```
                JUMP DATPRRX;
        ELSE    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS0 XWT XDK SDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DATPRR;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_RANDOM_WRITE                             %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
DATPRW:         IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS0 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX0]
                JUMP DATPRW;
        ELSEIF I4               %FRDK%
        THEN    [CTRLDEF
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX0]
                JUMP DATPRWX;
        ELSE    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS0 XWT XDK SDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DATPRW;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_SEC_ADDRESS_READ                         %
%       --SLOW-- --READ--                                               %
%-----------------------------------------------------------------------%
DATPSR:         IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTS;
        ELSEIF I7               %FRWT%
        THEN    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS2 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 NDSAK
MUX0]
                JUMP DATPSR;
        ELSEIF I4               %FRDK%
        THEN    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS2 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 DSAK MUX0]
                JUMP DATPSRX;
        ELSE    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS2 XWT XDK SDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT6 NDSAK MUX0]
```

```
                JUMP DATPSR;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_SEC_ADDRESS_WRITE                        %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
DATPSW:         IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS2 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 NDSAK
MUX0]
                JUMP DATPSW;
        ELSEIF I4               %FRDK%
        THEN    [CTRLDEF
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 NDSAK
MUX0]
                JUMP DATPSWX;
        ELSE    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS2 XWT XDK SDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT6 NDSAK MUX0]
                JUMP DATPSW;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_BLOCK_TRANSFER_READ                      %
%       --SLOW-- --READ--                                               %
%-----------------------------------------------------------------------%
DATPBR:                 [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX0]
                JUMP DATPBRX;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_BLOCK_TRANSFER_WRITE                     %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
DATPBW:                 [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX0]
                JUMP DATPBWX;


%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_BLOCK_TRANSFER_TERMINATE                 %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%       This routine does a graceful termination of a block transfer    %
%       where DS and DK are brought to their idle states without        %
%       effecting the slave.                                            %
```

```
%               MS=0 RD=0                                                %
%               then                                                     %
%               DS=0                                                     %
DATPBT:         IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CTRLDEF
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX1]
                JUMP DATPBT;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG NRD MS0 XWT XDK CDS XAS
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK MUX1]
                JUMP WAYHOME;

%-----------------------------------------------------------------------%
%       Routine LOCAL_COUNTER_LOAD                                      %
%       --SLOW-- --WRITE--                                              %
%-----------------------------------------------------------------------%
LC_LOAD:        [CLOAD NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT9 DSAK MUX1]
                JUMP LC_LOADX;


%-----------------------------------------------------------------------%
%       Routine LOCAL_COUNTER_READ                                      %
%       --SLOW-- --READ--                                               %
%-----------------------------------------------------------------------%
LC_READ:        [CREAD NSRT NDW COUNTIN NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT9 DSAK MUX1]
                JUMP LC_READX;

%-----------------------------------------------------------------------%
%       Routine FIFO_WRITE                                              %
%       --SLOW-- --WRITE--                                              %
%-----------------------------------------------------------------------%
FIFO_WR:        [CDEF NSRT DW DATAOUT NFDOE FCOE NIRQ DFFEN
                FBDEF
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT10 DSAK MUX1]
                JUMP FIFO_WRX;

%-----------------------------------------------------------------------%
%       Routine DATA_FIFO_BLOCK_TRANSFER_READ                           %
%       --FAST-- --WRITE or READ--                                      %
%-----------------------------------------------------------------------%
DATFBR:         IF /I6                  %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
```

```
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                JUMP DATFBR;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK SDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                JUMP DATFBRX;

%-----------------------------------------------------------------------%
%       Routine DATA_FIFO_PIPELINED_READ_1  (100ns transfers)           %
%       --FAST-- --WRITE or READ--                                      %
%-----------------------------------------------------------------------%
DATFP1:         IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP DATFP1;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK SDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP DATFP1X;

%-----------------------------------------------------------------------%
%       Routine DATA_FIFO_PIPELINED_READ_2  (200ns transfers)           %
%       --FAST-- --WRITE or READ--                                      %
%-----------------------------------------------------------------------%
DATFP2:         IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP DATFP2;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK SDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP DATFP2X;

%-----------------------------------------------------------------------%
%       Routine DATA_FIFO_PIPELINED_READ_4  (400ns transfers)           %
%       --FAST-- --READ or WRITE--                                      %
%-----------------------------------------------------------------------%
DATFP4:         IF /I6          %FBREQ*%
        THEN    [CTRLDEF
```

```
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP DATFP4;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK SDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP DATFP4X;

%-----------------------------------------------------------------------%
%       SLAVE_DATA_INPUT                                                %
%       --SLOW-- --READ--                                               %
%-----------------------------------------------------------------------%
SLVDIN:                 [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                NREQ NEOBA NEG NRD MS0 CWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT13 NDSAK
MUX0]
                JUMP SLVDINX;

%-----------------------------------------------------------------------%
%       SLAVE_DATA_OUTPUT                                               %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%       (WT has been cleared back at LETSGO).                           %
%       IF FBREQ* THEN release bus, jump WAYHOME.                       %
%       ELSEIF WT=1 THEN wait here.                                     %
%       ELSEIF DS=1 THEN set DK jump HOME.                              %
%       ELSE clear DK jump HOME.                                        %

SLVDOUT:IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP WAYHOME;
        ELSEIF I7       %FRWT%
        THEN    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT13 NDSAK
MUX0]
                JUMP SLVDOUT;
        ELSEIF I1       %FRDS%
        THEN    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                NREQ NEOBA NEG NRD MS0 XWT SDK XDS XAS
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT13 NDSAK
MUX1]
                JUMP WAYHOME;
        ELSE    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                NREQ NEOBA NEG NRD MS0 XWT CDK XDS XAS
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT13 NDSAK
MUX1]
                JUMP WAYHOME;
```

```
%-----------------------------------------------------------------------%
%       Null                                                            %
%       --FAST-- --WRITE or READ--                                      %
%-----------------------------------------------------------------------%
NULL:           [CTRLDEF
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT12 NDSAK
MUX1]
                JUMP NULLX;


%-----------------------------------------------------------------------%
%       End Of Event                                                    %
%       --SLOW-- --WRITE --                                             %
%-----------------------------------------------------------------------%
EOE:            [CHOLD NSRT DW DATAOUT NFDOE FCOE NIRQ DFFEN
                FBDEF
                PFFDIS NSR NSCLK NFCLK DEOE CEOE NTMR NCCLR STAT14 DSAK MUX1]
                JUMP EOEX;


%-----------------------------------------------------------------------%
%       Bulb Test                                                       %
%       --FAST-- --WRITE or READ--                                      %
%-----------------------------------------------------------------------%
BULBTST:        [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE IRQ DFFDIS
                HOLDB NEOBA EG RD MS7 XWT SDK SDS SAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT11 NDSAK
MUX1]
                JUMP BULBTST;


%-----------------------------------------------------------------------%
%       Sequencer Select                                                %
%       Wait here to be selected by the 68020                           %
%               FB1S* + FB2S means that the FASTBUS standard sequencer  %
%                       has been selected.                              %
%               FB1S* * FB2S* means that the USER sequencer has been    %
%                       selected.                                       %
%                                                                       %
%       Whichever sequencer has not been selected, puts it's outputs    %
%               into high Z, and is not accessable unless it is reset.  %
%-----------------------------------------------------------------------%
SELECT:         IF /I1 * /I5            %FB1S* * FB2S*%
        THEN [Z] CONTINUE;
        ELSEIF (/I1 * I5) + (I1 * /I5)          %FB2S* + FB1S*%
        THEN [RESETO2] JUMP HOME;
        ELSE [RESETO2] JUMP SELECT;

TURNOFF:[Z] JUMP TURNOFF;
%-----------------------------------------------------------------------%
%       If, in the cycle executed just previous to returning here,      %
%       DSACK was asserted, then the select line (FBxS*)                %
%       will still be asserted (it takes the 68020 and it's PAL's a     %
%       certain amount of time to negate a select line after it sees    %
%       DSACK).  HOME would then see FBxS* true and think that it is    %
```

```
%       being selected again, when, in fact, FBxS* is just lingering     %
%       from the previous cycle.  As you might imagine, if this were     %
%       to happen, the sequencer would jump to what ever address         %
%       just happened to be on the 68020's address bus at that time      %
%       and anarchy would reign.  So, to keep order, if we               %
%       want to come HOME right after DSACK, then we instead return      %
%       here where three NOP type instructions are executed.  This       %
%       allows the 68020 enough time to negate FBxS* from any            %
%       preveously executed cycle.                                       %
%-----------------------------------------------------------------------%
WAYHOME:        [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 NDSAK
MUX1]
                CONTINUE;

                [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 NDSAK
MUX1]
                CONTINUE;

                [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 NDSAK
MUX1]
                CONTINUE;

%       Since the IOD lines could be latched on the next cycle (if there%
%       is a pending instruction), the Processor transceivers must be    %
%       set to DATAOUT before coming HOME.  Any instructions leading to %
%       HOME must have the processor transceivers set in this way or it %
%       will be        bad.                                              %

%       When HOME, all outputs are default                        %
HOME:   IF /I1          %FB1S* (Fast cycle, return DSACK now.)%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 DSAK MUX2]
                JUMP LETSGOF;
        ELSEIF /I5      %FB2S* (Slow cycle, return DSACK in routine.%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 NDSAK
MUX2]
                JUMP LETSGOS;
        ELSE    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 NDSAK
MUX1]
                JUMP HOME;

%-----------------------------------------------------------------------%
%       These are the fast and slow jumps to the appropriate FASTBUS     %
%       primative routines as determined by the vector received from     %
%       the '020 at MUX2.                                                %
%-----------------------------------------------------------------------%
```

```
LETSGOF:        [CDEF NSRT NDW DATAOUTLATCH NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 DSAK MUX2]
                PUSHI;

%       Clear WT here is for Slave write routine, dosn't effect anything
else.%
                [CDEF NSRT NDW DATAOUTHOLD NFDOE FCOE NIRQ DFFDIS
                NREQ NEOBA NEG NRD MS0 CWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 NDSAK
MUX0]
                RETURN;

LETSGOS:        [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 NDSAK
MUX2]
                PUSHI;

                [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT0 NDSAK
MUX0]
                RETURN;

%-----------------------------------------------------------------------%
%       Routine ARBITRATE                                               %
%       --SLOW-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%                                                                       %
%       Starting with something simple, this routine checks to make     %
%       sure that FBREQ is false, then sets FREQ true, waits for        %
%       FRDY (means that we have the bus), then returns DSACK and goes  %
%       home.                                                           %
%-----------------------------------------------------------------------%
ARBX:           [CTRLDEF
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT1 DSAK MUX1]
                JUMP WAYHOME;

%-----------------------------------------------------------------------%
%       Bus Release                                                     %
%-----------------------------------------------------------------------%
%       Release Bus, Status=2                                           %
%                                                                       %
%       Status=2, Status clock = 1                                      %
BUSRELX:        [CTRLDEF
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT2 NDSAK MUX1]
                JUMP WAYHOME;

%-----------------------------------------------------------------------%
%       Routine ADDRESS_RELEASE                                         %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%               Wait for AK to go low, then go HOME.                    %
```

```
ADDRELX:IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I5               %FRAK%
        THEN    [CTRLDEF
                HOLDB NEOBA NEG NRD MS0 XWT CDK CDS CAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT4 NDSAK MUX0]
                JUMP ADDRELX;
        ELSE    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT4 NDSAK MUX1]
                JUMP HOME;

%-----------------------------------------------------------------------%
%-----------------------------------------------------------------------%
%       Data Cycle Routines                                             %
%-----------------------------------------------------------------------%
%-----------------------------------------------------------------------%

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_RANDOM_READ                              %
%       --SLOW-- --READ--                                               %
%-----------------------------------------------------------------------%
%       IF FBREQ THEN clear FASTBUS lines, TMR=0, JUMP TO INTS.         %
%       ELSEIF WT=1 THEN Enable FASTBUS transceivers and drivers        %
%               (FDOE*=0,DDIR=1,CPBA=0, DSAB=0,DSBA=0,DG*=0), enable    %
%               processor transceivers (POE*=0), Assert RD=1, MS=0 and  %
%               loop here.                                              %
%       ELSEIF DK=1 THEN return DSACK, and continue.                    %
%       ELSE Enable FASTBUS AD line drivers, enable proccessor xceivers %
%               Assert RD=1, MS=0, DS(u), TMR, and loop here.           %
%                                                                       %
%       Hold DSACK true for one more cycle, and continue.               %
%                                                                       %
%       IF FBREQ THEN clear FASTBUS lines, TMR=0, NDSACK, JUMP TO       %
%               WAYHOME.                                                %
%       ELSEIF WT=1 THEN negate DSACK, disable processor and FASTBUS    %
%               transceivers, continue to assert RD=1, MS=0, and DS=1,  %
%               and loop here.                                          %
%       ELSEIF DK=0 THEN negate DSACK, disable processor and FASTBUS    %
%               transceivers, deassert TMR, RD, and MS, and go WAYHOME. %
%       ELSE negate DSACK, disable processor and FASTBUS transceivers,  %
%               deassert RD, MS, and DS, assert TMR, and loop here.     %
%                                                                       %
%-----------------------------------------------------------------------%
DATPRRX:        [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS0 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 DSAK MUX0]
                CONTINUE;

WAIT1:  IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
```

```
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CTRLDEF
                HOLDB NEOBA NEG RD MS0 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX0]
                JUMP WAIT1;
        ELSEIF /I4              %/FRDK%
        THEN    [CTRLDEF
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK MUX1]
                JUMP WAYHOME;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG NRD MS0 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP WAIT1;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_RANDOM_WRITE                             %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%       IF FBREQ THEN clear FASTBUS lines, TMR=0, JUMP TO WAYHOME.      %
%       ELSEIF WT=1 THEN Enable FASTBUS transceivers and drivers        %
%               (FDOE*=0, DDIR=0, CPBA=0, DSAB=0, DSBA=1, DG*=0),       %
%               Assert RD=0, MS=0 and loop here.                        %
%       ELSEIF DK=1 THEN disable FASTBUS transceivers and drivers, and  %
%               continue.                                               %
%       ELSE Assert DS(u), TMR, keep FASTBUS lines asserted, and loop   %
%               here.                                                   %
%       be bop a lula she's my baby                                     %
%       IF FBREQ THEN clear FASTBUS lines, TMR=0, JUMP TO HOME.         %
%       ELSEIF WT=1 THEN continue to assert DS=1, and loop here.        %
%       ELSEIF DK=0 THEN go HOME.                                       %
%       ELSE deassert DS, assert TMR, and loop here.                    %
%-----------------------------------------------------------------------%
DATPRWX:IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CTRLDEF
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX0]
                JUMP DATPRWX;
        ELSEIF /I4              %/FRDK%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK MUX1]
                JUMP HOME;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG NRD MS0 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
```

```
                JUMP DATPRWX;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_SEC_ADDRESS_READ                         %
%       --SLOW-- --READ--                                               %
%-----------------------------------------------------------------------%
%       IF FBREQ THEN clear FASTBUS lines, TMR=0, JUMP TO INTS.         %
%       ELSEIF WT=1 THEN Enable FASTBUS transceivers and drivers        %
%               (FDOE*=0, DDIR=1,CPBA=0, DSAB=0,DSBA=0,DG*=0), enable   %
%               processor transceivers (POE*=0), Assert RD=1, MS=2 and  %
%               loop here.                                              %
%       ELSEIF DK=1 THEN return DSACK, and continue.                    %
%       ELSE Assert DS(u), TMR, and loop here.                          %
%                                                                       %
%       Hold DSACK true for one more cycle, and continue.               %
%                                                                       %
%       IF FBREQ THEN clear FASTBUS lines, TMR=0, DEassert DSACK, jump  %
%               to WAYHOME.                                             %
%       ELSEIF WT=1 THEN negate DSACK, disable processor and FASTBUS    %
%               transceivers, continue to assert RD=1, MS=2, and DS=1,  %
%               and loop here.                                          %
%       ELSEIF DK=0 THEN negate DSACK, disable processor and FASTBUS    %
%               transceivers, deassert RD, and MS, and go WAYHOME.      %
%       ELSE negate DSACK, disable processor and FASTBUS transceivers,  %
%               deassert RD, MS, and DS, assert TMR, and loop here.     %
%-----------------------------------------------------------------------%
DATPSRX:        [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS2 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 DSAK MUX0]
                CONTINUE;

WAIT2:  IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CTRLDEF
                HOLDB NEOBA NEG RD MS2 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 NDSAK
MUX0]
                JUMP WAIT2;
        ELSEIF /I4              %/FRDK%
        THEN    [CTRLDEF
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 NDSAK MUX1]
                JUMP WAYHOME;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG NRD MS2 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT6 NDSAK MUX0]
                JUMP WAIT2;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_SEC_ADDRESS_WRITE                        %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
```

```
%       IF FBREQ THEN clear FASTBUS lines, TMR=0, jump to WAYHOME.        %
%       ELSEIF WT=1 THEN Enable FASTBUS transceivers and drivers          %
%               (FDOE*=0, DDIR=0, CPBA=0, DSAB=0, DSBA=1, DG*=0),         %
%               Assert RD=0, MS=2 and loop here.                          %
%       ELSEIF DK=1 THEN disable FASTBUS transceivers and drivers, and    %
%               continue.                                                 %
%       ELSE Assert DS(u), keep FASTBUS lines asserted, assert TMR,       %
%               and loop here.                                            %
%       IF FBREQ THEN clear FASTBUS lines, TMR=0, jump to HOME.           %
%       ELSEIF WT=1 THEN continue to assert DS=1, and loop here.          %
%       ELSEIF DK=0 THEN go HOME.                                         %
%       ELSE deassert DS, assert TMR, and loop here.                      %
%-----------------------------------------------------------------------%
DATPSWX:IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CTRLDEF
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 NDSAK
MUX0]
                JUMP DATPSWX;
        ELSEIF /I4              %/FRDK%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT6 NDSAK MUX1]
                JUMP HOME;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG NRD MS0 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT6 NDSAK MUX0]
                JUMP DATPSWX;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_BLOCK_TRANSFER_READ                        %
%       --SLOW-- --READ--                                                 %
%-----------------------------------------------------------------------%
%       This routine is going to be a little bit different than any of    %
%               the others, in that, DSACK will have to be returned       %
%               after each DK(t) to allow the 68020 to read the data.     %
%       Here goes:                                                        %
%                                                                         %
%       Turn on FASTBUS AD line receivers, Assert RD=1, MS=1,switch       %
%               to MUX1 and continue.                                     %
%       IF FBREQ THEN Release bus, JUMP TO INTS.                          %
%       ELSEIF WT=1 THEN Assert RD=1, MS=1, on FASTBUS  Turn on FASTBUS %
%               xceivers and enable processor xceivers                    %
%               (POE*=0, DDIR=1, CPBA=0, DSAB=0, DSBA=0, DG*=0, FDOE*=0)%
%               and loop here.                                            %
%       ELSEIF DS=0 THEN Assert RD=1, MS=1, DS(u), enable FASTBUS and     %
%               processor transceivers (POE*=0, DDIR=1, CPBA=0,           %
%               DSAB=0, DSBA=0, DG*=0, FDOE*=0)and assert TMR, and JUMP %
%               to DKUPR.                                                 %
%       ELSE Assert RD=1, MS=1, DS(d), enable FASTBUS and processor       %
%               transceivers (POE*=0, DDIR=1, CPBA=0, DSAB=0,             %
```

```
%               DSBA=0, DG*=0, FDOE*=0) assert TMR, and JUMP to DKDWNR.  %
%DKDWNR                                                                  %
%       IF FBREQ THEN RD=1, MS=1, Clear DS, jump to INTS.                %
%       ELSEIF DK=0 THEN Assert DSACK, continue to drive FASTBUS lines, %
%               and processor and FASTBUS transceivers, deassert TMR,    %
%               and JUMP to next1.                                       %
%       ELSE Continue to drive FASTBUS lines, and transceivers, and      %
%               TMR, and loop here.                                      %
%DKUPR                                                                   %
%       IF FBREQ THEN RD=1, MS=1, Clear DS, assert DSACK, jump to        %
%               NEXT2.                                                   %
%       ELSEIF DK=1 THEN Assert DSACK, continue to drive FASTBUS lines, %
%               and processor and FASTBUS transceivers, deassert TMR,    %
%               and JUMP to next1.                                       %
%       ELSE Continue to drive FASTBUS lines, and transceivers,          %
%               assert TMR, and loop here.                               %
%NEXT1                                                                   %
%       Continue to drive FASTBUS lines, DSACK, and transceivers, and    %
%               Continue.                                                %
%       Continue to drive FASTBUS lines, and transceivers negate DSACK   %
%               and JUMP WAYHOME.                                        %
%                                                                        %
%NEXT2  Clear FASTBUS lines, release bus, keep DSACK asserted, jump       %
%               to WAYHOME.                                              %
%------------------------------------------------------------------------%
DATPBRX:IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTS;
        ELSEIF I7               %FRWT%
        THEN    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX0]
                JUMP DATPBRX;
        ELSEIF /I1              %/FRDS%
        THEN    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK SDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DKUPR;
        ELSE    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DKDWNR;

DKDWNR:         IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                HOLDB NEOBA NEG RD MS1 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTS;
        ELSEIF /I4              %/FRDK%
        THEN    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 DSAK MUX1]
```

```
                JUMP NEXT1;
        ELSE    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DKDWNR;

DKUPR:  IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                HOLDB NEOBA NEG RD MS1 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 DSAK
MUX1]
                JUMP NEXT2;
        ELSEIF I4               %FRDK%
        THEN    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 DSAK MUX1]
                JUMP NEXT1;
        ELSE    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DKUPR;

NEXT1:          [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 DSAK MUX1]
                CONTINUE;

                [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK MUX1]
                JUMP WAYHOME;

NEXT2:          [CTRLDEF
                REL
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 DSAK MUX1]
                JUMP WAYHOME;

%-----------------------------------------------------------------------%
%       Routine DATA_PROCESSOR_BLOCK_TRANSFER_WRITE                     %
%       --FAST-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%       As with DATPBR, DSACK will have to be returned after each DK(t) %
%               to allow the 68020 to output new data.                  %
%                                                                       %
%       Keep FASTBUS AD line drivers enabled, assert MS=1, set MUX1,    %
%               and continue.                                           %
%       IF FBREQ THEN Release bus, JUMP TO WAYHOME.                     %
%       ELSEIF WT=1 THEN Assert MS=1, keep FASTBUS transceivers enabled,%
%               and enable FASTBUS drivers (POE*=1, DDIR=0,             %
%               CPBA=0, DSAB=0, DSBA=1, DG*=0, FDOE*=0), and loop here. %
%       ELSEIF DS=0 THEN Assert MS=1, DS(u), TMR, keep FASTBUS          %
%               transceivers enabled and enable FASTBUS drivers (POE*=1,%
%               DDIR=0, CPBA=0, DSAB=0, DSBA=1, DG*=0, FDOE*=0), and    %
%               JUMP to DKUPW.                                          %
%       ELSE Assert MS=1, DS(d), TMR, keep FASTBUS transceivers enabled,%
%               and enable FASTBUS drivers (POE*=1, DDIR=0,             %
%               CPBA=0, DSAB=0, DSBA=1, DG*=0, FDOE*=0), and JUMP to    %
```

```
%               DKDWNW.                                                 %
%DKDWNW                                                                 %
%       IF FBREQ THEN Release bus, JUMP TO HOME.                        %
%       ELSEIF DK=0 THEN deassert MS, TMR, disable FASTBUS transceivers %
%               and drivers (POE*=0, DDIR=0, CPBA=0, DSAB=0, DSBA=0,    %
%               DG*=1, FDOE*=1), and JUMP home.                         %
%       ELSE continue to assert MS, TMR, and keep FASTBUS transceivers  %
%               and drivers enabled, and loop here.                     %
%DKUPW                                                                  %
%       IF FBREQ THEN Release bus, JUMP TO HOME.                        %
%       ELSEIF DK=1 THEN deassert MS, TMR, disable FASTBUS transceivers %
%               and drivers (POE*=0, DDIR=0, CPBA=0, DSAB=0, DSBA=0,    %
%               DG*=1, FDOE*=1), and JUMP home.                         %
%       ELSE continue to assert MS, TMR, and keep FASTBUS transceivers  %
%               and drivers enabled, and loop here.                     %
%-----------------------------------------------------------------------%
DATPBWX:IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK
MUX0]
                JUMP DATPBWX;
        ELSEIF /I1              %/FRDS%
        THEN    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS1 XWT XDK SDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DKUPW;
        ELSE    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS1 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DKDWNW;

DKDWNW:         IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF /I4              %/FRDK%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK MUX1]
                JUMP HOME;
        ELSE    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DKDWNW;

DKUPW:  IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
```

```
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I4               %FRDK%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT5 NDSAK MUX1]
                JUMP HOME;
        ELSE    [CDEF NSRT NDW DATAOUTHOLD FDOE FCOE NIRQ DFFDIS
                HOLDB NEOBA NEG NRD MS1 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT5 NDSAK MUX0]
                JUMP DKUPW;

%-----------------------------------------------------------------------%
%       Routine LOCAL_COUNTER_LOAD                                      %
%       --SLOW-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%       This routine loads the Local Word Counter with data from        %
%       the processor.  Only the lower 12 bits of the 32 bit word       %
%       are used.                                                       %
%                                                                       %
%       Set POE* true, put Local Word Counter into load mode,           %
%               assert DSACK and continue.                              %
%       Negate POE*, put Local Word Counter into hold, keep DSACK       %
%               asserted, and continue.                                 %
%       Negate DSACK (hold DSACK false for two cycles to prevent        %
%               sequencer from seeing FBSEL from this cycle when        %
%               returning home).                                        %
%                                                                       %
LC_LOADX:       [CHOLD NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT9 DSAK MUX1]
                CONTINUE;

                [CHOLD NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT9 NDSAK MUX1]
                JUMP WAYHOME;

%-----------------------------------------------------------------------%
%       Routine LOCAL_COUNTER_READ                                      %
%       --SLOW-- --READ--                                               %
%-----------------------------------------------------------------------%
%       Routine to allow the processor to read the value of the Local   %
%               Word Counter.                                           %
%                                                                       %
%       Set POE* true, put Local Word Counter into Read mode, assert    %
%               DSACK, and continue.                                    %
%       Keep all outputs the same for one more cycle.                   %
%       Negate DSACK (hold DSACK false for two cycles to prevent        %
%               sequencer from seeing FBSEL from this cycle when        %
%               returning home).                                        %
%                                                                       %
LC_READX:       [CREAD NSRT NDW COUNTIN NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT9 DSAK MUX1]
                JUMP WAYHOME;
```

```
%-----------------------------------------------------------------------%
%       Routine FIFO_WRITE                                              %
%       --SLOW-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%       Routine to write a word from the 68020 into the processor       %
%       and data FIFO's.                                                %
%                                                                       %
%       Enable processor transceivers (POE*=0), set FIFO Write true     %
%               (DW=1), enable FIFO's, set DSACK true.                  %
%       Set FCLK true.                                                  %
%       Negate DSACK, DW, FCLK, and POE*, JUMP TO WAYHOME.              %
%                                                                       %
FIFO_WRX:       [CDEF NSRT DW DATAOUT NFDOE FCOE NIRQ DFFEN
                FBDEF
                PFFEN NSR SCLK FCLK NDEOE NCEOE NTMR NCCLR STAT10 DSAK MUX1]
                JUMP WAYHOME;

%-----------------------------------------------------------------------%
%       Routine DATA_FIFO_BLOCK_TRANSFER_READ                           %
%       --FAST-- --WRITE or READ--                                      %
%-----------------------------------------------------------------------%
%       This routine does a FASTBUS handshake block transfer read from  %
%       the FASTBUS crate segment to the FSCC's Output Port Data FIFO,  %
%       and also the Processor Data FIFO.                               %
%-----------------------------------------------------------------------%
%       Note: DDIR, and SAB must be held high throughout the block      %
%               transfer operation.                                     %
%                                                                       %
%DATFBR:IF FBREQ                                                        %
%       THEN release bus jump to INTF                                   %
%       ELSEIF WT=1                                                     %
%       THEN Enable FASTBUS xceivers, enable processor and Data FIFO's  %
%               MS=1, RD=1, Place local word counter into hold, and     %
%               loop here.                                              %
%       ELSE Turn on FASTBUS xcievers, enable processor and Data FIFO's %
%               MS=1, RD=1, Place local word counter into count, set    %
%               DS=1, NO FCLK!, and jump to DATFBRX.                    %
%                                                                       %
%BXFERLP:IF (/FRDK * FRSS1) + EOB + FBREQ                               %
%       THEN Enable FASTBUS xceivers, enable processor and Data FIFO's  %
%               MS=1, RD=1, Place local word counter into hold mode,    %
%               and JUMP to EOBIRQ.                                     %
%       ELSEIF WT=1                                                     %
%       THEN Enable FASTBUS xceivers, enable processor and Data FIFO's  %
%               MS=1, RD=1, Place local word counter into hold, and     %
%               loop here.                                              %
%       ELSEIF DK=0                                                     %
%       THEN Turn on FASTBUS xcievers, enable processor and Data FIFO's %
%               MS=1, RD=1, Place local word counter into count, set    %
%               DS=1, set FCLK, and continue.                           %
%       ELSE Place local word counter into hold, and loop here.         %
%                                                                       %
%DATFBRX:Place counter into hold mode, negate FCLK, set TMR, and        %
%               continue.  This       cycle ensures that the FSCC's minimum %
%               FASTBUS cycle time (100ns) is not violated.  If some    %
%               slave returns DK too quickly, the FSCC could            %
```

```
%               try to do another DS before the FIFO's were ready for    %
%               the next word.  This cycle keeps the sequencer from      %
%               looking for DK before it should.                         %
%                                                                        %
%       Keep TMR true, and do another NOP for the same reason as above.  %
%                                                                        %
%       IF (FRDK* FRSS1) + EOB + FBREQ                                   %
%       THEN counter=hold, NTMR, keep FIFO's enabled and jump to EOBIRQ.%
%       ELSEIF WT=1                                                      %
%       THEN counter=hold, NTMR, keep both FIFO's enabled and loop here.%
%       ELSEIF DK=1                                                      %
%       THEN counter=decr, set DS=0, set FCLK, keep FIFO's enabled,      %
%               Clear TMR, and continue.                                 %
%       ELSE counter=hold, TMR, keep FIFO's enabled, and loop here.      %
%                                                                        %
%       Place local word counter in hold, negate FCLK, keep processor    %
%               and oport FIFO's enabled, set TMR, and JUMP to DATFBR.  %
%               This cycle ensures that the FSCC's minimum cycle time    %
%               is not violated in the same way as above.                %
%       Keep TMR set, do another NOP for the same reason as above, and   %
%               jump to BXFERLP.                                         %
%                                                                        %
%EOBIRQ:IF FBREQ THEN Release bus, jump to HOME.                          %
%       ELSEIF DS=1                                                      %
%       THEN counter=hold, TMR, continue.                                %
%       ELSE counter=hold, TMR, JUMP DSLOWH.                             %
%                                                                        %
%       --case DS=1--                                                    %
%       IF FBREQ THEN Release bus, jump to HOME.                         %
%       ELSEIF DK=1 THEN Clear TMR, continue.                            %
%       ELSE keep TMR true, and wait here.                               %
%                                                                        %
%       Wait two cycles to allow for FIFO write.                         %
%                                                                        %
%       IF FBREQ THEN Release bus, jump to HOME.                         %
%       ELSEIF WT=1                                                      %
%       THEN RD=1, MS=0, counter=hold, FIFO's disabled loop here.        %
%       ELSE DS=0, RD=0, MS=0, FIFO's Disabled continue.                 %
%                                                                        %
%       IF FBREQ THEN Release bus, jump HOME.                            %
%       ELSEIF DK=0 THEN NTMR, EOBA, IRQ, jump HOME.                     %
%       ELSE TMR, wait here.                                             %
%                                                                        %
%DSLOWH:IF FBREQ THEN Release bus, jump to HOME.                          %
%       ELSEIF DK=0 THEN clear TMR, continue.                            %
%       ELSE TMR, wait here.                                             %
%                                                                        %
%       wait three cycles to ensure FIFO write, then IRQ, EOBA, jump     %
%       home.                                                            %
%                                                                        %
BXFERLP:IF (/I4 * I0) + I3 + /I6      %(/FRDK * FRSS1) + FEOB + FBREQ*%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                JUMP EOBIRQ;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
```

```
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                JUMP BXFERLP;
        ELSEIF /I4              %/FRDK%
        THEN    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK SDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                JUMP BXFERLP;

DATFBRX:        [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;

WAIT3:  IF (I4 * I0) + I3 + /I6                %(FRDK * FRSS1) + FEOB + FBREQ*%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                JUMP EOBIRQ;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                JUMP WAIT3;
        ELSEIF I4               %FRDK%
        THEN    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK CDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                JUMP WAIT3;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                JUMP BXFERLP;

EOBIRQ:         IF /I0          %/FRSS1%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
```

```
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;

        IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX0]
                JUMP INTF;
        ELSEIF I1               %FRDS%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                JUMP DSLOWH;

%       Case--DS=1
DSHIGHH:IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX0]
                JUMP INTF;
        ELSEIF I4               %FRDK%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                JUMP DSHIGHH;

                [CHOLD NSRT NDW DATAINFIFO NFDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO NFDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;

WAIT5:  IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CTRLDEF
```

```
                HOLDB NEOBA NEG RD MS0 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK
MUX0]
                JUMP WAIT5;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG NRD MS0 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                CONTINUE;

WAIT6:  IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF /I4              %FRDK=0%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE IRQ DFFDIS
                HOLDB EOBA NEG NRD MS0 XWT XDK XDS XAS
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX1]
                JUMP HOME;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG RD MS0 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                JUMP WAIT6;

%       Case--DS=0
DSLOWH:         IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF /I4              %FRDK=0%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX1]
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT7 NDSAK MUX0]
                JUMP DSLOWH;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX1]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX1]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE IRQ DFFEN
                HOLDB EOBA NEG RD MS1 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT7 NDSAK MUX1]
                JUMP HOME;
```

```
%-----------------------------------------------------------------------%
%       Routine DATA_FIFO_PIPELINED_READ_1  (100ns transfers)           %
%       --FAST-- --WRITE or READ--                                      %
%-----------------------------------------------------------------------%
%       FASTBUS pipeline block transfer read from the FASTBUS crate     %
%       segment to the FSCC's Output Port Data FIFO, and also the       %
%       Processor Data FIFO if enabled through the parrallel port.      %
%                                                                       %
%       Set FASTBUS xcievers for input (DDIR=1, SAB=1, FDOE*=0),        %
%               enable data and processor FIFO's (DFIFOEN=1, PFIFOEN=1),%
%               MS=3, RD=1, and continue.                               %
%                                                                       %
%       Note: DDIR, and SAB must be held high throughout the block      %
%               transfer operation.                                     %
%                                                                       %
%DATFP1:IF FBREQ THEN Release bus, jump to INTS.                         %
%       ELSEIF WT=1                                                     %
%       THEN Enable FASTBUS xceivers for input, MS=3, RD=1, Place local %
%               word counter into hold, and loop here.                  %
%       ELSE Enable FASTBUS xceivers for input, MS=3, RD=1, Place local %
%               word counter into count, set DS=1, and JUMP to DATFP1X:  %
%                                                                       %
%PXFR1LP:IF FBREQ THEN Release bus, jump to INTS.                        %
%       ELSEIF WT=1                                                     %
%       THEN Enable FASTBUS xceivers for input, MS=3, RD=1, Place local %
%               word counter into hold, and loop here.                  %
%       ELSEIF EOB                                                      %
%       THEN Enable FASTBUS xceivers for input, MS=3, RD=1, Place local %
%               word counter into hold mode, and JUMP to EOBPIPE.       %
%       ELSE Enable FASTBUS xceivers for input, MS=3, RD=1, Place local %
%               word counter into count, set DS=1, set FCLK, and        %
%               continue.                                               %
%                                                                       %
%DATFP1X:Place counter into hold mode, negate FCLK, and                 %
%               continue.  This       cycle provides the FSCC's         %
%               FASTBUS cycle time (100ns).                             %
%                                                                       %
%       IF FBREQ THEN Release bus, jump to HOME.                        %
%       ELSEIF WT=1                                                     %
%       THEN counter=hold, keep both FIFO's enabled and loop here.      %
%       ELSEIF EOB                                                      %
%       THEN counter=hold, keep FIFO's enabled and jump to EOBPIPE.     %
%       ELSE counter=decr, set DS=0, set FCLK, keep FIFO's enabled,     %
%               and continue.                                           %
%                                                                       %
%       Place local word counter in hold, negate FCLK, keep processor   %
%               and oport FIFO's enabled, and JUMP to PXFR1LP.          %
%               This cycle provides the FSCC's cycle time.              %
%                                                                       %
%EOBPIPE:IF SS <> 2 (Local counter terminated the block transfer)       %
%       THEN FCLK continue.                                             %
%       ELSE (SS=2 terminated the block transfer) continue.             %
%                                                                       %
%       IF FBREQ THEN Release bus, jump to HOME.                        %
%       ELSEIF DS=1                                                     %
%       THEN counter=hold, TMR, continue.                               %
%       ELSE counter=hold, TMR, JUMP DSLOWP.                            %
```

```
%                                                                       %
%       --case  DS=1--                                                  %
%DSHIGHP:IF FBREQ THEN Release bus, jump to HOME.                        %
%       ELSEIF DK=1 THEN Clear TMR, continue.                           %
%       ELSE keep TMR true, and wait here.                              %
%                                                                       %
%       Wait two cycles to allow for FIFO write.                        %
%                                                                       %
%       IF FBREQ THEN Release bus, jump to HOME.                        %
%       ELSEIF WT=1 THEN RD=1, counter=hold, disable FIFO's, loop here. %
%       ELSE RD=0, MS=0, DS=0, disable FIFO', continue.                 %
%                                                                       %
%       IF FBREQ THEN Release bus, jump to HOME.                        %
%       ELSEIF DK=0 THEN clear TMR, EOBA, IRQ, continue.                %
%       ELSE TMR, wait here.                                            %
%                                                                       %
%       --case  DS=0--                                                  %
%DSLOWP:IF FBREQ THEN Release bus, jump to HOME.                        %
%       ELSEIF DK=0 THEN clear TMR, CONTINUE.                           %
%       ELSE TMR, wait here.                                            %
%                                                                       %
%       Wait three cycles to ensure FIFO write, then IRQ, EOBA, jump    %
%       home.                                                           %
%                                                                       %
PXFR1LP:IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP PXFR1LP;
        ELSEIF I3               %FEOB%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP EOBPIPE;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK SDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

DATFP1X:        [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

WAIT7:  IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
```

```
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP WAIT7;
        ELSEIF I3               %FEOB%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP EOBPIPE;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK CDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP PXFR1LP;

EOBPIPE:IF /I0          %/FRSS1%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

        IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I1               %FRDS%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX0]
                JUMP DSLOWP;

%       Case  DS=1                                                      %
DSHIGHP:IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I4               %FRDK%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;
```

```
        ELSE    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX0]
                JUMP DSHIGHP;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

WAIT9:  IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CTRLDEF
                HOLDB NEOBA NEG RD MS0 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK
MUX0]
                JUMP WAIT9;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG RD MS0 XWT XDK CDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

WAIT10:         IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF /I4      %FRDK=0%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE IRQ DFFDIS
                HOLDB EOBA NEG NRD MS0 XWT XDK XDS XAS
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR          ;AK MUX1]
                JUMP HOME;
        ELSE    [CTRLDEF
                HOLDB NEOBA NEG RD MS0 XWT XDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX0]
                JUMP WAIT10;

%       Case  DS=0                                                      %
DSLOWP:         IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF /I4              %FRDK=0%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
```

```
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX1]
                CONTINUE;
        ELSE    [CHOLD NSRT NDW DATAINFIFO NFDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP DSLOWP;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX1]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX1]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE IRQ DFFEN
                HOLDB EOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE TMR NCCLR STAT8 NDSAK MUX1]
                JUMP HOME;

%-----------------------------------------------------------------------%
%       Routine DATA_FIFO_PIPELINED_READ_2  (200ns transfers)           %
%       --FAST-- --WRITE or READ--                                      %
%-----------------------------------------------------------------------%
%       This routine does a FASTBUS pipeline block transfer read from   %
%       the FASTBUS crate segment to the FSCC's Output Port Data FIFO,  %
%       and also the Processor Data FIFO.                               %
%                                                                       %
%       Note: DDIR, and SAB must be held high throughout the block      %
%               transfer operation.                                     %
%                                                                       %
%DATFP2:IF FBREQ THEN Release bus, JUMP WAYHOME.                        %
%       ELSEIF WT=1                                                     %
%       THEN Enable FASTBUS xceivers for input, Enable FIFO's, MS=3,    %
%               RD=1, Place local word counter into hold, and loop here.%
%       ELSEIF EOB                                                      %
%       THEN Enable FASTBUS xceivers for input, enable FIFO's, MS=3,    %
%               RD=1, Place local word counter into hold mode, and      %
%               JUMP to EOBPIPE.                                        %
%       ELSE Enable FASTBUS xceivers for input, enable FIFO's, MS=3,    %
%               RD=1, Place local word counter into count, set DS=1,    %
%               and JUMP to DATFP2X.                                    %
%                                                                       %
%PXFR2LP:IF FBREQ THEN Release bus, JUMP WAYHOME.                       %
%       ELSEIF WT=1                                                     %
%       THEN Enable FASTBUS xceivers for input, Enable FIFO's, MS=3,    %
%               RD=1, Place local word counter into hold, and loop here.%
%       ELSEIF EOB                                                      %
%       THEN Enable FASTBUS xceivers for input, enable FIFO's, MS=3,    %
%               RD=1, Place local word counter into hold mode, and      %
%               JUMP to EOBPIPE.                                        %
%       ELSE Enable FASTBUS xceivers for input, enable FIFO's, MS=3,    %
%               RD=1, Place local word counter into count, set DS=1,    %
%               set FCLK, and continue.                                 %
```

```
%                                                                       %
%DATFP2X:Place counter into hold mode, negate FCLK, and                 %
%               continue.  These cycles provide the FSCC's              %
%               FASTBUS cycle time (200ns).                             %
%                                                                       %
%       Continue (add 50ns to cycle time).                              %
%                                                                       %
%       Continue (add 50ns to cycle time).                              %
%                                                                       %
%       IF FBREQ THEN Release bus, jump to HOME.                        %
%       ELSEIF WT=1                                                     %
%       THEN counter=hold, keep both FIFO's enabled and loop here.      %
%       ELSEIF EOB                                                      %
%       THEN counter=hold, keep FIFO's enabled and jump to EOBPIPE.     %
%       ELSE counter=decr, set DS=0, set FCLK, keep FIFO's enabled,     %
%               and continue.                                           %
%                                                                       %
%       Place local word counter in hold, negate FCLK, keep processor   %
%               and oport FIFO's enabled, and Continue.                 %
%               These cycles provide the FSCC's cycle time.             %
%                                                                       %
%       Continue (add 50ns to cycle time).                              %
%                                                                       %
%       Jump to PXFR2LP. (add 50ns to cycle time).                      %
%                                                                       %
PXFR2LP:IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP PXFR2LP;
        ELSEIF I3               %FEOB%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP EOBPIPE;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK SDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

DATFP2X:        [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
```

```
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

WAIT11:         IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP WAIT11;
        ELSEIF I3               %FEOB%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP EOBPIPE;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK CDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP PXFR2LP;

%-----------------------------------------------------------------------%
%       Routine DATA_FIFO_PIPELINED_READ_4  (400ns transfers)           %
%       --FAST-- --READ or WRITE--                                      %
%-----------------------------------------------------------------------%
%       This routine does a FASTBUS pipeline block transfer read from   %
%       the FASTBUS crate segment to the FSCC's Output Port Data FIFO,  %
%       and also the Processor Data FIFO.                               %
%                                                                       %
%       Note: DDIR, and SAB must be held high throughout the block      %
%               transfer operation.                                     %
%                                                                       %
%DATFP4:IF FBREQ THEN Release bus, jump to WAYHOME.                      %
%       ELSEIF WT=1                                                     %
%       THEN Enable FASTBUS xceivers for input, enable data and         %
%               processor FIFO's, MS=3, RD=1, Place local word counter  %
%               into hold, and loop here.                               %
%       ELSEIF EOB                                                      %
```

```
%       THEN Enable FASTBUS xceivers for input, enable FIFO's, Place     %
%               local word counter into hold mode, MS=3, RD=1, and JUMP %
%               to EOBPIPE.                                              %
%       ELSE Enable FASTBUS xceivers for input, enable FIFO's, Place     %
%               local word counter into count, MS=3, RD=1, set DS=1,     %
%               and JUMP to DATFP4X.                                     %
%                                                                        %
%PXFR4LP:IF FBREQ THEN Release bus, jump to WAYHOME.                      %
%       ELSEIF WT=1                                                      %
%       THEN Enable FASTBUS xceivers for input, enable data and          %
%               processor FIFO's, MS=3, RD=1, Place local word counter   %
%               into hold, and loop here.                                %
%       ELSEIF EOB                                                       %
%       THEN Enable FASTBUS xceivers for input, enable FIFO's, Place     %
%               local word counter into hold mode, MS=3, RD=1, and JUMP %
%               to EOBPIPE.                                              %
%       ELSE Enable FASTBUS xceivers for input, enable FIFO's, Place     %
%               local word counter into count, MS=3, RD=1, set DS=1,     %
%               set FCLK, and continue.                                  %
%                                                                        %
%       Place counter into hold mode, negate FCLK, and                   %
%               continue.  These cycles provide the FSCC's               %
%               FASTBUS cycle time (400ns).                              %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       IF FBREQ THEN Release bus, jump to HOME.                         %
%       ELSEIF WT=1                                                      %
%       THEN counter=hold, keep both FIFO's enabled and loop here.       %
%       ELSEIF EOB                                                       %
%       THEN counter=hold, keep FIFO's enabled and jump to EOBPIPE.      %
%       ELSE counter=decr, set DS=0, set FCLK, keep FIFO's enabled,      %
%               and continue.                                            %
%                                                                        %
%       Place local word counter in hold, negate FCLK, keep processor    %
%               and oport FIFO's enabled, and Continue.                  %
%               These cycles provide the FSCC's cycle time.              %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
%                                                                        %
%       Continue (add 50ns to cycle time).                               %
```

```
%                                                                       %
%       Jump to PXFR4LP.(add 50ns to cycle time).                       %
%                                                                       %
PXFR4LP:IF /I6          %FBREQ*%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP PXFR4LP;
        ELSEIF I3               %FEOB%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP EOBPIPE;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK SDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

DATFP4X:        [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;


                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
```

```
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

WAIT12:         IF /I6          %FBREQ*%
        THEN    [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFDIS
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTF;
        ELSEIF I7               %FRWT%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP WAIT12;
        ELSEIF I3               %FEOB%
        THEN    [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                JUMP EOBPIPE;
        ELSE    [COUNT NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK CDS XAS
                PFFEN NSR NSCLK FCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
                CONTINUE;

                [CHOLD NSRT NDW DATAINFIFO FDOE FCOE NIRQ DFFEN
                HOLDB NEOBA NEG RD MS3 XWT XDK XDS XAS
                PFFEN NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT8 NDSAK MUX0]
```

```
                JUMP PXFR4LP;

%-----------------------------------------------------------------------%
%       SLAVE_DATA_INPUT                                                %
%       --SLOW-- --READ--                                               %
%-----------------------------------------------------------------------%
%       Enable FASTBUS AD line receivers, Clear WT, jump to SLVDINX.    %
%                                                                       %
%       IF FBREQ then release bus, jump to INTS.                        %
%       ELSEIF WT=1, THEN wait here.                                    %
%       ELSEIF DS=1 THEN latch FASTBUS AD lines, set DK, DSACK,         %
%               continue.                                               %
%       ELSE (DS=0) THEN latch FASTBUS AD lines, clear DK, DSACK,       %
%               continue.                                               %
%                                                                       %
%       DSACK, Hold latched FASTBUS data, jump WAYHOME.                 %
SLVDINX:IF /I6          %FBREQ%
        THEN    [CTRLDEF
                REL
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP INTS;
        ELSEIF I7       %FRWT%
        THEN    [CDEF NSRT NDW DATAINPROC FDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT13 NDSAK
MUX0]
                JUMP SLVDINX;
        ELSEIF I1       %FRDS%
        THEN    [CDEF NSRT NDW DATAINLATCH FDOE FCOE NIRQ DFFDIS
                NREQ NEOBA NEG NRD MS0 XWT SDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT13 DSAK
MUX0]
                CONTINUE;
        ELSE    [CDEF NSRT NDW DATAINLATCH FDOE FCOE NIRQ DFFDIS
                NREQ NEOBA NEG NRD MS0 XWT CDK XDS XAS
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT13 DSAK
MUX0]
                CONTINUE;

                [CDEF NSRT NDW DATAINHOLD NFDOE FCOE NIRQ DFFDIS
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT13 DSAK MUX1]
                JUMP WAYHOME;

%-----------------------------------------------------------------------%
%       Null                                                            %
%       --FAST-- WRITE or READ--                                        %
%-----------------------------------------------------------------------%
NULLX:          [CTRLDEF
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT12 NDSAK
MUX1]
                JUMP WAYHOME;

%-----------------------------------------------------------------------%
%       End Of Event                                                    %
```

```
%       --SLOW-- --WRITE--                                              %
%-----------------------------------------------------------------------%
%       --Two dummy words--                                             %
%       DW (Data FIFO Write), DEOE, CEOE, DFFEN, Set FASTBUS interface  %
%               transceivers for output to FIFO (Data and Processor).   %
%               DSACK, continue.                                        %
%                                                                       %
%       Hold all lines for one more cycle (make sure that the write     %
%               pulse to the FIFO's is long enough).                    %
%                                                                       %
%       Hold DEOE true for one more cycle while DW is false.  This      %
%               provides the "hold" time for the fifo's.                %
%                                                                       %
%       Set DW, for two cycles to insert another dummy word with DEOE   %
%               false.                                                  %
%                                                                       %
%       Hold DEOE true for one more cycle while DW is false, to         %
%               ensure that the fifo data hold time spec is met.        %
%                                                                       %
EOEX:           [CDEF NSRT DW DATAOUT NFDOE FCOE NIRQ DFFEN
                FBDEF
                PFFDIS NSR NSCLK NFCLK DEOE CEOE NTMR NCCLR STAT14 DSAK MUX1]
                CONTINUE;

                [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFEN
                FBDEF
                PFFDIS NSR NSCLK NFCLK DEOE NCEOE NTMR NCCLR STAT14 NDSAK
MUX1]
                CONTINUE;

                [CDEF NSRT DW DATAOUT NFDOE FCOE NIRQ DFFEN
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT14 NDSAK
MUX1]
                CONTINUE;

                [CDEF NSRT DW DATAOUT NFDOE FCOE NIRQ DFFEN
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT14 NDSAK
MUX1]
                CONTINUE;

                [CDEF NSRT NDW DATAOUT NFDOE FCOE NIRQ DFFEN
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT14 NDSAK
MUX1]
                JUMP HOME;

%-----------------------------------------------------------------------%
%       Interrupt handler                                               %
%-----------------------------------------------------------------------%
%       Being interrupted means that there was a FASTBUS error, or      %
%       the RB (Reset Bus) line has been driven true.  In either case   %
%       the sequencer's assigned task is simply to abort any operation  %
%       it is currently doing, and return to idle state.                %
%                                                                       %
%       Returning DSACK to the 68020, is a prcisely timed function      %
```

```
%       which the sequencer must perform.  DSACK is asserted true        %
%       for exactly two sequencer cycles.  Then it is deasserted, and    %
%       the FBxS* lines may not be sampled for three sequencer cycles.   %
%       What this means is that when a routine wants to return DSACK     %
%       and return to idle, it must do three nop type cycles before      %
%       testing either of the two FBxS* lines.                           %
%                                                                        %
%       The interrupt handler is a strange routine since the IRQ line    %
%       is tested at almost every "IF" instruction there is.  If IRQ is  %
%       true at any of these places, the IH must make sure that DSACK    %
%       is returned with the proper timing in mind.                      %
%                                                                        %
INTS:           [CTRLDEF
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 DSAK MUX1]
                CONTINUE;

                [CTRLDEF
                FBDEF
                PFFDIS NSR NSCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 DSAK
MUX1]
                JUMP WAYHOME;

INTF:           [CTRLDEF
                FBDEF
                PFFDIS NSR SCLK NFCLK NDEOE NCEOE NTMR NCCLR STAT15 NDSAK
MUX1]
                JUMP WAYHOME;
```

