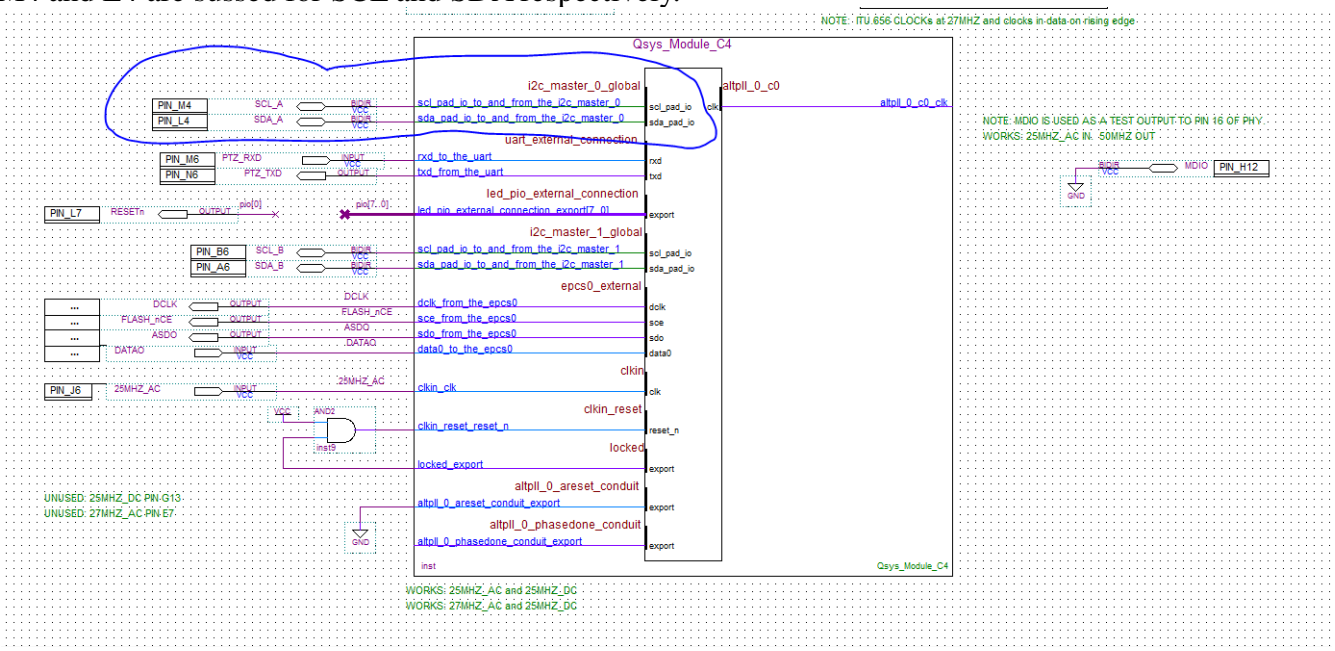
NIOS II OpenCores I2C Block

This shows the SCL and SDA pins in the QSYS block from the Block Editor Point of View.
Note M4 and L4 are sussed for SCL and SDA respectively.



