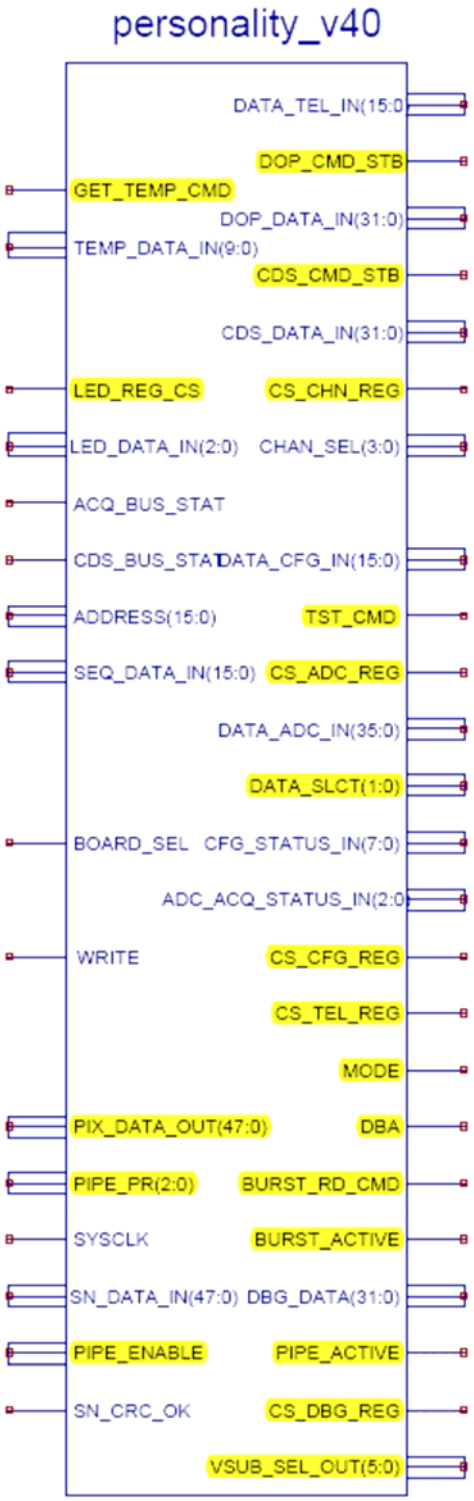
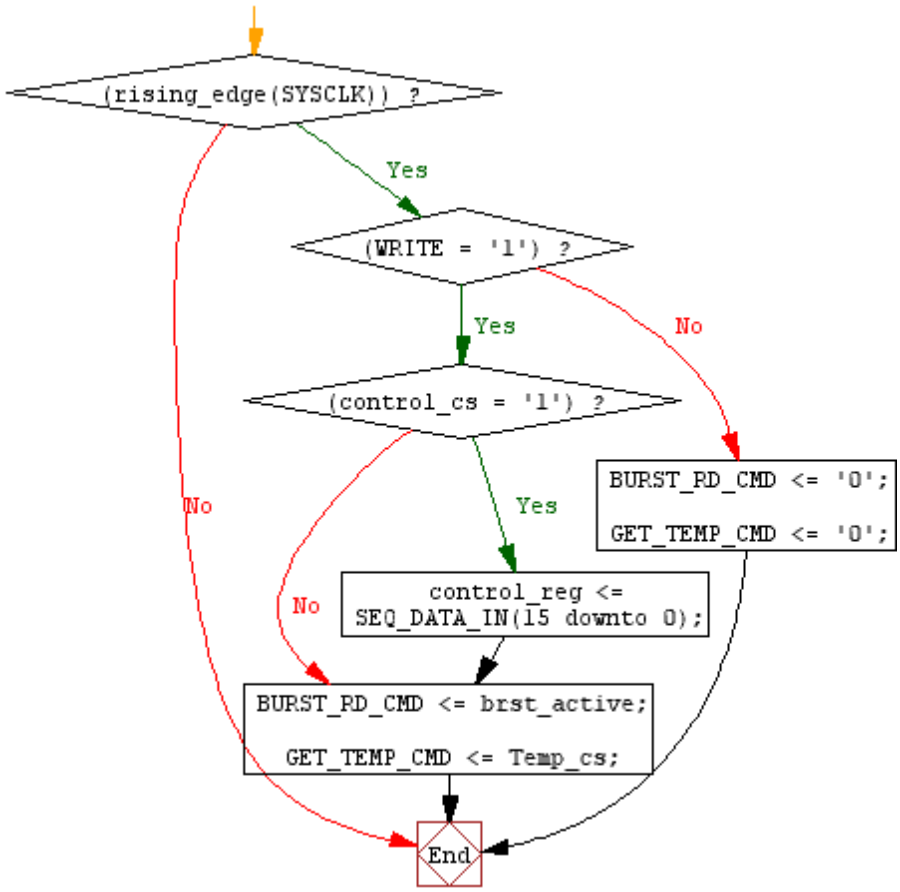
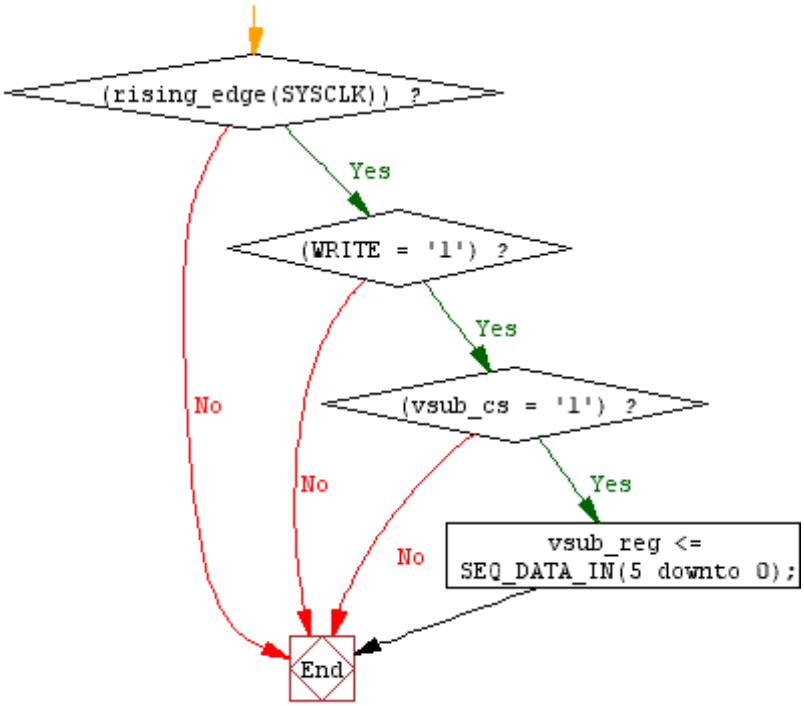
```
-----------------------------------------------------------------------
-- IMPLEMENTATION NOTES :
--
-- FIXED ADDRESS ASSIGNMENTS
--
-- VSUBSTRATE ON/OFF               = 0xFFF8 (6 BITS WRITE - 6 BITS READ) **** dlh added for Vsub control
-- LED INDICATOR SELECT PORT       = 0xFFF9 (3 BITS WRITE - 3 BITS READ)
-- CIRCUIT BOARD SERIAL NUMBER     = 0xFFFA (NO WRITE - 32 BITS READ)
-- CIRCUIT BOARD TEMPERATURE       = 0xFFFB (NO WRITE - 8 BITS READ)
-- CONTROL REGISTER ADDRESS        = 0xFFFC (16 BIT WRITE - 16 BIT READ)
-- STATUS REGISTER ADDRESS         = 0XFFFD (NO WRITE - 16 BIT READ)
-- BOARD ID REGISTER ADDRESS       = 0xFFFE (NO WRITE - 16 BIT READ)
-- CODE ID REGISTER ADDRESS        = 0xFFFF (NO WRITE - 16 BIT READ)
--
-- CCD ACQUISITION BOARD SPECIFIC ADDRESS MAP
--
-- CDS CONTROL DATA WRITE CMD      = 0x0000 (32 BITS WRITE - 32 BITS READ)
-- DOP CONTROL DATA WRITE CMD      = 0x0001 (32 BITS WRITE - 32 BITS READ)
-- SET CHANNEL IDENTITY COMMAND    = 0x0003 (WRITE NO SIGNIFICANT DATA - NOT READABLE)
-- BURST MODE COMMAND              = 0x0004 (WRITE NO SIGNIFICANT DATA - NOT READABLE)
-- ADC DATA MEMORY ADDRESS RANGE   = 0x0100 => 0x010F (NO WRITE - 16 BIT READ)
-- UNUSED                          = 0x0120 => 0x01FF
-- ADC CHANNEL CONFIG DATA         = 0x0200 => 0x020F (16 BIT WRITE - 16 BIT READ)  **** dlh there is enough
room here for more ADC channels
-- VIDEO CHANNNEL OFFSET DAC DATA  = 0x0210 => 0x021F (16 BIT WRITE - 16 BIT READ)  **** dlh also here for
the OFFSET DAC
-- HV BIAS CHAN DAC DATA           = 0x0220 => 0x023F (16 BIT WRITE - 16 BIT READ)  **** dlh no extra room
here! we will have to expand
-- HV BIAS CHAN DAC DATA           = 0x0220 => 0x024F (16 BIT WRITE - 16 BIT READ)  NEW  ASSIGNMENT
-- HV BIAS TELEMETRY CONFIG DATA   = 0x0240 => 0x025F (16 BIT WRITE - 16 BIT READ)  **** dlh no extra room