FF_CTRL.ABL-2/28/90

```
MODULE FB_CTRL_FF
TITLE 'FB_CTRL_FF        VERSION 14-AUG-89 1000 hrs'
FF_CTRL DEVICE 'P22V10';

FSAS    PIN     1;
FCAS    PIN     2;
FSDS    PIN     3;
FCDS    PIN     4;
FSDK    PIN     5;
FCDK    PIN     6;
!SPOE   PIN     7;
SLCOE   PIN     8;
!SDG    PIN     9;
!SR     PIN     10;
DDIR    PIN     11;
!FCOE   PIN     13;
!DG     PIN     14;
!LCOE   PIN     15;
!POE    PIN     16;
!FB2S   PIN     18;
!FB1S   PIN     19;
!PWRITE         PIN     20;
FDDK    PIN     21;
FDDS    PIN     22;
FDAS    PIN     23;

EQUATIONS

FDAS    =       (FSAS # (FDAS & !FCAS)) & FCOE;

FDDS    =       (FSDS # (FDDS & !FCDS)) & FCOE;

FDDK    =       (FSDK # (FDDK & !FCDK)) & FCOE;

"       Equations to prevent bus conflicts on the IOD bus, by preventing
"       any two enables from being true at one time.
"       POE* is only true during a processor write when selected (SPOE &
"       PWRITE & (FB1S # FB2S)), when the 646's (!(SDG & DDIR)), the Data
"       FIFO (!SR), and the Local Counter (!SLCOE), are not driving the IOD
"       bus.  POE* is also true during processor reads (# (SPOE & !PWRITE).
POE     = ((SPOE & PWRITE) & !(SDG & DDIR) & !SR & !SLCOE # (SPOE & !PWRITE))
& (FB1S # FB2S);

"       In similar fashion to POE*, LCOE is only true during processor reads
"       (!(SPOE & PWRITE)), when the 646's !(SDG & DDIR), and the Data FIFO
"       (!SR), are not enabled.
LCOE    =       SLCOE & !(SDG & DDIR) & !SR & !(SPOE & PWRITE);

"       Again, as above, DG* (646's) can only be true during processor reads
"       (!(SPOE & PWRITE)), when the Data FIFO (!SR), and the Local Counter
"       (!SLCOE), are not enabled.  Also, DG* can be true when driving the
"       FASTBUS AD lines (# (SDG & !DDIR)).
DG      = (SDG & DDIR) & !SR & !SLCOE & !(SPOE & PWRITE) # (SDG & !DDIR);

END FB_CTRL_FF
```