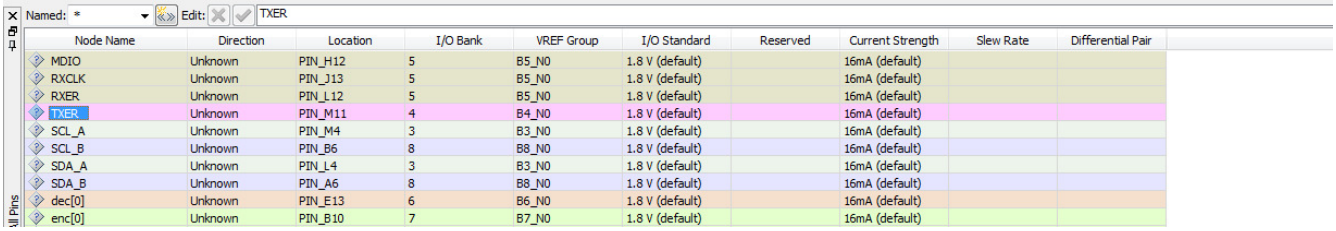

This shows the I/O declarations in the .v file for the Qsys Module above. Note the inout type for both SCL and SDA in verilog.

```
// Qsys_Module_C4.v

// Generated using ACDS version 12.1sp1 243 at 2013.04.04.10:53:43

`timescale 1 ps / 1 ps
module Qsys_Module_C4 (
      input  wire        clkin_clk,                                //                      clkin.clk
      output wire        altpll_0_c0_clk,                          //                 altpll_0_c0.clk
      inout  wire        scl_pad_io_to_and_from_the_i2c_master_0,  //        i2c_master_0_global.scl_pad_io
      inout  wire        sda_pad_io_to_and_from_the_i2c_master_0,  //                            .sda_pad_io
      input  wire        rxd_to_the_uart,                          //    uart_external_connection.rxd
      output wire        txd_from_the_uart,                        //                            .txd
      input  wire        clkin_reset_reset_n,                      //                 clkin_reset.reset_n
      input  wire        altpll_0_areset_conduit_export,           //    altpll_0_areset_conduit.export
      output wire [7:0]  led_pio_external_connection_export,       // led_pio_external_connection.export
      output wire        altpll_0_phasedone_conduit_export,        //  altpll_0_phasedone_conduit.export
      inout  wire        scl_pad_io_to_and_from_the_i2c_master_1,  //        i2c_master_1_global.scl_pad_io
      inout  wire        sda_pad_io_to_and_from_the_i2c_master_1,  //                            .sda_pad_io
      output wire        locked_export,                            //                      locked.export
      output wire        dclk_from_the_epcs0,                      //              epcs0_external.dclk
      output wire        sce_from_the_epcs0,                       //                            .sce
      output wire        sdo_from_the_epcs0,                       //                            .sdo
      input  wire        data0_to_the_epcs0                        //                            .data0
  );
```

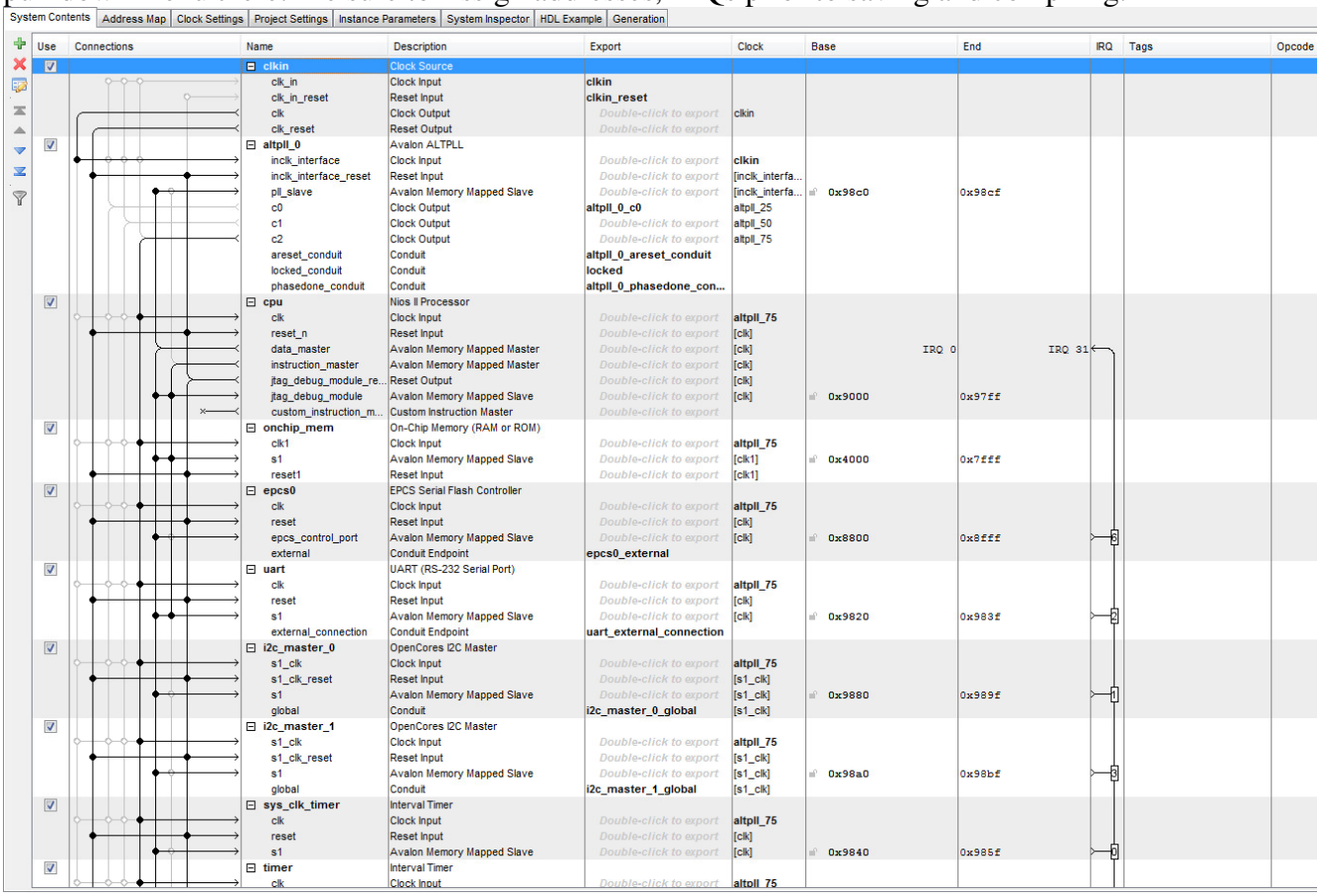
The Pins are shown as unknown in the pin editor: Earlier versions of Quartus required the direction to be set prior to compiling or errors would be generated. I used 7.x for the first project long ago.