here! we will have to expand
-- HV BIAS TELEMETRY CONFIG DATA   = 0x0250 => 0x027F (16 BIT WRITE - 16 BIT READ)  NEW  ASSIGNMENT
-- HV BIAS CHAN OUTPUT ENABLES     = 0x0260 (4 BIT WRITE - 4 BIT READ)              **** dlh might be easier
to more this up to 0x0280
-- HV BIAS CHAN OUTPUT ENABLES     = 0x0280 (4 BIT WRITE - 4 BIT READ)              NEW  ASSIGNMENT
-- UNUSED                          = 0x0261 => 0x02FF
-- ADC CHANNEL ORDERING TABLE      = 0x0300 => 0x030F (4 BIT WRITE - 4 BITS READ)
-- ADC PIPELINE CHANNEL COUNT      = 0x033F (4 BITS WRITE - 4 BITS READ)
-- UNUSED                          = 0x0340 => 0x043F
-- HV BIAS CHAN TELEMETRY DATA     = 0x0440 => 0x045F (WRITE TRIGGERS READOUT - 16 BIT READ)
-- DEBUG REGISTERS                 = 0x0480 => 0X0483
--
-- CONTROL REGISTER BITS
-- BIT  0  DAUGHTER BOARD PRESENT       BIT  8  LSB PIPELINE PRIORITY
-- BIT  1  ENABLE PIPELINE READ         BIT  9      PIPELINE PRIORITY
-- BITS 2  ENABLE BURST MODE READ       BIT 10  MSB PIPELINE PRIORITY
-- BITS 3  SPARE BIT                    BIT 11  SPARE BIT
-- BIT  4  LSB DATA WIDTH SELECT        BIT 12  SELECT FAST ACQUISITION MODE
-- BIT  5  MSB DATA WIDTH SELECT        BIT 13  SPARE BIT
-- BITS 6  SPARE BIT                    BIT 14  SPARE BIT
-- BITS 7  SPARE BIT                    BIT 15  SPARE BIT
-----------------------------------------------------------------------
```




```
ACQ_BUS_STAT : in  std_logic;    -- **** dlh alphabetized
ADC_ACQ_STATUS_IN: in std_logic_vector(2 downto 0);
ADDRESS      : in  std_logic_vector(15 downto 0);
BOARD_SEL    : in  std_logic;
CDS_BUS_STAT : in  std_logic;
CDS_DATA_IN  : in  std_logic_vector(31 downto 0);
CFG_STATUS_IN: in  std_logic_vector(7 downto 0);
CHAN_SEL     : in  std_logic_vector(3 downto 0);
DATA_ADC_IN  : in  std_logic_vector(35 downto 0);
DATA_CFG_IN  : in  std_logic_vector(15 downto 0);
DATA_TEL_IN  : in  std_logic_vector(15 downto 0);
DBG_DATA     : in  std_logic_vector(31 downto 0);