FIFO_CTR.ABL-2/28/90

```
MODULE FIFO_CTRL
TITLE 'FIFO_CTRL        VERSION 28-FEB-90 1500 hrs'
FIFO_CTR DEVICE 'P22V10';

CLK     PIN     1;
DFIFOEN         PIN     2;
PFIFOEN PIN     3;
COPYEN  PIN     4;
FRDK    PIN     5;
FW      PIN     6;
SDCPAB  PIN     7;
FRSS1   PIN     8;

DKT     PIN     15;
FRDKZ   PIN     16;
SA      PIN     17;
SB      PIN     18;
SC      PIN     19;
SD      PIN     20;
DCPAB   PIN     21;

!DW     PIN     23;
!PW     PIN     22;

NSTATE  =       [SA,SB]
PSTATE  =       [SC,SD]
S0      =       [0,0];
S1      =       [0,1];
S2      =       [1,0];
S3      =       [1,1];

EQUATIONS

FRDKZ   :=      FRDK;

DKT     :=      ((!FRSS1 & (FRDK != FRDKZ)) # DKT) & !(NSTATE == 0);

WHEN (NSTATE == S0) & ((!FRSS1 & (FRDK != FRDKZ)) # DKT) THEN NSTATE := S1;

WHEN (NSTATE == S1) THEN NSTATE := S2;

WHEN (NSTATE == S2) THEN NSTATE := S3;

WHEN (NSTATE == S3) THEN NSTATE := S0;

DW      :=      ((((NSTATE == S0) & ((!FRSS1 & (FRDK != FRDKZ)) # DKT))
                # (NSTATE == S1)
                # (NSTATE == S2))
                & (DFIFOEN == 1))
                # (FW & DFIFOEN);

WHEN ((PSTATE == S0) & (!COPYEN)) # ((PSTATE == S3) & (!PFIFOEN))
THEN PSTATE := S0;

WHEN ((PSTATE == S1) & (PFIFOEN)) # ((PSTATE == S0) & (COPYEN))
THEN PSTATE := S1;
```

```
WHEN ((PSTATE == S2) & (!PFIFOEN)) # ((PSTATE == S1) & (!PFIFOEN))
THEN PSTATE := S2;

WHEN ((PSTATE == S3) & (PFIFOEN)) # ((PSTATE == S2) & (PFIFOEN))
THEN PSTATE := S3;

PW      :=      ((((NSTATE == S0) & ((!FRSS1 & (FRDK != FRDKZ)) # DKT))
                # (NSTATE == S1)
                # (NSTATE == S2))
                & (PSTATE == S3))
                # (FW & PFIFOEN);

DCPAB   :=      ((NSTATE == S0) & ((DFIFOEN == 1) # (PFIFOEN == 1))
                & ((!FRSS1 & (FRDK != FRDKZ)) # DKT))
                # SDCPAB;

END FIFO_CTRL
```

