
***********************************%
DESIGN IS "adr_dec"
  DEVICE IS "c1010cam";
SUBDESIGN adr_dec
(
  rd            : INPUT;  % Read Strobe                %
  wr            : INPUT;  % Write Strobe               %
  io16, io8     : INPUT;  % Memory data size           %
  ad[15..8]     : INPUT;  % Processor addresses ad9-15  %
  bufcs         : OUTPUT; % ARCnet data buffer access   %
  comcs         : OUTPUT; % ARCnet register access      %
  camcs         : OUTPUT; % CAMAC buffer                %
  clkcs         : OUTPUT; % TevClock Mask RAM           %
  ciack         : OUTPUT; % Read TCLK FIFO              %
  duart         : OUTPUT; % Serial I/O - 2692 DUART     %
  nvram         : OUTPUT; % Battery Backed SRAM         %
  ledrd         : OUTPUT; % Diagnostic LED array (8)    %
  ledwr         : OUTPUT; % Diagnostic LED array (8)    %
  )

BEGIN


  camcs  = (ad[15..8]==h"00") & !io16 &  io8;
  bufcs  = (ad[15..8]==h"00") & !io16 &  io8;

  comcs  = (ad[15..8]==h"00") &  io16 & !io8;
  clkcs  = (ad[15..8]==h"01") &  io16 & !io8;
  ciack  = (ad[15..8]==h"02") &  io16 & !io8;
  duart  = (ad[15..8]==h"03") &  io16 & !io8;
  ledrd  = (ad[15..8]==h"04") &  io16 & !io8 & !rd;
  ledwr  = (ad[15..8]==h"04") &  io16 & !io8 & !wr;
  nvram  = (ad[15..8]> h"07") &  io16 & !io8;


END;
```

```
%**************************************
*   Canmac function decoder for      *
*     C1010 Camac Interface.         *
*   'C1010DEC' in 'c1010cam.gdf'     *
*   (MADC Controller)        *
*   W. Knopf, 21-Apr-1992            *
**************************************%
DESIGN IS "C1010DEC"
  DEVICE IS "EPM5128";
SUBDESIGN C1010DEC
(
  N             : INPUT; % Station select line      %
  /F16,/F8,/F4,/F2,/F1  : INPUT; % function codes   %
  /A8,/A4,/A2,/A1   : INPUT; % Camac sub-addresses  %
  ABORT         : INPUT; % abort cycle, collision with CPU %
  STB           : INPUT; % Gated S2 clock for latching F & A    %
  CMCRD         : OUTPUT; % Camac Read cycle (F0-F7)        %
  CMCWR         : OUTPUT; % Camac write cycle (F16-F23)     %
  CMCCT         : OUTPUT; % control cycle (F8-F15,F24-F31)  %
  CRES, TLAM        : OUTPUT; % Reset and Test LAM cycles   %
  CYCLE, /X     : OUTPUT; % Valid cycle, x-response %
  FMATCH        : OUTPUT; % Same Function as last one %
  AMATCH        : OUTPUT; % Same Subaddress as last one %
  CRQ[7..0]     : OUTPUT; % Interrupt vectors from latched F & A %
)
VARIABLE
  vff[7..0], ffcf[4..0], ffca[3..0] : DFF;
  valid, cpu_req, fa[8..0] : NODE;
  cx : NODE;

BEGIN
DEFAULTS
  CYCLE = 0;
  cpu_req = 0;
END DEFAULTS;
  vff[].clk = STB;
  ffcf[].clk = STB;
  ffca[].clk = STB;


% declare series for F & A for easy table definitions %

  !fa8      = /F16;
  !fa7      = /F8;
  !fa6      = /F4;
  !fa5      = /F2;
  !fa4      = /F1;
  !fa3      = /A8;
  !fa2      = /A4;
  !fa1      = /A2;
  !fa0      = /A1;
  /X        = !cx;

ffcf[4..0].d = fa[8..4];
ffca[3..0].d = fa[3..0];
```

```
%   All Camac functions are mapped in the table.    %
% The left column defines the F and A value (in hex).   %
% The right column defines the interrupt vector     %
% generated for this function and if a valid dataway    %
% cycle occured requiring an interrupt to the cpu.  %