DOP_DATA_IN  : in  std_logic_vector(31 downto 0);
LED_DATA_IN  : in  std_logic_vector(2 downto 0);
SEQ_DATA_IN  : in  std_logic_vector(15 downto 0);
SN_CRC_OK    : in  std_logic; SN_DATA_IN   : in
std_logic_vector(47 downto 0);  -- digital serial number
TEMP_DATA_IN : in  std_logic_vector(9 downto 0);
WRITE        : in  std_logic;
SYSCLK       : in  std_logic
BURST_ACTIVE : out std_logic;
BURST_RD_CMD : out std_logic;
CDS_CMD_STB  : out std_logic;
CS_ADC_REG   : out std_logic;
CS_CFG_REG   : out std_logic;
CS_CHN_REG   : out std_logic;
CS_DBG_REG   : out std_logic;
CS_TEL_REG   : out std_logic;
DATA_SLCT    : out std_logic_vector(1 downto 0);
DBA          : out std_logic;
DOP_CMD_STB  : out std_logic;
GET_TEMP_CMD : out std_logic;
LED_REG_CS   : out std_logic;
MODE         : out std_logic;
PIPE_ACTIVE  : out std_logic;
PIPE_ENABLE  : out std_logic;
PIPE_PR      : out std_logic_vector(2 downto 0);
PIX_DATA_OUT : out std_logic_vector(47 downto 0);
TST_CMD      : out std_logic;
VSUB_SEL_OUT : out std_logic; -- **** dlh
```

```
signal code_ident, board_ident : std_logic_vector(15 downto 0);
signal control_reg : std_logic_vector(15 downto 0);
signal status_reg : std_logic_vector(15 downto 0);
signal vsub_reg : std_logic;    -- **** dlh added
signal control_cs, status_cs, board_cs, code_cs : std_logic; -- std chip selects for board regs
signal vsub_cs : std_logic;                      -- **** dlh
signal cs_adc, cs_cfg : std_logic;        -- decoded chip selects for adc data and cfg data
signal cs_tel, cs_dbg : std_logic;     -- decoded chip select for tel adc data and dbg registers
signal cs_chan, cs_chan_cnt : std_logic;  -- decoded chip select for Channel Select Ram
signal cds_active, dop_active : std_logic;
signal brst_active, tst_active : std_logic;
signal LedReg_cs, SNumber_cs, Temp_cs : std_logic;   Cntrl_Reg : Process (SYSCLK)


code_cs <= '1' when ((ADDRESS(15 DOWNTO 0) = x"FFFF") AND (BOARD_SEL = '1')) ELSE '0';