HEADAV2.ADF-7/11/90

```
module counter
Gustavo I. Cancelo
Fermilab
10/9/89
title header latch & 12 bit event total counter
%       Changed PL equation from registered output to combinatorial.  RK %
%       11-dec-89  1400hrs%
%       --Added state 7 (CLEAR2) to the state machine to lengthen the clear%
%       pulse to the counter.  11-JUL-90  RK%

PART: EPB1400

INPUTS: DS@25, CS@12, A1@13, RW@36, RESET@9, EOE@8, CK@14, CLK@7

OUTPUTS: CARRY@17, CKE@6, WRITE@37, RC@38
         P0@24, P1@23, P2@22, P3@21, P4@20, P5@19, P6@18, %counter%
         DAT0@26, DAT1@27, DAT2@28, DAT3@29, DAT4@32, DAT5@33, DAT6@34,
DAT7@35
         Q3@39, Q2@40, Q1@1, Q0@2, PL@5

NETWORK:
        DS=INP(DS)
        CS=INP(CS)
        A1=INP(A1)
        RW=INP(RW)
        RESET=INP(RESET)
        EOE=INP(EOE)
        CK=INP(CK)
        CLK=INP(CLK)

% data tranceiver bus %
DAT0,DAT1,DAT2,DAT3,DAT4,DAT5,DAT6,DAT7 = BUSX (IBUS,,XOE)

% low order counter. input register #1 LBUSI %
B0,B1,B2,B3,B4,B5,B6,B7 = LBUSI (IBUS,DS,WE1)

% control register. input register #2 LBUSI %
SE0,SE1,RF,WRT,QE2,QE1,QE0,X0 = LBUSI (IBUS,DS,WE2)

% low order counter.  output register #1 %
IBUS = LBUSO(P0,P1,P2,P3,P4,P5,P6,CARRY,,OE1,RE1)

% status register. output register #2 %
IBUS = LBUSO (S0,S1,Q0,Q1,Q2,Q3,RC,WRITE,,OE2,RE2)

%macrocell connection %
S0 = NOCF(S0c)
S1 = NOCF(S1c)

RC,RC = RORF(RCd,CLK,,,)          %software reset macrocell%

Q3,Q3 = RORF(Q3d,CLK,RESET,,)

Q2,Q2 = RORF(Q2d,CLK,RESET,,)
Q1,Q1 = RORF(Q1d,CLK,RESET,,)
Q0,Q0 = RORF(Q0d,CLK,RESET,,)
```

```
%PL,PL = RORF(PLd,CLK,RES,,)        %%enables Preload/Transparent modes for
HEAD1%
PL,PL = COCF(PLd,)     %Changed from registered to combinatorial rk 11-dec-89%

WRITE,WRITE = RORF(WRITEd,CLK,RESET,,WOE)  %control FIFO's write pulse%

CKE = RONF(CKEd,CLK,RES,,)              %Fastbus clock enable for HEAD1%

% counter macrocells %
P0,P0 = TOTF(P0t,RELO,RES,,)
P1,P1 = TOTF(P1t,RELO,RES,,)
P2,P2 = TOTF(P2t,RELO,RES,,)
P3,P3 = TOTF(P3t,RELO,RES,,)
P4,P4 = TOTF(P4t,RELO,RES,,)
P5,P5 = TOTF(P5t,RELO,RES,,)
P6,P6 = TOTF(P6t,RELO,RES,,)

CARRY,CARRY = COCF(CARRYc,)

EQUATIONS:
          RE1 = /DS * /CS * RW * /A1;       % low_counter read enable %
          RE2 = /DS * /CS * RW *  A1;       % status register read enable %
          % RE = RE1 + RE2  read function is activated for either RE1 or RE2%

          WE1 = /RW * /CS * /A1;            % low_counter write enable %
          WE2 = /RW * /CS * A1;             % control register write enable %

          OE1 = /A1 + A1;          % output registers tri state always enabled

          OE2 = /A1 + A1;

          XOE = /CS * RW;

% counter equations %
% INCR mode == S1 * !s0 %
% if (mode=INCR) RELO=ck   else RELO=clk   %

RELO =  Q3 * /Q2 * Q1 * /Q0 * CK + /Q1 *  CLK;

RES = RC;

P0t = Q3 * /Q2 *  Q1 * /Q0 *  S1 * /S0% counter increments if (mode==INCR)  %
 + Q3 * /Q2 * /Q1 *  Q0 *  B0 * /P0   % preload mode !S1 * S0, state mach=9 %
 + Q3 * /Q2 * /Q1 *  Q0 * /B0 *  P0   % toggle if (mode==LOAD & Bi != Pi)   %
 + Q3 *  Q2 * /Q1 *  Q0 *  B0 * /P0   % transparent mode S1 * S0, SM=13     %
 + Q3 *  Q2 * /Q1 *  Q0 * /B0 *  P0;  % toggle if (mode==WRDT & Bi != Pi)   %
P1t = Q3 * /Q2 *  Q1 * /Q0 *  S1 * /S0 *  P0
    + Q3 * /Q2 * /Q1 *  Q0 *  B1 * /P1 + Q3 * /Q2 * /Q1 *  Q0 * /B1 *  P1
    + Q3 *  Q2 * /Q1 *  Q0 *  B1 * /P1 + Q3 *  Q2 * /Q1 *  Q0 * /B1 *  P1;
P2t = Q3 * /Q2 *  Q1 * /Q0 *  S1 * /S0 *  P0 *  P1
    + Q3 * /Q2 * /Q1 *  Q0 *  B2 * /P2 + Q3 * /Q2 * /Q1 *  Q0 * /B2 *  P2
    + Q3 *  Q2 * /Q1 *  Q0 *  B2 * /P2 + Q3 *  Q2 * /Q1 *  Q0 * /B2 *  P2;
P3t = Q3 * /Q2 *  Q1 * /Q0 *  S1 * /S0 *  P2 *  P1 *  P0
    + Q3 * /Q2 * /Q1 *  Q0 *  B3 * /P3 + Q3 * /Q2 * /Q1 *  Q0 * /B3 *  P3
    + Q3 *  Q2 * /Q1 *  Q0 *  B3 * /P3 + Q3 *  Q2 * /Q1 *  Q0 * /B3 *  P3;
P4t = Q3 * /Q2 *  Q1 * /Q0 *  S1 * /S0 *  P3 * P2 * P1 * P0
```

```
    + Q3 * /Q2 * /Q1 *  Q0 *  B4 * /P4 + Q3 * /Q2 * /Q1 *  Q0 * /B4 *  P4
    + Q3 *  Q2 * /Q1 *  Q0 *  B4 * /P4 + Q3 *  Q2 * /Q1 *  Q0 * /B4 *  P4;
P5t = Q3 * /Q2 *  Q1 * /Q0 *  S1 * /S0 *  P4 *  P3 *  P2 *  P1 *  P0
    + Q3 * /Q2 * /Q1 *  Q0 *  B5 * /P5 + Q3 * /Q2 * /Q1 *  Q0 * /B5 *  P5
    + Q3 *  Q2 * /Q1 *  Q0 *  B5 * /P5 + Q3 *  Q2 * /Q1 *  Q0 * /B5 *  P5;
P6t = P6 * /P6;

S0c = SE0;
S1c = SE1;

% state machine, controls the counter's operation  %
% states: Off_line=0, Idle=8, Preload=9, Incr=10, Write=11, %
%         Pass=12, pass_1=13, Clear=14, DS_High=15 %
Q0d =  Q3 * /Q2 * /Q1 * /Q0 * /S1 *  S0 +  Q3 * /Q2 * /Q1 *  Q0 * /S1 *  S0
    +  Q3 *  Q2 * /Q1 * /Q0 * /DS * /A1 * /RW * /CS +  Q3 *  Q2 * /Q1 *  Q0
    +  Q3 *  Q2 *  Q1 *  Q0 +  Q3 * /Q2 *  Q1 * /Q0 *  S1 * /S0 *  EOE
    +  Q3 *  Q2 *  Q1 * /Q0 * /EOE;
Q1d =  Q3 *  Q2 * /Q1 *  Q0 *  DS +  Q3 *  Q2 *  Q1 *  Q0
    +  Q3 * /Q2 * /Q1 * /Q0 *  S1 * /S0 +  Q3 * /Q2 *  Q1 *  Q0 *  EOE
    +  Q3 *  Q2 *  Q1 * /Q0 +  /Q2 *  Q1 * /Q0 *  S1 * /S0
    + /Q3 *  Q2 *  Q1 *  Q0;
Q2d =  Q3 * /Q2 * /Q1 * /Q0 *  S1 *  S0 +  Q3 *  Q2 * /Q1 * /Q0 *  S1 *  S0
    +  Q3 *  Q2 * /Q1 * /Q0 * /DS +  Q3 *  Q2 * /Q1 *  Q0
    +  Q3 * /Q2 *  Q1 *  Q0 *  S1 *  S0 * /EOE +  Q3 * /Q2 *  Q1 *  Q0 *  EOE
    +  Q3 *  Q2 *  Q1 * /Q0;
Q3d = /CS * /RW * /DS * A1
    +  Q3 * /Q2 * /Q1 * /Q0          %Q=8%
    +  Q3 * /Q2 * /Q1 *  Q0          %Q=9%
    +  Q3 * /Q2 *  Q1 * /Q0          %Q=A%
    +  Q3 * /Q2 *  Q1 *  Q0          %Q=B%
    +  Q3 *  Q2 * /Q1 * /Q0          %Q=C%
    +  Q3 *  Q2 * /Q1 *  Q0          %Q=D%
    +  Q3 *  Q2 *  Q1 * /Q0 *  EOE   %Q=E * EOE=1%
    +  Q3 *  Q2 *  Q1 *  Q0          %Q=F%
    + /Q3 *  Q2 *  Q1 *  Q0;         %Q=7%

PLd =  Q3 * /Q2 * /Q1 * /Q0 * /S1 *  S0 +  Q3 * /Q2 * /Q1 * /Q0 *  S1 *  S0
    +  Q3 *  Q2 * /Q1 * /Q0 *  S1 *  S0 +  Q3 *  Q2 * /Q1 *  Q0
    +  Q3 *  Q2 *  Q1 *  Q0 +  Q3 * /Q2 *  Q1 *  Q0 *  S1 *  S0 * /EOE;

WOE =  Q3 + /Q3 *  Q2 *  Q1 *  Q0;

WRITEd = /(Q3 *  Q2 *  Q1 *  Q0 +   Q3 *  Q2 * /Q1 *  Q0 *  DS
       +   Q3 * /Q2 *  Q1 * /Q0 *  S1 * /S0 *  EOE
       +   Q3 * /Q2 *  Q1 *  Q0 *  S1 * /S0);    %CAMBIE EOE POR S1 * !S0%

CLR =  Q3 * /Q2 * /Q1 * Q0;

CARRYc = P5 * P4 * P3 * P2 * P1 * P0;

CKEd =  Q3 * /Q2 * /Q1 * /Q0 *  S1 * /S0 +  Q3 * /Q2 *  Q1 * /Q0 *  S1 * !S0
* /EOE + Q3 * Q2 *  Q1 * /Q0 *  S1 * !S0 + /Q3 *  Q2 *  Q1 *  Q0 *  S1 * /S0;

RCd =   Q3 *  Q2 *  Q1 * /Q0 + /Q3 *  Q2 *  Q1 *  Q0 + RESET + RF;

ZEROc =  A1 * /A1;
END$
```