TABLE fa[8..4],fa[3..0] => cpu_req, vff[7..0].d;
     0,  0      => 1, h"00";    % read group 1, channels 0-15 %
     0,  1      => 1, h"01";    %  MADC channels 0-15 %
     0,  2      => 1, h"02";
     0,  3      => 1, h"03";
     0,  4      => 1, h"04";
     0,  5      => 1, h"05";
     0,  6      => 1, h"06";
     0,  7      => 1, h"07";
     0,  8      => 1, h"08";
     0,  9      => 1, h"09";
     0, 10      => 1, h"0A";
     0, 11      => 1, h"0B";
     0, 12      => 1, h"0C";
     0, 13      => 1, h"0D";
     0, 14      => 1, h"0E";
     0, 15      => 1, h"0F";
     1,  0      => 1, h"10";    % read group 2, channels 0-15 %
     1,  1      => 1, h"11";    %  MADC channels 16-31 %
     1,  2      => 1, h"12";
     1,  3      => 1, h"13";
     1,  4      => 1, h"14";
     1,  5      => 1, h"15";
     1,  6      => 1, h"16";
     1,  7      => 1, h"17";
     1,  8      => 1, h"18";
     1,  9      => 1, h"19";
     1, 10      => 1, h"1A";
     1, 11      => 1, h"1B";
     1, 12      => 1, h"1C";
     1, 13      => 1, h"1D";
     1, 14      => 1, h"1E";
     1, 15      => 1, h"1F";
     2,  0      => 1, h"20";    % read group 3, channels 0-15 %
     2,  1      => 1, h"21";    %  MADC channels 32-47 %
     2,  2      => 1, h"22";
     2,  3      => 1, h"23";
     2,  4      => 1, h"24";
     2,  5      => 1, h"25";
     2,  6      => 1, h"26";
     2,  7      => 1, h"27";
     2,  8      => 1, h"28";
     2,  9      => 1, h"29";
     2, 10      => 1, h"2A";
     2, 11      => 1, h"2B";
     2, 12      => 1, h"2C";
     2, 13      => 1, h"2D";
     2, 14      => 1, h"2E";
     2, 15      => 1, h"2F";
     3,  0      => 1, h"30";    % read group 4, channels 0-15 %
     3,  1      => 1, h"31";    %  MADC channels 48-63 %
     3,  2      => 1, h"32";
     3,  3      => 1, h"33";
     3,  4      => 1, h"34";
     3,  5      => 1, h"35";
     3,  6      => 1, h"36";
     3,  7      => 1, h"37";
     3,  8      => 1, h"38";
     3,  9      => 1, h"39";
```

```
   3, 10      => 1, h"3A";
   3, 11      => 1, h"3B";
   3, 12      => 1, h"3C";
   3, 13      => 1, h"3D";
   3, 14      => 1, h"3E";
   3, 15      => 1, h"3F";
   5,  2      => 1, h"42";    % read status word  %
   6,  0      => 1, h"40";    % read module i.d. %
   6,  1      => 1, h"41";    % read firmware version %
   6,  3      => 1, h"43";    % read FOP status %
   6,  4      => 1, h"44";    % read FOP data %
   6,  5      => 1, h"45";    % read module Serial Number %
   6,  6      => 1, h"46";    % read ARCNET node number %
   6, 13      => 1, h"4D";    % read LAM mask register %
   6, 14      => 1, h"4E";    % read LAM request register %
   8,  0      => 0, h"7F";    % special - test lam %
   9,  0      => 0, h"7F";    % special - hardware reset %
  10,  0      => 1, h"50";    % F(10)*A(0)-Clear LAM %
  14,  0      => 1, h"51";    % F(14)*A(0)-Set LAM %
  19,  2      => 1, h"52";    % write FOP control %
  19,  3      => 1, h"53";    % write FOP data %
  19,  6      => 1, h"56";    % Reset ARCNET, write node number %
  19, 12      => 1, h"5C";    % Clear LAM (Sel BITS) %
  19, 13      => 1, h"5D";    % write LAM mask register %
  19, 14      => 1, h"5E";    % write LAM request register %
  24, 15      => 1, h"54";    % F(24)*A(15)-Disable LAM %
  26, 15      => 1, h"55";    % F(26)*A(15)-Enable LAM %
END TABLE;
```

```
  CRQ[7..0] = vff[7..0];
  CYCLE = N & cpu_req;
  valid = CYCLE & !ABORT;
  CMCRD = valid & !fa[8] & !fa[7];
  CMCWR = valid &  fa[8] & !fa[7];
  CMCCT = valid & fa[7];
  CRES  = N & (fa[8..0]==h"90");
  TLAM  = N & (fa[8..0]==h"80");
  cx    = CYCLE # CRES # TLAM;
  FMATCH=(ffcf[4..0]==fa[8..4]) & CMCRD;
  AMATCH = (ffca[3..0]==fa[3..0]) & valid;

END;
```