board_cs <= '1' when ((ADDRESS(15 DOWNTO 0) = X"FFFE") and (BOARD_SEL = '1')) else '0';
status_cs <= '1' when ((ADDRESS(15 downto 0) = X"FFFD") and (BOARD_SEL = '1')) else '0';
control_cs <= '1' when ((ADDRESS(15 downto 0) = X"FFFC") and (BOARD_SEL = '1')) else '0';
Temp_cs <= '1' when ((ADDRESS(15 downto 0) = X"FFFB") and (BOARD_SEL = '1')) else '0';
SNumber_cs <= '1' when ((ADDRESS(15 downto 0) = X"FFFA") and (BOARD_SEL = '1')) else '0';
LedReg_cs <= '1' when ((ADDRESS(15 downto 0) = X"FFF9") and (BOARD_SEL = '1')) else '0';
LED_REG_CS <= LedReg_cs;
vsub_cs <=  '1' when ((ADDRESS(15 downto 0) = X"FFF8") and (BOARD_SEL = '1')) else '0';
VSUB_SEL_OUT <= vsub_reg;
cds_active <= '1' when ((ADDRESS(15 downto 0) = X"0000") and (BOARD_SEL = '1')) else '0';
CDS_CMD_STB  <= cds_active;
dop_active <= '1' when ((ADDRESS(15 downto 0) = X"0001") and (BOARD_SEL = '1')) else '0';
DOP_CMD_STB <= dop_active;
tst_active <= '1' when ((ADDRESS(15 downto 0) = X"0003") and (BOARD_SEL = '1')) else '0';
TST_CMD <= tst_active and WRITE;
brst_active <= '1' when ((ADDRESS(15 downto 0) = X"0004") and (BOARD_SEL = '1')) else '0';
cs_adc <= '1' when ((ADDRESS(15 downto 4) = X"010") and (BOARD_SEL = '1')) else '0';
CS_ADC_REG <= cs_adc;
cs_cfg <= '1' when ((ADDRESS(15 downto 8) = X"02") and (BOARD_SEL = '1')) else '0';
CS_CFG_REG <= cs_cfg;
cs_chan <= '1' when ((ADDRESS(15 downto 4) = X"030") and (BOARD_SEL = '1')) else '0';
cs_chan_cnt <= '1' when ((ADDRESS(15 downto 0) = X"0310") and (BOARD_SEL = '1')) else '0';
CS_CHN_REG <= cs_chan or cs_chan_cnt;
cs_tel <= '1' when ((ADDRESS(15 downto 0) >= X"0440") and (ADDRESS(15 downto 0) <= X"047F") and (BOARD_SEL =
'1')) else '0';
CS_TEL_REG <= cs_tel;
cs_dbg <= '1' when ((ADDRESS(15 downto 0) >= x"0480") and (ADDRESS (15 downto 0) <= x"0483") and (BOARD_SEL =
'1')) else '0';
CS_DBG_REG <= cs_dbg and WRITE;
DBA           <= control_reg(0);
PIPE_ENABLE   <= control_reg(1) or control_reg(2);
PIPE_ACTIVE   <= control_reg(1);
BURST_ACTIVE  <= control_reg(2);
