



```

BCK                pin 69      istype 'reg' ;
DATA               pin 70      istype 'reg' ;
n384FSOUT0, n384FSOUT1  pin 67,36 ;
DASYSClk          pin 58 ;
INSEL             pin 28      istype 'reg' ;

!CS1              pin 26 ;
!CS2              pin 27 ;

IYAN, NRESET      pin 32,29 ;

// Connect with 1016 //

SIN               pin 40 ;
SOUT              pin 110 ;
!S0               pin 30 ;
S1               pin 31      istype 'reg' ;
AD1               pin 73 ;
G1R              pin 55 ;
FMVR, FMVW       pin 37,38 ;

// node //

CLKRATE, OUT_IN_  node istype 'reg' ;
DLRCKEXT, DTACKOUT node istype 'reg' ;
QA1, QA2, QA5     node istype 'reg' ;
QA6, QA7, QA8     node istype 'reg' ;
QA9, QA10, QA11, QA12 node istype 'reg' ;
PREQ, PEXREQ, EXREQ node istype 'reg' ;
DEXREQ           node istype 'reg' ;
AST              node istype 'reg' ;
DDTA, SGD, DSGD   node istype 'reg' ;
LG, RG, STEREO_MONO node istype 'reg' ;
CSR1, CSR2, CSR3, CSR4 node istype 'reg' ;
CSR5, CSR6, CSR7, CSR8 node istype 'reg' ;
CSR9, CSR10, CSR11, CSR12 node istype 'reg' ;
M256             node istype 'reg' ;
ACKIG, TERR       node istype 'reg' ;
DHSYNC, PHSYNC    node istype 'reg' ;

S384FS           node istype 'com' ;
QALOAD, QAEXCLR, EXEN node istype 'com' ;
QA1EN, QA2EN, BCKEN, QA5EN node istype 'com' ;
QA6EN, QA7EN, QA8EN, QA9EN node istype 'com' ;
QA10EN, QA11EN   node istype 'com' ;
QA12EN           node istype 'com' ;
DATAEXT128FS     node istype 'com' ;
ASTS, ASTE       node istype 'com' ;
DPLS, G1         node istype 'com' ;
G2, G2W, G2UW, G2R, G3R node istype 'com' ;
ERR, n48K, n44K, n32K node istype 'com' ;
DOE              node istype 'com' ;
MCLKSELA, MCLKSELB node istype 'com' ;
CSREN            node istype 'com' ;
LRATEEN          node istype 'com' ;
REQ256S, MD256   node istype 'com' ;

plsi property 'PULLUP OFF' ;

plsi property 'PULLUP EXPCLOUT' ;
plsi property 'PULLUP EXREQOUT' ;
plsi property 'PULLUP IACK2' ;

```

```

plsi property 'PULLUP IACK4' ;
plsi property 'PULLUP HSYNC' ;
plsi property 'PULLUP VSYNC' ;
plsi property 'PULLUP IYAN' ;

```

## Equations

```

//      Clocking      //

// EX Enable //
EXEN      = !OUT_IN_ & !ERROR ;

LRATEEN   = !CLKRATE & OUT_IN_ ;

S384FS     = EXEN
            & n384FSEXT
            # CLKSELA & CLKSELB & !EXEN
            & CLK12M
            # !CLKSELA & !EXEN
            & CLK16M
            # CLKSELA & !CLKSELB & !EXEN
            & CLK18M ;

n384FSOUT0 = S384FS ; // 1016
n384FSOUT1 = S384FS ;

DASYSClk   = S384FS & !LRATEEN # !QA1 & LRATEEN ;

DLRCKEXT    := LRCKEXT ;
DLRCKEXT.clk = n384FS ;

DHSYNC      := HSYNC # VSYNC ;
DHSYNC.clk  = n384FS ;

PHSYNC      := DHSYNC ;
PHSYNC.clk  = n384FS ;

// Clear pulse //
QAEXCLR     = LRCKEXT & !DLRCKEXT & EXEN #
              ( DHSYNC & !PHSYNC ) & MD256 ;

QALOAD      = QA1 & BCK &
              !( QA11 & QA10 & QA9 & QA8 &
                  QA7 & QA6 & QA5 & MD256 ) #
              QAEXCLR ;

// Synchronous counter //
// Load '100' when EXEN and LRATEEN is 'Low'. //
// Load '001' when EXEN is 'High' and LRATEEN is 'High'. //

QA1EN       = 1 ;
QA1          := ( QA1 & !QA1EN # !QA1 & QA1EN )
              & !QALOAD
              # QALOAD & ( !LRATEEN $ QAEXCLR ) ;
QA1.clk      = n384FS ;

QA2EN       = QA1EN & QA1 ;
QA2          := ( QA2 & !QA2EN # !QA2 & QA2EN )
              & !QALOAD
              # QALOAD & ( !LRATEEN & !QAEXCLR ) ;
QA2.clk      = n384FS ;

//      128fs      //
BCKEN       = QA2EN & QA2 ;