Named: * Edit: TXER

| Node Name | Direction | Location | I/O Bank | VREF Group | I/O Standard | Reserved | Current Strength | Slew Rate | Differential Pair |
|---|---|---|---|---|---|---|---|---|---|
| MDIO | Unknown | PIN_H12 | 5 | B5_N0 | 1.8 V (default) | | 16mA (default) | | |
| RXCLK | Unknown | PIN_J13 | 5 | B5_N0 | 1.8 V (default) | | 16mA (default) | | |
| RXER | Unknown | PIN_L12 | 5 | B5_N0 | 1.8 V (default) | | 16mA (default) | | |
| TXER | Unknown | PIN_M11 | 4 | B4_N0 | 1.8 V (default) | | 16mA (default) | | |
| SCL_A | Unknown | PIN_M4 | 3 | B3_N0 | 1.8 V (default) | | 16mA (default) | | |
| SCL_B | Unknown | PIN_B6 | 8 | B8_N0 | 1.8 V (default) | | 16mA (default) | | |
| SDA_A | Unknown | PIN_L4 | 3 | B3_N0 | 1.8 V (default) | | 16mA (default) | | |
| SDA_B | Unknown | PIN_A6 | 8 | B8_N0 | 1.8 V (default) | | 16mA (default) | | |
| dec[0] | Unknown | PIN_E13 | 6 | B6_N0 | 1.8 V (default) | | 16mA (default) | | |
| enc[0] | Unknown | PIN_B10 | 7 | B7_N0 | 1.8 V (default) | | 16mA (default) | | |

This is the Qsys module showing the connections to the OpenCores I2C devices. There are two in this project: One for the video encoder/decoder and one SFP module. The other for the second SFP module. Usually there is a slower clock for all the low speed I/O devices but this one uses 75 MHZ everywhere. The input is 25MHZ from the PHY clock. The connections are assigned on the LEFT by clicking the desired sources. The selected clock is shown in the Clock column so you can re-select in a pull down menu there. Be sure to Assign addresses, IRQs prior to saving and compiling.