HEADBV1.ADF-12/9/89

```
module counter
Gustavo I. Cancelo
Fermilab
10/9/89
 Added counter reset (RC) input.      Richard Kwarciany  17-NOV-89
 Removed HAB1 from Header pin equations,  Header outputs are now always
enabled.
  Reversed WE2, with WE1, RE2, with RE1, and OE2 with OE1, in input and
output
  register equations.   9-dec-89  rk.
title header latch & 12 bit event total counter

PART: EPB1400

INPUTS: DS@25, CS@12, A1@13, RW@36, RESET@9, CK@14, CKE@8, PL@5, CARRY@6,
CLK@7,
        RC@15

OUTPUTS: H0@2, H1@1, H2@40, H3@39, H4@38  %header%
         P6@24, P7@23, P8@22, P9@21, P10@20, P11@19, %counter%
         DAT0@26, DAT1@27, DAT2@28, DAT3@29, DAT4@32, DAT5@33, DAT6@34,
DAT7@35

NETWORK:
        DS=INP(DS)
        CS=INP(CS)
        A1=INP(A1)
        RW=INP(RW)
        RESET=INP(RESET)
        CKE=INP(CKE)
        PL=INP(PL)
        CARRY=INP(CARRY)
        CK=INP(CK)
        CLK=INP(CLK)
        RC=INP(RC)      %added 17-NOV-89  RK%

% data tranceiver bus %
DAT0,DAT1,DAT2,DAT3,DAT4,DAT5,DAT6,DAT7 = BUSX (IBUS,,XOE)

% header. input register #2 LBUSI %
XE1,XE2,HE0,HE1,HE2,HE3,HE4,XE3 = LBUSI (IBUS,DS,WE2)   %diferente nombre%

% counter's 6 highest bits. input register #1 LBUSI %
B0,B1,B2,B3,B4,B5,B6,B7 = LBUSI (IBUS,DS,WE1)

% output register #2 %
IBUS = LBUSO (X1,X2,H0,H1,H2,H3,H4,X3,,OE2,RE2)

% output register #1 %
IBUS = LBUSO (P6,P7,P8,P9,P10,P11,N1,N2,,OE1,RE1)

% header outputs%      %Removed HAB1 from OE   9-dec-89  RK%
H0,H0 = COCF (HE0,)
H1,H1 = COCF (HE1,)
H2,H2 = COCF (HE2,)
H3,H3 = COCF (HE3,)
```

```
H4,H4 = COCF (HE4,)

% junk. just to give a name to unused macrocells %
X1 = NOCF(X1c)
X2 = NOCF(X2c)
X3 = NOCF(X3c)

% counter outputs %            %changed RESET to CRESET in counter eqs.%
                               %17-nov-89  RK%
P6,P6 = TOTF(P6t,RELO,CRESET,,)
P7,P7 = TOTF(P7t,RELO,CRESET,,)
P8,P8 = TOTF(P8t,RELO,CRESET,,)
P9,P9 = TOTF(P9t,RELO,CRESET,,)
P10,P10 = TOTF(P10t,RELO,CRESET,,)
P11,P11 = TOTF(P11t,RELO,CRESET,,)

N1 = NOCF(N2c)
N2 = NOCF(N2c)

RELO = NOCF(RELOc)

CRESET = NOCF(CRESETc)

EQUATIONS:
          RE1 = !DS * !CS * RW * !A1;
          RE2 = !DS * !CS * RW *  A1;
% RE = RE1 + RE2 %

          WE1 = !RW * !CS * !A1;
          WE2 = !RW * !CS * A1;

          OE1 = !DS * !CS * RW * !A1;
          OE2 = !DS * !CS * RW * A1;

          XOE = !CS * RW;

           HAB1 = !CS;                 9-DEC-89 RK%

          CRESETc = RESET + RC;                %added 17-NOV-89  RK%

% counter equations %

RELOc = CLK * !CKE
      + CK  *  CKE;

P6t = !PL * CKE * CARRY
     + B0 * !P6 * PL
     + !B0 * P6 * PL;

P7t = P6 * !PL * CKE * CARRY
    + B1 * !P7 * PL
    + !B1 * P7 * PL;

P8t = P6 * P7 * !PL * CKE * CARRY
    + B2 * !P8 * PL
    + P8 * !B2 * PL;
```

```
P9t = P6 * P7 * P8 * !PL * CKE * CARRY
    + B3 * !P9 * PL
    + P9 * !B3 * PL;

P10t = P9 * P8 * P7 * P6 * !PL * CKE * CARRY
     + B4 * !P10 * PL
     + P10 * !B4 * PL;

P11t = P10 * P9 * P8 * P7 * P6 * !PL * CKE * CARRY
     + B5 * !P11 * PL
     + P11 * !B5 * PL;

X2c = PL * !PL;

X3c = PL * !PL;

N1c = PL * !PL;

N2c = PL * !PL;

X1c = PL * !PL;

END$_
```