```

```

BCK                := ( BCK & !BCKEN # !BCK & BCKEN )
                   & !QALOAD
                   # QALOAD & ( !LRATEEN & QAEXCLR ) ;
BCK.clk            = n384FS ;

//      64fs      //
QA5EN              = QALOAD ;
QA5                := ( QA5 & !QA5EN # !QA5 & QA5EN )
                   & !QAEXCLR ;
QA5.clk            = n384FS ;

//      32fs      //
QA6EN              = QA5EN & QA5 ;
QA6                := ( QA6 & !QA6EN # !QA6 & QA6EN )
                   & !QAEXCLR ;
QA6.clk            = n384FS ;

QA7EN              = QA6EN & QA6 ;
QA7                := ( QA7 & !QA7EN # !QA7 & QA7EN )
                   & !QAEXCLR ;
QA7.clk            = n384FS ;

QA8EN              = QA7EN & QA7 ;
QA8                := ( QA8 & !QA8EN # !QA8 & QA8EN )
                   & !QAEXCLR ;
QA8.clk            = n384FS ;

QA9EN              = QA8EN & QA8 ;
QA9                := ( QA9 & !QA9EN # !QA9 & QA9EN )
                   & !QAEXCLR ;
QA9.clk            = n384FS ;

QA10EN             = QA9EN & QA9 ;
QA10               := ( QA10 & !QA10EN # !QA10 & QA10EN )
                   & !QAEXCLR ;
QA10.clk           = n384FS ;

QA11EN             = QA10EN & QA10 ;
QA11               := ( ( QA11 & !QA11EN # !QA11 & QA11EN )
                   & !QAEXCLR ) ;
QA11.clk           = n384FS ;

QA12EN             = QA10EN & QA11 & !QA10;
QA12               := ( QA12 & !QA12EN # !QA12 & QA12EN ) ;
QA12.clk           = n384FS ;

//      Converting 32fs to 128fs (DATAEXT)      //

//      Shift Registers                          //

CSREN              = QA6EN & !QA6 ;

CSR1               := DATAEXT & CSREN # CSR1 & !CSREN ;
CSR1.clk           = n384FS ;

CSR2               := CSR1 & CSREN # CSR2 & !CSREN ;
CSR2.clk           = n384FS ;

CSR3               := CSR2 & CSREN # CSR3 & !CSREN ;
CSR3.clk           = n384FS ;

CSR4               := CSR3 & CSREN # CSR4 & !CSREN ;
CSR4.clk           = n384FS ;

```

```

CSR5      := CSR4 & CSREN # CSR5 & !CSREN ;
CSR5.clk  = n384FS ;

CSR6      := CSR5 & CSREN # CSR6 & !CSREN ;
CSR6.clk  = n384FS ;

CSR7      := CSR6 & CSREN # CSR7 & !CSREN ;
CSR7.clk  = n384FS ;

CSR8      := CSR7 & CSREN # CSR8 & !CSREN ;
CSR8.clk  = n384FS ;

CSR9      := CSR8 & CSREN # CSR9 & !CSREN ;
CSR9.clk  = n384FS ;

CSR10     := CSR9 & CSREN # CSR10 & !CSREN ;
CSR10.clk = n384FS ;

CSR11     := CSR10 & CSREN # CSR11 & !CSREN ;
CSR11.clk = n384FS ;

CSR12     := CSR11 & CSREN # CSR12 & !CSREN ;
CSR12.clk = n384FS ;

"
DATAEXT128FS = CSR12 & ( QA8 & QA7 & QA6 & QA5 ) #
                CSR11 & ( !QA8 & !QA7 & !QA6 & !QA5 ) #
                CSR10 & ( !QA8 & !QA7 & ( QA6 $ QA5 ) ) #
                CSR9 & ( !QA8 & !QA7 & QA6 & QA5 ) #
                CSR8 & ( !QA8 & QA7 & !QA6 & !QA5 ) #
                CSR7 & ( !QA8 & QA7 & ( QA6 $ QA5 ) ) #
                CSR6 & ( !QA8 & QA7 & QA6 & QA5 ) #
                CSR5 & ( QA8 & !QA7 & !QA6 & !QA5 ) #
                CSR4 & ( QA8 & !QA7 & ( QA6 $ QA5 ) ) #
                CSR3 & ( QA8 & !QA7 & QA6 & QA5 ) #
                CSR2 & ( QA8 & QA7 & !QA6 & !QA5 ) #
                CSR1 & ( QA8 & QA7 & ( QA6 $ QA5 ) ) ;