DATA_SLCT     <= control_reg(5 downto 4);
PIPE_PR       <= control_reg(10 downto 8);
MODE          <= control_reg(12);
VSUB_SEL_OUT <= vsub_reg;
status_reg <= '0' & ADC_ACQ_STATUS_IN(2 downto 0) & ACQ_BUS_STAT & CDS_BUS_STAT & '0' & SN_CRC_OK &
CFG_STATUS_IN(7 downto 0);
PIX_DATA_OUT <= ("000000" & DATA_ADC_IN(35 downto 18) & "000000" & DATA_ADC_IN(17 downto 0)) when (BOARD_SEL
= '0') else (others => 'Z');
PIX_DATA_OUT <= (x"00000000" & control_reg(15 downto 0))  when (control_cs = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"00000000" & status_reg(15 downto 0)) when (status_cs = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"00000000" & board_ident(15 downto 0))  when (board_cs = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"00000000" & code_ident(15 downto 0))   when (code_cs = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"00000000000" & CHAN_SEL(3 downto 0))   when (cs_chan = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"00000000" & DATA_CFG_IN(15 downto 0))  when (cs_cfg = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"00000000" & DATA_TEL_IN(15 downto 0))  when (cs_tel = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"0000000" & "00" & DATA_ADC_IN(17 downto 0))  when (cs_adc = '1') else  (others => 'Z');
PIX_DATA_OUT <= (x"00000000000" & "0" & LED_DATA_IN) when (LedReg_cs = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"0000" & CDS_DATA_IN(31 downto 0)) when (cds_active = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"0000" & DOP_DATA_IN(31 downto 0)) when (dop_active = '1') else (others => 'Z');
PIX_DATA_OUT <= SN_DATA_IN when (SNumber_cs = '1') else (others => 'Z');
PIX_DATA_OUT <= (x"000000000" & "00" & TEMP_DATA_IN) when (Temp_cs = '1') else  (others => 'Z');
PIX_DATA_OUT <= (x"0000" & DBG_DATA(31 downto 0)) when (cs_dbg = '1') else (others => 'Z');
IDENT : Identity_v4f port map (board_ident_out => board_ident, code_ident_out => code_ident);
```



Vsubtrate_Reg : Process (SYSCLK)