INTF3.ADF-9/06/90

```
module oport_interface
Gustavo Cancelo
Fermilab
10/5/89                INTF3
-Changed RES equation.   Rick Kwarciany  16-NOV-89
-Changed clock on START, INHIB, MODE, RES, pins, from RELO (CLK2C), to CS;
-Changed polarity of WE2.  Kwar  16-DEC-89
-Changes from INTF2.... added RPERMIN, and RWAIT to register PERMIN, and
       WAIT signals.  Kwar  4-20-90
-Changed interrupt, and interrupt mask equations.  kwar  9-4-90
-Changed INCOMP interrupt to PERMIN interrupt.  kwar 9-4-90
-Added clear to IRQ line.  kwar 9-6-90

title interface the Oport controller to the CPU system

PART:EPB1400

INPUTS: CS@12,RW@36,DS@25,A0@13,CLK@14,CLK2@7,RESET@9
        PERMIN@8,PRMOUT@40,STPACK@18,ERROR@17,SM3@21,SM2@22,SM1@23,SM0@24
        WAIT@5

OUTPUTS: DB0@26,DB1@27,DB2@28,DB3@29,DB4@32,DB5@33,DB6@34,DB7@35
         MODE2@2,MODE0@4,MODE1@3,RES@1,STPMK@39,PERMINMK@38,ERRMK@37
         IRQ@16,RPERMIN@19,RWAIT@6,IRQS@20

NETWORK:

A0=INP(A0)
CLK=INP(CLK)
CLK2=INP(CLK2)
CS=INP(CS)
RW=INP(RW)
DS=INP(DS)
RESET=INP(RESET)

SM3=INP(SM3) %SM3,SM2,SM1,SM0: OPORT's state machine status%
SM2=INP(SM2)
SM1=INP(SM1)
SM0=INP(SM0)
PERMIN=INP(PERMIN)
PRMOUT=INP(PRMOUT)
STPACK=INP(STPACK)
ERROR=INP(ERROR)
WAIT=INP(WAIT)

MODE0,MODE0=RORF(MODE0d,RELO,RESET,,)
MODE1,MODE1=RORF(MODE1d,RELO,RESET,,)
MODE2,MODE2=RORF(MODE2d,RELO,RESET,,)
RES,RES=RORF(RESd,CLK2,,,)

IRQ,IRQ=RORF(IRQd,CLK,,,)
STPMK,STPMK=RORF(STPMKc,RELO,RESET,,)
PERMINMK,PERMINMK=RORF(PERMINMKc,RELO,RESET,,)
ERRMK,ERRMK=RORF(ERRMKc,RELO,RESET,,)
IRQS,IRQS=COCF(IRQSd,)
```

```
RPERMIN,RPERMIN=RORF(PERMINd,CLK,RESET,,)
RWAIT,RWAIT=RORF(WAITd,CLK2,RESET,,)

%high add/data byte tranceiver w/tri_state enable%
DB0,DB1,DB2,DB3,DB4,DB5,DB6,DB7 = BUSX (IBUS,,XOE)

%high address byte input register w/tri_state enable 2%
MODE01,MODE11,MODE21,RES1,OR10,STPMK1,PERMINMK1,ERRMK1 = LBUSI (IBUS,,WE2)

%high data byte output register 1 %
IBUS = LBUSO (SM0,SM1,SM2,SM3,IRQS,RPERMIN,STPACK,ERROR,,OLE1,RE1)

%high data byte output register 2 %
IBUS = LBUSO (MODE0,MODE1,MODE2,RES,PRMOUT,STPMK,PERMINMK,ERRMK,,OLE2,RE2)


EQUATIONS:

          RELO = !DS * !CS * !RW * A0 * CLK2;

          RE1 = !DS * !CS * RW * !A0;
          RE2 = !DS * !CS * RW *  A0;
          % RE = RE1 + RE2 %

          WE2 = (!RW * !CS * A0);     %Changed polarity 16-dec-89 rk%

           OLE1 = CLK;
           OLE2 = CLK2;

          XOE = !CS * RW;


MODE0d = MODE01;
MODE1d = MODE11;
MODE2d = MODE21;

!RESd = RESET + RES1;
!IRQd = (STPACK * STPMK)
       + (!PERMIN * PERMINMK)
       + (ERROR * ERRMK)
       + (!IRQ * RES * !(!DS * !CS * RW * !A0));

STPMKc = STPMK1;
PERMINMKc = PERMINMK1;
ERRMKc = ERRMK1;
IRQSd = IRQ;

WAITd = WAIT;
PERMINd = PERMIN;

END$
```

LC_HIGH.ABL-2/5/90

```
MODULE LOCAL_COUNTER_HIGH
TITLE 'LOCAL_COUNTER_HIGH  PC2 VERSION 5-FEB-90 0900 hrs'
LC_HIGH DEVICE 'P22V10';

CLK     PIN     1;
LC0     PIN     2;
LC1     PIN     3;

!CIN    PIN     22;
!TC     PIN     23;

!OE     PIN     13;

I0      PIN     4;
I1      PIN     5;
I2      PIN     6;
I3      PIN     7;
I4      PIN     8;
I5      PIN     9;

Q0      PIN     16;
Q1      PIN     17;
Q2      PIN     18;
Q3      PIN     19;
Q4      PIN     20;
Q5      PIN     21;

DATA    =       [Q5..Q0];
INPUT   =       [I5..I0];
MODE    =       [LC1,LC0];
HOLD    =       [0,.X.];
WRITE   =       [1,0];
COUNT   =       [1,1];
MATCH   =       0;

EQUATIONS


ENABLE DATA     =       OE;

WHEN (MODE == COUNT) & (CIN == 1) THEN DATA:=DATA-1;

WHEN (MODE == WRITE) THEN DATA:=INPUT;

WHEN (MODE == HOLD) # ((MODE == COUNT) & (CIN == 0)) THEN DATA:=DATA;

TC      =       (DATA == MATCH);

ENABLE CIN=0;

END LOCAL_COUNTER_HIGH
```

LC_LOW.ABL-2/5/90

```
MODULE LOCAL_COUNTER_LOW
TITLE 'LOCAL_COUNTER_LOW   PC2 VERSION 5-FEB-90  0900hrs'
LC_LOW DEVICE 'P22V10';

CLK     PIN     1;
LC0     PIN     2;
LC1     PIN     3;

!COUT   PIN     22;
!TC     PIN     23;

!OE     PIN     13;

I0      PIN     4;
I1      PIN     5;
I2      PIN     6;
I3      PIN     7;
I4      PIN     8;
I5      PIN     9;

Q0      PIN     16;
Q1      PIN     17;
Q2      PIN     18;
Q3      PIN     19;
Q4      PIN     20;
Q5      PIN     21;

DATA    =       [Q5..Q0];
INPUT   =       [I5..I0];
MODE    =       [LC1,LC0];
HOLD    =       [0,.X.];
WRITE   =       [1,0];
COUNT   =       [1,1];
MATCH   =       0;

EQUATIONS


ENABLE DATA     =       OE;

WHEN (MODE == COUNT) THEN DATA:=DATA-1;

WHEN (MODE == WRITE) THEN DATA:=INPUT;

WHEN (MODE == HOLD) THEN DATA:=DATA;

TC      =       (DATA == MATCH);

COUT    =       (MODE == COUNT) & (DATA == 0);

END LOCAL_COUNTER_LOW
```