```

```

// Sequence //

LRCK      := ( !( QA11 & QA11EN ) & LRCK #
              ( QA11EN & ( !QA11 # MD256 ) ) ) ;
LRCK.clk  = n384FS ;

EXPCLOUT  = !QA11 ;
EXPCLOUT.oe = !( M256 & ACKIG ) ;

REQ256S   = !QA10 & !QA9 &
            !QA8 & !QA7 & QA6 & !QA5 & QALOAD ;

PREQ      := ( ( ASTE & !MD256 # REQ256S & MD256 ) &
              ( STEREO_MONO_ # ( !QA11 $ !OUT_IN_ ) ) &
              ( OUT_IN_ # CLKRATE # QA12 ) ) #
              PREQ & !ASTS ;
PREQ.clk  = n384FS ;

PEXREQ    := PREQ ;
PEXREQ.clk = CLK20M ;

DEXREQ    := PEXREQ ;
DEXREQ.clk = CLK20M ;

EXREQ     := ( PEXREQ & !DEXREQ ) #
              EXREQ & !( EXACK & !ACKIG ) &

```

```

EXREQ.clk      = CLK20M ;
EXREQOUT       = EXREQ ;
EXREQOUT.oe    = !( M256 & ACK1G ) ;

SIN            = !OUT_IN_ & DATAEXT128FS #
               OUT_IN_ & SOUT ;

DATA           := ( !QALOAD & DATA # QALOAD &
               ( SIN & !( !LG & LRCK ) & !( !RG & !LRCK ) &
                 !S1 ) ) ;
DATA.clk       = n384FS ;

// Start at $2E //
ASTS           = QA10 & !QA9 &
               QA8 & QA7 & QA6 & !QA5 & QALOAD ;
// Ended at $3E //
ASTE          = QA10 & QA9 &
               QA8 & QA7 & QA6 & !QA5 & QALOAD ;

AST            := ASTS # AST & !ASTE ;
AST.clk        = n384FS ;

SGD            := G1 & OUT_IN_ & !R_W_ ; // & DPLS ;
SGD.clk        = n384FS ;

DSGD           := SGD ;
DSGD.clk       = n384FS ;

// Enable of clock //
S0             = !S1 & QALOAD #
               SGD & !DSGD ;

S1             := !AST ;
S1.clk         = n384FS ;

TERR           := ( !D12 & G2UW # TERR & !G2UW ) #
               !S1 & ( SGD & !DSGD # G1R ) ;
TERR.clk       = n384FS ;

// Address decoding //

// $0ecc0** //
AD1            = A23 & A22 & A21 & !A20 &
               A19 & A18 & !A17 & !A16 &
               A15 & A14 & !A13 & !A12 &
               !A11 & !A10 & !A9 & !A8 &
               A7 & AS & ( LDS # UDS ) ;

DTACKOUT_PIN   = DTACKOUT & ( AD1 # ( IACK2 # IACK4 ) & AS ) ;

DTACKOUT       := AD1 # ( IACK2 # IACK4 ) & AS ;
DTACKOUT.clk   = CLK10M_ ;

DDTA           := DTACKOUT ;
DDTA.clk       = CLK10M_ ;

DPLS           = DTACKOUT & !DDTA ;

G1             = !A6 & !A5 & !A4 & AD1 ;
G1R            = G1 & R_W_ ;