MAIN “C” Source  Please close your eyes as you read this. I am not a programmer so it is not pretty.

```
// This program runs on internal memory in the 'Nios II/e

#include <stdio.h>
#include <string.h>
#include <system.h>
#include <unistd.h>                          // usleep()
#include "i2c.h"
#include "links.h"
#include "altera_avalon_pio_regs.h"


int main()
{
       unsigned char x;
       int v1;
       int old_v1;
       unsigned char ch;
       unsigned int stat;
       printf("\n\r\nNIOS2/e Running: Internal SRAM...");
       printf("\n\r\n\rINIT_UART");
       init_uart ();    // for STDIO vector to PTZ Port at 9600 baud.

       printf("\n\r\n\rINIT_I2C");
       init_i2c(I2C_MASTER_0_BASE);
       init_i2c(I2C_MASTER_1_BASE);

    printf("\n\r RESETn low for 35mS and high for 130mS: ");
//     IOWR_ALTERA_AVALON_PIO_DIRECTION(LED_PIO_BASE, 0xff);  //All Output
       IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, 0xfe);  //Force RESETn low
       usleep(25000);  // RESETn low time TVP5150 data sheet
       IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, 0xff);  //Force RESETn HIGH
       usleep(100000);  // wait for PIO outputs to stabilize high

       init_MCP23017_PIO ();
       init_TVP5151_video_decoder ();               //TVP5151
       init_ADV7393BCP_video_encoder ();      //ADV7393BCP
//     init_audio_codec_digital_passthru ();        // TLV320AIC3254
       init_audio_codec_analog_passthru ();     //TLV320AIC3254

//      ADV7393BCP_color_bars ();

//      Poll Initial SFP status
        v1=i2c_read(I2C_MASTER_0_BASE, 0x40>>1, 0x12);  //GPIOA (0x12) $41= Read Port A
        v1=v1 + i2c_read(I2C_MASTER_0_BASE, 0x40>>1, 0x13);  //GPIOA (0x12) $41= Read Port
B
        old_v1=v1;
        poll_sfp_module_io ();

        printf("\n\r\n\rPoll SFP Status and Echo PTZ Characters Forever...\n\r");

         while (1)  // Event loop never exits
         {
//     Code to select live video in if the horizontal PLL is locked on the TVP5151 decoder
          usleep(65000);  // RESETn low time TVP5150 data sheet
          x = i2c_read(I2C_MASTER_0_BASE, 0xba>>1, 0x88);     // poll SR1 for Horizontal PLL
Locked
```

```
            if ((x & 0x0e) == 0xe)
               {
                           IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, 0xfd);  //Force MUX
SELECT low  (0xfd= Video)
//                         printf("\b\b\b\bLOCK ");
            }
            else
            {
                           IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, 0xff);  //Force MUX
SELECT high (0xff=ColorBars)
//                         printf("\b\b\b\bLOST ");
            }
         }


   while (1);  // NIOS never gets to here! This is the last infinite loop if one does not
exit above this one due to a typo or missing end-bracket.
   {
         printf("\n\r\n\rEndLess");
   }

   return 0;
}
```