OPOV7.02.ASM-6/4/90

```
OPORT State Machine SAM format file
VDAS controller
V7.02  Monday 4-JUN-90        1530hrs
Change from V3: Keep oe true during reset to prevent 485 drivers from
                oscillating on cable.
Change from V4: Change Start, and Step inputs to Mode0, and Mode1.
                Four possible modes are now:
                M1      M0      Mode
                0       0       Hold
                0       1       Start
                1       0       Force_Start
                1       1       CPU
Change from V5:Reverse polarity of Strobe output, and remove Strobe, and
                OE from RESET state.  Now a negative going ~25 ns pulse
                will be generated by a one shot on the AUX card from the
                rising edge of the the VDAS controller Strobe output.
                Change made 21-Mar-90  by Richard Kwarciany.
Change from V6:Inverted PERMIN/PERMOUT signals as per E771's request, and
                corrected error in macros for vdas_oe line.  24-Mar-90  RK
Change from V6.01:Change polarity of WAIT input to match VDAS's /WAIT signal.
                10-APR-90, RK.
Change from V6.02: Change INHIB line to MODE2.  Encode the three mode lines
                to allow for two new modes.  The new modes are; EMPO, and PO.
                (Event With Manual Permit Out, and Permit Out).  The Eight
                possible modes are now:
                Mode            M2      M1      M0
                Hold            0       0       0
                Event           0       0       1
                Force_Event     0       1       0
                CPU (Step)      0       1       1
                Inhibit         0       1       0
                EMPO            1       0       1
                PO              1       1       0
                Reserved        1       1       1
Change from V7.00: Added two new states; Active2, and Wait3. Active2 is the
                same as Active, except that EOE is not tested. This state will
                only be used on the first data cycle, where EOE is not valid.
                Wait3 is the same as Wait, except that it returns to Active2
                instead of Active.
Change from V7.01: Made changes to fix bugs in Step mode operation.

PART: EPS448

INPUTS: wait@2, eoe@3, def@4, cef@5, permin@9, mode2@10, mode1@11, mode0@12

OUTPUTS: Q3@25, Q2@24, Q1@23, Q0@22,
        lck@13, strobe@20, /cfifor@17, /dfifor@18,
        /loe@14, error@26, stpack@27, /prmout@28,
        /lclr@16, vdas_oe@1

MACROS:  % outNo   state     outputs %
          RESET = "0000    0011100100"      %LCLR*%
           OUT0  = "0000    0011000101"      %LCLR*,LOE*,VDAS_OE%
           OUT1  = "0001    0001000111"      %CFR*,LOE*,VDAS_OE%
           OUT2  = "0010    0011000101"      %LCLR*,LOE*,VDAS_OE%
           OUT3  = "0011    0001000111"      %CFR*,LOE*,VDAS_OE%
```

```
            OUT4  = "0100    1010000111"      %LCK,DFR*,LOE*,VDAS_OE%
            OUT5  = "0101    0011001011"      %STPACK,LOE*,VDAS_OE,PERMOUT*%
            OUT6  = "0110    0001000111"      %CFR*,LOE*,VDAS_OE%
            OUT7  = "0111    0011110100"      %ERROR,LCLR*%
            OUT8  = "1000    0011100110"      %no outputs%
            OUT9A = "1001    0011000101"      %LCLR*,LOE*,VDAS_OE%
            OUT9B = "1001    0010000001"      %DFR*,PERMOUT*,LOE*,VDAS_OE%
            OUT9C = "1001    0011000001"      %PERMOUT*,LCLR*,LOE*,VDAS_OE%
            OUT9D = "1001    0010100100"      %DFR*,LCLR*%
            OUT10 = "1010    1011000111"      %LCK,LOE*,VDAS_OE%
            OUT11 = "1011    1011000111"      %LCK,LOE*,VDAS_OE%
            OUT12 = "1100    0110000111"      %STROBE,DFR*,LOE*,VDAS_OE%
            OUT13 = "1101    0011001011"      %STPACK,LOE*,VDAS_OE,PERMOUT*%
            OUT14 = "1110    1011000111"      %LCK,LOE*,VDAS_OE%
            OUT15 = "1111    1011000111"      %LCK,LOE*,VDAS_OE%

EQUATIONS:
        MODE_0  =       /mode2  *  /model  *  /mode0;
        %hold%
        MODE_1  =       /mode2  *  /model  *   mode0;
        %Event%
        MODE_2  =       /mode2  *   model  *  /mode0;
        %Force_Event%
        MODE_3  =       /mode2  *   model  *   mode0;
        %CPU%
        MODE_4  =        mode2  *  /model  *  /mode0;
        %Inhibit%
        MODE_5  =        mode2  *  /model  *   mode0;
        %EMPO%
        MODE_6  =        mode2  *   model  *  /mode0;
        %PO%
        MODE_7  =        mode2  *   model  *   mode0;
        %Reserved%

PROGRAM:
%*************************************************************************%
%    State = IDLE, CK=L, strobe=H, cfifor=H, dfifor=H, oe=H, error=0,     %
%    stpack=L,  prmout=L                                                  %
%*************************************************************************%
        [RESET] CONTINUE;

RES:    IF (MODE_1 + MODE_3 + MODE_5)          %Event + CPU + EMPO%
        THEN [RESET] CONTINUE;
        ELSEIF MODE_2 THEN [OUT0] JUMP IDLES;
        %Force_Event%
        ELSEIF MODE_4 THEN [OUT8] JUMP INHIBS;
        %inhibit%
        ELSE [RESET] JUMP RES;

PINWAIT1:IF /permin + MODE_2                   %permin* + Force_Event%
        THEN [OUT0] JUMP IDLES;
        ELSEIF MODE_4 THEN [OUT8] JUMP INHIBS;
        %inhibit%
        ELSE [RESET] JUMP PINWAIT1;
```

```
%*************************************************************************%
%    State = IDLE, CK=L, strobe=H, cfifor=H, dfifor=H, oe=H, error=0,      %
%    stpack=L,  prmout=L                                                  %
%    IF inhib THEN State=INHIB (8)                                        %
%    ELSEIF ((Event + Force_Event + CPU + EMPO) * cef) State=SETUP (2)    %
%    ELSE state=IDLES (0)                                                 %
%*************************************************************************%
IDLES:  IF MODE_4                                  %Inhibit%
        THEN [OUT8] JUMP INHIBS;
        ELSEIF (MODE_1 + MODE_2 + MODE_3 + MODE_5) * cef
        THEN [OUT2] JUMP SETUPS;
        ELSE [OUT0] JUMP IDLES;

%*************************************************************************%
%    State = INHIB, CK=L, strobe=H, cfifor=H, dfifor=H, oe=H, error=0,    %
%    stpack=L,  prmout=L                                                  %
%*************************************************************************%
INHIBS: IF MODE_4 THEN [OUT8] JUMP INHIBS;   %Inhibit%
        ELSE [OUT0] JUMP RES;

%*************************************************************************%
%    State = SETUP, CK=L, strobe=H, cfifor=H, dfifor=H, oe=H, error=0,    %
%    stpack=L,  prmout=L                                                  %
%    IF /wait State=ERROR (7), error go to H                              %
%    ELSE State=CONTS (6), cfifor=L, oe=L                                 %
%*************************************************************************%
SETUPS: IF (/wait) THEN [OUT7] JUMP ERRORS;
        ELSEIF MODE_3 THEN [OUT3] JUMP STEPS;
        %Step%
        ELSE [OUT6] JUMP CONTS;

%*************************************************************************%
%    State = CONTS, CK=L, strobe=H, cfifor=L, dfifor=H, oe=L, error=0,    %
%    stpack=L,  prmout=L                                                  %
%    IF cef State=ACTIVE (14), ck go to H, strobe=L, cfifor=H             %
%*************************************************************************%
CONTS: IF MODE_0 THEN [OUT4] JUMP HOLD1S;    %hold%
       ELSE [OUT14] JUMP ACTIVE2S;

%*************************************************************************%
%    State = ACTIVES, CK=H, strobe=L, cfifor=H, dfifor=H, oe=L, error=0,  %
%    stpack=L,  prmout=L                                                  %
%    IF /WAIT THEN WAITS                                                  %
%    IF eoe State=EOES (9), ck go to H, oe=H,                             %
%    ELSEIF def THEN State=DATA (12), ck=L, strobe=H, dfifor=H,           %
%    ELSE State = ACTIVES                                                 %
%*************************************************************************%
ACTIVES: IF (/wait) THEN [OUT15] JUMP WAITS;
         ELSEIF (eoe) THEN [OUT9A] JUMP CHKMODE;
         ELSEIF (def) THEN [OUT12] JUMP DATAS;
         ELSE [OUT14] JUMP ACTIVES;

ACTIVE2S:IF (/wait) THEN [OUT15] JUMP WAIT3S;
         ELSEIF (def) THEN [OUT12] JUMP DATAS;
         ELSE [OUT14] JUMP ACTIVE2S;
```

```
%*************************************************************************%
%    State = DATAS, CK=L, strobe=H, cfifor=H, dfifor=H, oe=L, error=0,     %
%    stpack=L,  prmout=L                                                  %
%    IF /start THEN HOLD1S.                                               %
%    ELSE State=ACTIVE (14), ck go to H, strobe=L, cfifor=H               %
%*************************************************************************%
DATAS: IF MODE_0 THEN [OUT4] JUMP HOLD1S;    %hold%
       ELSE [OUT14] JUMP ACTIVES;

%*************************************************************************%
%    State = WAITS, CK=H, strobe=L, cfifor=H, dfifor=H, oe=L, error=0,     %
%    stpack=L,  prmout=L                                                  %
%    IF cef State=ACTIVE (14), same outputs                               %
%*************************************************************************%
WAITS: IF (wait) THEN [OUT14] JUMP ACTIVES;  %/wait*%
       ELSE [OUT15] JUMP WAITS;

WAIT3S:IF (wait) THEN [OUT14] JUMP ACTIVE2S; %/wait*%
       ELSE [OUT15] JUMP WAIT3S;

%*************************************************************************%
%    State = HOLD1S, CK=H, strobe=L, cfifor=H, dfifor=H, oe=L, error=0,    %
%    stpack=L,  prmout=L                                                  %
%    IF inhib State=INHIBS (8), ck go to L, strobe=H, oe=H                %
%    ELSEIF (step * !start) State=IDLE, ck goto L, strobe=H, oe=H         %
%    ELSEIF (start) State=ACTIVE, same outputs                            %
%*************************************************************************%
HOLD1S: IF MODE_4 THEN [OUT8] JUMP INHIBS;
        ELSEIF MODE_3 THEN [OUT0] JUMP IDLES;
        %Step%
        ELSEIF MODE_1 + MODE_2 + MODE_5
        %Event + Force_Event + EMPO%
        THEN [OUT12] JUMP DATAS;
        ELSE [OUT4] JUMP HOLD1S;

%*************************************************************************%
%    State = STEPS, CK=L, strobe=H, cfifor=L, dfifor=H, oe=L, error=0,     %
%    stpack=L,  prmout=L                                                  %
%    IF /wait State=WAIT2S (1), same outputs                              %
%    ELSEIF (!cef) State=STPACK1 cfifor=H, oe=H, stpack=H                 %
%    ELSE State=IDLE, stpack goto H, cfifor=H, oe=H                       %
%*************************************************************************%
STEPS: IF (/wait) THEN [OUT1] JUMP WAIT2S;
       ELSEIF (!cef) THEN [OUT5] JUMP STPACK1S;
       ELSE [OUT11] JUMP SDATAS;

%*************************************************************************%
%    State = SDATAS, CK=H, strobe=L, cfifor=H, dfifor=H, oe=L, error=0,    %
%    stpack=L,  prmout=L                                                  %
%    IF (!step) State=HOLD2S, same outputs                                %
%    ELSEIF (cef) State=STEPS (3), ck go to L, strobe=H, cfifor=L         %
%*************************************************************************%
SDATAS: IF MODE_3 THEN [OUT3] JUMP STEPS;    %step%
        ELSE [OUT10] JUMP HOLD2S;
```

```
%*****************************************************************************%
%    State = HOLD2S, CK=H, strobe=L, cfifor=H, dfifor=H, oe=L, error=0,       %
%    stpack=L,  prmout=L                                                      %
%    IF inhib State=INHIBS (8), ck go to L, strobe=H, oe=H                    %
%    ELSEIF (/step * /start) State=IDLES, ck goto L, strobe=H, oe=H           %
%    ELSEIF (start * step) State=SDATAS, same outputs                         %
%*****************************************************************************%
HOLD2S: IF MODE_4 THEN [OUT8] JUMP INHIBS;   %Inhibit%
        ELSEIF MODE_1 + MODE_2 + MODE_5
        %Event + Force_Event + EMPO%
        THEN [OUT0] JUMP IDLES;
        ELSEIF MODE_3 THEN [OUT11] JUMP SDATAS;
        %step%
        ELSE [OUT10] JUMP HOLD2S;

%*****************************************************************************%
%    State = WAIT2S, CK=L, strobe=H, cfifor=L, dfifor=H, oe=L, error=0,       %
%    stpack=L,  prmout=L                                                      %
%    IF cef State=STEP (3), same outputs                                      %
%*****************************************************************************%
WAIT2S: IF (wait) THEN [OUT3] JUMP STEPS;
        ELSE [OUT1] JUMP WAIT2S;

%*****************************************************************************%
%     EOE goto IDLE                                                           %
%*****************************************************************************%
EOES:   [OUT9C] CONTINUE;

PINWAIT2:IF (/permin) THEN [OUT0] JUMP IDLES;
        %/permin*%
        ELSEIF MODE_0 THEN [RESET] JUMP RES;  %hold%
        ELSEIF MODE_4 THEN [OUT8] JUMP INHIBS;
        %inhibit%
        ELSE [RESET] JUMP PINWAIT2;

%*****************************************************************************%
%     STPACK1S goto STPACK2                                                   %
%*****************************************************************************%
STPACK1S:  [OUT13] JUMP STPACK2S;

%*****************************************************************************%
%     STPACK2S goto PINWAIT2                                                  %
%*****************************************************************************%
STPACK2S:   [OUT0] JUMP PINWAIT2;
```

```
%*************************************************************************%
%     State = ERRORS, CK=L, strobe=H, cfifor=H, dfifor=H, oe=H, error=1,     %
%     stpack=L,  prmout=L                                                   %
%     stay in ERRORS until hardware Reset                                   %
%*************************************************************************%
ERRORS: [OUT7] JUMP ERRORS;

CHKMODE:IF MODE_6 + MODE_1 + MODE_2          %PO + Event + Force_Event%
        THEN [OUT9B] JUMP EOES;
        ELSEIF MODE_0 THEN [OUT9D] JUMP CLEANUP;
        %hold%
        ELSEIF MODE_4 THEN [OUT9D] JUMP INHIBS;
                %Inhibit%
        ELSE [OUT9A] JUMP CHKMODE;

CLEANUP:[RESET] JUMP RES;

END$
```

STIMER.ABL-10/13/89

```
MODULE SHORT_TIMER
TITLE 'SHORT_TIMER        VERSION 21-AUG-89 1430 hrs'
STIMER DEVICE 'P22V10';

CLK     PIN     1;
FRBH    PIN     3;
FRRB    PIN     4;
FDRB    PIN     5;
STEN    PIN     8;
TIMER   PIN     9;
FRWT    PIN     10;
FCLERR  PIN     13;

T0      PIN     14;
T1      PIN     15;
T2      PIN     16;
T3      PIN     17;
T4      PIN     18;
!STO    PIN     20;
R0      PIN     21;
R1      PIN     22;
!FRESET         PIN     23;

COUNT   =       [T4..T0];
RCOUNT  =       [R1..R0];
RINIT   =       [1,0];
CINIT   =       [1,1,1,1,1];

EQUATIONS

WHEN (FCLERR == 1) # (FRBH == 1) # (FRRB == 0) # (FDRB == 1) THEN
RCOUNT:=RINIT;
                                                ELSE RCOUNT:=RCOUNT-1;

WHEN (TIMER == 0) # (FCLERR == 1) # (STEN == 0) # (FRWT == 1) THEN
COUNT:=CINIT;
                                                ELSE COUNT:=COUNT-1;

STO     :=      ((COUNT == 0) # (STO == 1)) & (FCLERR == 0);

FRESET  :=      ((RCOUNT == 0) # (FRESET ==1)) & (FCLERR == 0);

END SHORT_TIMER
```

STROBE.ABL-9/28/89

```
module ds_generator
title 'Fastbus Readout Controller, data strobe generator
Gustavo Cancelo, Fermilab, 4/3/89'

"Added DS to LDS and UDS 19-SEP-89 RK
"Changed LDS, and UDS from pins 16 and 17 to pins 15 and 16.  27-SEP-89 RK

        strobe   device    'P22V10';

AD0,AD1,SIZ0,SIZ1,RW,DS       pin     8,7,6,5,4,3;
LDS,UDS                       pin     15,16;
WLDS,WUDS                     pin     22,23;
WUUDS,WUMDS,WLMDS,WLLDS       pin     21,20,19,18;
RDS                           pin     14;

L,H,X,Z = 0,1,.X.,.Z.;

TRANF_TYPE = [SIZ1,SIZ0];
LONG = ^b00;
TRI_BY = ^b11;
WORD = ^b10;
BYTE = ^b01;

AD1_0 = [AD1,AD0];

equations
         !UDS = !AD0 * !DS;
         !LDS =  AD0 * !DS # !SIZ0 * !DS # SIZ1 * !DS;

         !WUDS = !AD0 & !RW;
         !WLDS =  (AD0 # !SIZ0 # SIZ1) & !RW;

         !RDS = RW;


         !WUUDS = !AD0 & !AD1 & !RW;

         !WUMDS = (!AD1 & !SIZ0
                #  AD0 & !AD1
                # !AD1 & SIZ1) & !RW;

         !WLMDS = (!AD0 & AD1
               # !AD1 & !SIZ0 & !SIZ1
               # !AD1 & SIZ0 & SIZ1
               #  AD0 & !AD1 & !SIZ0) & !RW;

         !WLLDS = (AD0 & SIZ0 & SIZ1
               # !SIZ0 & !SIZ1
               #  AD0 & AD1
               #  AD1 & SIZ1) & !RW;

test_vectors

([TRANF_TYPE,AD1,AD0, RW] -> [WUUDS,WUMDS,WLMDS,WLLDS])
 [   LONG,      0, 0,  L ] -> [  L,     L,     L,     L  ];
 [   TRI_BY,    0, 0,  L ] -> [  L,     L,     L,     H  ];
```

```
  [   WORD,      0, 0,  L ] -> [  L,    L,    H,    H  ];
  [   BYTE,      0, 0,  L ] -> [  L,    H,    H,    H  ];
  [   LONG,      0, 1,  L ] -> [  H,    L,    L,    L  ];
  [   TRI_BY,    0, 1,  L ] -> [  H,    L,    L,    L  ];
  [   WORD,      0, 1,  L ] -> [  H,    L,    L,    H  ];
  [   BYTE,      0, 1,  L ] -> [  H,    L,    H,    H  ];

  [   LONG,      1, 1,  H ] -> [  H,    H,    H,    H  ];
  [   TRI_BY,    1, 1,  H ] -> [  H,    H,    H,    H  ];
  [   WORD,      1, 1,  H ] -> [  H,    H,    H,    H  ];
  [   BYTE,      1, 1,  H ] -> [  H,    H,    H,    H  ];
  [   LONG,      1, 0,  H ] -> [  H,    H,    H,    H  ];
  [   TRI_BY,    1, 0,  H ] -> [  H,    H,    H,    H  ];
  [   WORD,      1, 0,  H ] -> [  H,    H,    H,    H  ];
  [   BYTE,      1, 0,  H ] -> [  H,    H,    H,    H  ];


test_vectors
([TRANF_TYPE, DS,AD1,AD0, RW] -> [WUDS,WLDS,UDS,LDS])
  [  LONG,      0, 0,  0,  L ] -> [ L,   L,   L,  L];
  [  WORD,      0, 0,  1,  H ] -> [ H,   H,   H,  L];
  [  BYTE,      0, 1,  0,  H ] -> [ H,   H,   L,  H];
  [  BYTE,      0, 1,  1,  H ] -> [ H,   H,   H,  L];
  [  BYTE,      0, 0,  0,  H ] -> [ H,   H,   L,  H];
  [  BYTE,      0, 0,  1,  H ] -> [ H,   H,   H,  L];


END_
```