```

```

G2          = !A6 & !A5 & A4 & AD1 ;
G2W         = G2 & !R_W_ & LDS & DPLS ;
G2UW        = G2 & !R_W_ & UDS & DPLS ;
G2R         = G2 & R_W_ ;

G3R         = !A6 & A5 & !A4 & AD1 & R_W_ ;

// Registers //

NRESET      = EXRESET & IYAN ;

OUT_IN_     := ( D0 & G2W # OUT_IN_ & !G2W )
              & !NRESET ;
OUT_IN_.clk = CLK20M ;

STEREO_MONO_ := ( D1 & G2W # STEREO_MONO_ & !G2W ) ;
STEREO_MONO_.clk = CLK20M ;

LG          := ( D2 & G2W # LG & !G2W ) #
              NRESET ;
LG.clk      = CLK20M ;

RG          := ( D3 & G2W # RG & !G2W ) #
              NRESET ;
RG.clk      = CLK20M ;

CLKSELA     := ( ( D4 & G2W # CLKSELA & !G2W )
              & !EXEN
              # FS2 & EXEN ) ;
CLKSELA.clk = CLK20M ;

CLKSELB     := ( ( !D5 & G2W # CLKSELB & !G2W )
              & !EXEN #
              FS1 & EXEN ) ;
CLKSELB.clk = CLK20M ;

MCLKSELA    = CLKSELA & !( ERROR & !OUT_IN_ ) ;
MCLKSELB    = !CLKSELB & !( ERROR & !OUT_IN_ ) ;

INSEL       := ( D6 & G2W # INSEL & !G2W ) & !EXRESET ;
INSEL.clk   = CLK20M ;

CLKRATE     := ( D7 & G2W # CLKRATE & !G2W ) ;
CLKRATE.clk = CLK20M ;

ACKIG       := ( !D8 & G2UW # ACKIG & !G2UW ) #
              EXRESET ;
ACKIG.clk   = CLK20M ;

M256        := ( !D9 & G2UW # M256 & !G2UW ) #
              EXRESET ;
M256.clk    = CLK20M ;
MD256       = M256 & !ACKIG ;

//

ERR         = ( FS1 & !FS2 ) # ERROR ;
n48K        = !FS1 & FS2 & !ERROR ;
n44K        = !FS1 & !FS2 & !ERROR ;
n32K        = FS1 & FS2 & !ERROR ;

```

```

//      Output to X680x0 data bus      //
DOE          = G2R # G3R ;
D0           = G2R & OUT_IN_ #
              G3R & ERR ;
D0.oe        = DOE ;
D1           = G2R & STEREO_MONO_ #
              G3R & !n48K ;
D1.oe        = DOE ;
D2           = G2R & LG #
              G3R & !n44K ;
D2.oe        = DOE ;
D3           = G2R & RG #
              G3R & !n32K ;
D3.oe        = DOE ;
D4           = ( G2R # G3R ) & MCLKSELA ;
D4.oe        = DOE ;
D5           = ( G2R # G3R ) & MCLKSELB ;
D5.oe        = DOE ;
D6           = ( G2R # G3R ) & INSEL ;
D6.oe        = DOE ;
D7           = ( G2R # G3R ) & CLKRATE ;
D7.oe        = DOE ;
D8           = ( G2R # G3R ) & !ACKIG ;
D8.oe        = DOE ;
D9           = ( G2R # G3R ) & !M256 ;
D9.oe        = DOE ;
D10          = ( G2R # G3R ) & !EXREQ ;
D10.oe       = DOE ;
D11          = ( G2R # G3R ) &
              ( QA11 # !( OUT_IN_ # CLKRATE # QA12 ) ) ;
D11.oe       = DOE ;
D12          = ( G2R # G3R ) & !TERR ;
D12.oe       = DOE ;

//                                     //
//      OPNA                         //
//                                     //

CS1          = AD1 & A6 & !A5 & !A4 & !A3 ;
CS2          = AD1 & A6 & !A5 & !A4 & A3 ;
FMVW         = AD1 & !A6 & A5 & A4 &
              DPLS & !R_W_ ;
FMVR         = AD1 & !A6 & A5 & A4 &
              R_W_ # !ACK2 # !ACK4 ;

END

```