Appendix D - FSCC Documentation

Fermilab Drawing Numbers

FNAL #	Title	Description
0882-MB-199070	Mounting Bracket	FASTBUS Module Front Panel Mount
0880.000-ED-215714	FSCC/VDAS Interface	E771 FSCC/VDAS interface schematic
0880.000-ED-269065	FSCC/VDAS Interface	E791 FSCC/VDAS interface schematic
0880.000-AC-269129	FSCC Layer 1	Trace Layer 1
0880.000-AC-269130	FSCC Layer 2	-2.0V & -5.2V Layer 2
0880.000-AC-269131	FSCC Layer 3	Ground Layer 3
0880.000-AC-269132	FSCC Layer 4	+5.0V Layer 4
0880.000-AC-269133	FSCC Layer 5	Trace Layer 5
0880.000-AC-269134	FSCC Top Silk	Top Silkscreen Photo
0880.000-AC-269135	FSCC Bot Silk	Bottom Silkscreen Photo
0880.000-AC-269136	FSCC Solder Mask	Top & Bot. Solder Mask
0880.000-AC-269137	FSCC Front Panel	Front Panel Silkscreen Photo
0880.000-MD-269138	FSCC Assembly	Assembly drawing
0880.000-MD-269139	FSCC Front Panel	Front Panel Mechanical Drawing
0880.000-MD-269140	FSCC Mechanical	Board Dimensions & Pads
0880.000-MD-269141	FSCC OPORT Test Board	OPORT Test Board Schematic
0880.000-MD-269142	FSCC FPP Test Board	Front Panel Port Test Board Schematic
	FSCC FPP Top Layer	Front Panel Port Test Board Top Layer
	FSCC FPP Bottom Layer	Front Panel Port Test Board Bottom Layer
0880.000-MD-269143	FSCC Schematic	11 Page Schematic and Block Diagram

Fermilab Documents

FASTBUS Smart Crate Controller - PC3, Design Specification
Fermilab Computing Division
Mark Bernett - Online and Data Acquisition Software Groups
Mark Bowden, Rick Kwarciany, John Urish - Data Acquisition Electronics Group
Fermilab Physics Department
Gustavo Cancelo

Diagnostics for the FASTBUS Smart Crate Controller - PN417
Fermilab Computing Division
Mark Bernett, Dave Slimmer - Online and Data Acquisition Software Groups
Fermilab Computing Division
Rick Kwarciany, John Urish - Data Acquisition Electronics Group

Release Notes for SCG68K V2.3
PN 376
David M. Berg, Bryan MacKinnon - Fermilab Computing Division
Online Systems Software Group

SCG68K User's Guide and Reference
PN369
Peter Heinicke, David Berg, Bryan MacKinnon, Tom Nicinski, Gene Oleynik -
Fermilab Computing Division, Online Systems and Data Acquisition Software Groups

Serial Port Driver for the PAN-DA pSOS Environment
PN379.2

Bryan MacKinnon - Fermilab Computing Division, Data Acquisition Software Group

FSCC Parts List, Front Panel Port Test Board Parts List, OPORT Parts List and FSCC/VDAS interface Parts List

The parts lists are implemented in a Microsoft EXCEL spreadsheet program to allow easy updating and sorting. They are available from John Urish at Fermilab as a printed copy or on a Macintosh floppy disk.

Non-Fermilab Documents

pSOS-68K, pROBE-68K, pRISM-68K

Software Components Group, Inc.
4655 Old Ironsides Drive
Santa Clara, CA 95054

IEEE Standard FASTBUS
IEEE 960

Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, New York - 10017

68020 32-Bit Microprocessor Manual

Motorola Literature Distribution
P.O. Box 20912
Phoenix, AZ 85036

FASTBUS Standard Routines
DOE/ER 0325

National Technical Information Service, U.S. Dept. of Commerce
Springfield, Virginia 22161

FSCC Software

FSCC System Software

Software Components Group, Inc.
4655 Old Ironsides Drive
Santa Clara, CA 95054
(Refer to "SCG68K User's Guide and Reference" or contact David Berg at Fermilab)

FSCC Diagnostic Software

The diagnostic software is supplied by Fermilab and described in "Diagnostics for the FASTBUS Smart Crate Controller". The source code or compiled code may be obtained by contacting Dave Slimmer at Fermilab.

FSCC Microcode

PAL/PLD Equations for FSCC

The PAL/PLD sources are listed in Appendix C of this document. The source listings and JEDEC files may be obtained on a Macintosh or IBM format floppy disk from John Urish at Fermilab.

FPORT Equations for FSCC

The EPS448 sources are listed in Appendix C of this document. The source listings and JEDEC files may be obtained on a Macintosh or IBM format floppy disk from John Urish at Fermilab.

PAL/PLD Equations for OPORT Test Board

The source listings and JEDEC files may be obtained on a Macintosh or IBM format floppy disk from John Urish at Fermilab.

Appendix E - FSCC Front Panel

FASTBUS Slave: Yellow LED

FASTBUS Master: Green LED

Front Panel Output (20 pin IDC):

Pin#	RS422 Funciton
1	Front Panel 0 +
2	Front Panel 0 -
3	Front Panel 1 +
4	Front Panel 1 -
5	Front Panel 2 +
6	Front Panel 2 -
7	Front Panel 3 +
8	Front Panel 3 -
9-20	Reserved

Trigger Input (20 pin IDC):

Pin#	RS422 Funciton
1-10	Reserved
11	Trigger Strobe +
12	Trigger Strobe -
13	Trigger ID0 +
14	Trigger ID0 -
15	Trigger ID1 +
16	Trigger ID1 -
17	Trigger ID2 +
18	Trigger ID2 -
19	Trigger ID3 +
20	Trigger ID3 -

Reset Pushbutton: Hard processor reset.

Remote Reset:: Hard processor reset. This is an active low TTL input. Shorting the connetor or applying a TTL low will cause a reset. This input may be Daisy-Chained.

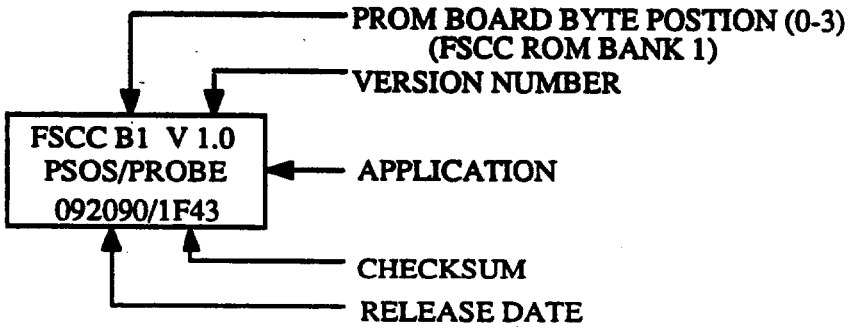
Perm_In/Perm_Out: Serial "daisy-chain" signals for FSCC auxiliary port bussing, LEMO connectors.

2 Serial Ports: RS232 signal levels (4 pin LEMO connectors) one for Host connection and one for Terminal connection.
Terminal port
Host port (Null Modem)

Ethernet Port: Cheapernet signal levels, Isolated BNC Connector.

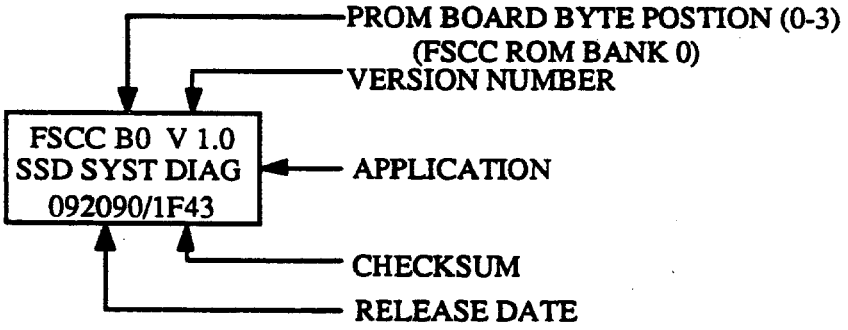
Appendix F - FSCC EPROMs

ROM BANK 1: OPERATING SYSTEM AND FSCC DIAGNOSTICS

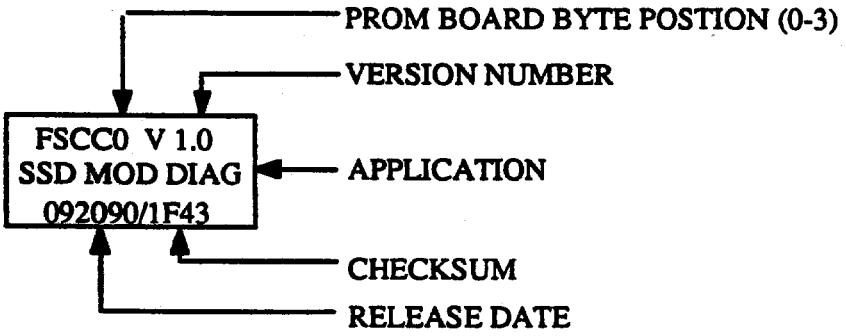


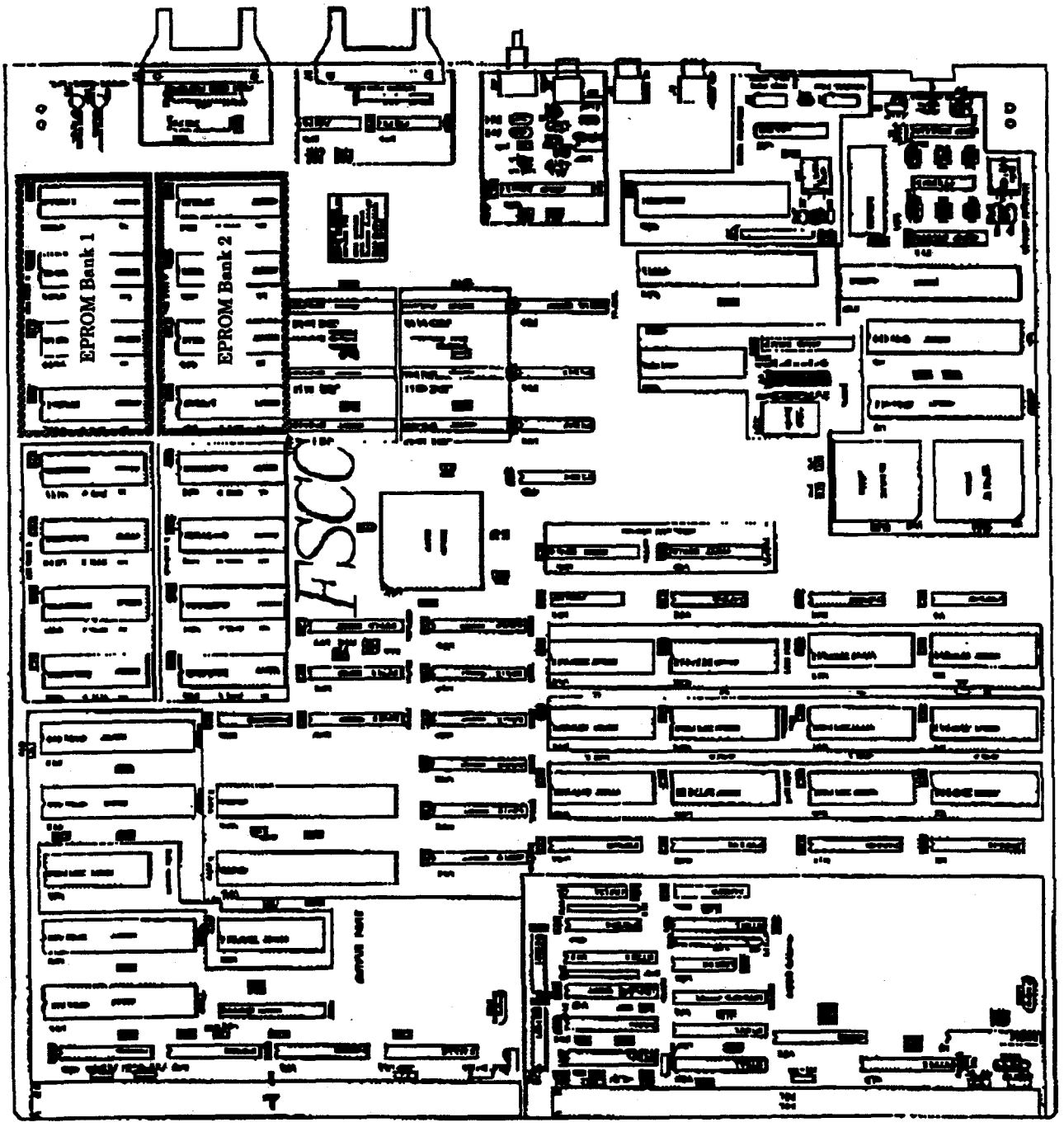
ROM BANK 2: SYSTEM OR INDIVIDUAL MODULE DIAGNOSTICS

OPTION 1; System Diagnostic Tests



OPTION 2; Individual Module Tests





FASTBUS SMART CRATE CONTROLLER

VERSION PC3

EPROM Assembly Drawing