I2C ROUTINES:  Don't open your eyes yet.  There is more below.  I removed audio codec src.

```
#include "system.h"
#include "i2c.h"
#include "altera_avalon_pio_regs.h"
#include <stdio.h>
#include <string.h>
#include "altera_avalon_uart_regs.h"
#include <unistd.h>

void init_i2c(long port) {
// Setup I2C pre-scaler for 100KHz with osc_clk of 27 MHz
//  int prescale = 16000000 /(6*100000);

  int prescale = 32000000 /(6*100000);
  IOWR_I2C_PRERLO(port, prescale & 0xff);
  IOWR_I2C_PRERHI(port, (prescale & 0xff00)>>8);
  IOWR_I2C_CTR(port, 0x80);   // Enable core
}

void i2c_write(long port, unsigned char address, unsigned char reg, unsigned char data) {
  i2c_wait_tip(port);

// write address
  IOWR_I2C_TXR(port, address<<1);
  IOWR_I2C_CR(port, I2C_CR_STA | I2C_CR_WR | I2C_CR_ACK);
  i2c_wait_tip(port);

// write register address
  IOWR_I2C_TXR(port, reg);
  IOWR_I2C_CR(port, I2C_CR_WR | I2C_CR_ACK);
  i2c_wait_tip(port);

// write data
  IOWR_I2C_TXR(port, data);
  IOWR_I2C_CR(port, I2C_CR_WR | I2C_CR_STO | I2C_CR_IACK);
  i2c_wait_tip(port);
}


unsigned char i2c_read(long port, unsigned char address, unsigned char reg) {
  i2c_wait_tip(port);

// write address
  IOWR_I2C_TXR(port, address<<1);
  IOWR_I2C_CR(port, I2C_CR_STA | I2C_CR_WR);
  i2c_wait_tip(port);

// write register address
  IOWR_I2C_TXR(port, reg);
  IOWR_I2C_CR(port, I2C_CR_WR);
  i2c_wait_tip(port);

// write address for reading
  IOWR_I2C_TXR(port, (address<<1) | 1);
  IOWR_I2C_CR(port, I2C_CR_STA | I2C_CR_WR);
  i2c_wait_tip(port);
```

```
// read data
  IOWR_I2C_CR(port, I2C_CR_RD | I2C_CR_ACK | I2C_CR_STO);
  i2c_wait_tip(port);

  return IORD_I2C_RXR;
}

void i2c_wait_tip(long port) {
  while ((IORD_I2C_SR & I2C_SR_TIP) > 0) {}
}

//  ********************POLL sfp-modules ******************
void poll_sfp_module_io ()
 {
        int (val);
        printf("\n\nPoll SFP Module I/O:");

// READ MCP23017_PIO PORT A
     val=i2c_read(I2C_MASTER_0_BASE, 0x40>>1, 0x12);  //GPIOA $41= Read Port A
       printf("\n  SFP-A: ");
     if ((val & 0x10)  ==0)
        {
       printf("INSTALLED/");
       ((val & 0x40)  ==0) ? printf("FAULT/")     : printf("NO-FAULT/");
       ((val & 0x20)  ==0) ? printf("LOSS-DETECT") : printf("SIGNAL-DETECT");
        }
     else
          {
               printf("NOT INSTALLED");
          }

// READ MCP23017_PIO PORT B
     val=i2c_read(I2C_MASTER_0_BASE, 0x40>>1, 0x13);  //GPIOB $41= Read Port B
       printf("\n  SFP-B: ");
     if ((val & 0x02)  ==0)
        {
       printf("INSTALLED/");
          ((val & 0x04)  ==0) ? printf("FAULT/")    : printf("NO-FAULT/");
            ((val & 0x01)  ==0) ? printf("LOSS-DETECT"): printf("SIGNAL-DETECT");
        }
     else
          {
               printf("NOT INSTALLED");
          }
}

//  ********************INIT PIO MCP23017 ******************
void init_MCP23017_PIO ()
{
         int val;

       printf("\n\r\nInit_MCP23017_PIO Port A");  // MCP23017 is at I2C address $40
(write) & $41 (read)

// Setup MCP23017_PIO REGISTER A
     i2c_write(I2C_MASTER_0_BASE, 0x40>>1, 0x0c, 0xFF ); // GPPU = FF 1=PULLUP ENABLED)
       printf("\n\r  Write $40: $0c: 0xFF = Port A Pullups");
```

```
        i2c_write(I2C_MASTER_0_BASE, 0x40>>1, 0x00, 0xFC ); // IODIRA (0x00) = 7F (0111
1111 1=inputs)
        printf("\n\r  Write $40: $00: 0x7C = IODIRA");

        i2c_write(I2C_MASTER_0_BASE, 0x40>>1, 0x12, 0xff ); //PIOA (0x12)
        printf("\n\r  Write $40: $12: 0xFF PIOA DATA REG"); //MIC & DE HIGH

        printf("\n\r  Dump Port A=");
        val=i2c_read(I2C_MASTER_0_BASE, 0x40>>1, 0x12);  //GPIOA (0x12) $41= Read Port A
        printf("\r  $41:$12=%x MSB Should be 1XXX XX11",val);

// Setup MCP23017_PIO REGISTER B
        printf("\n\r\nInit_PIO Port B");  // MCP23017 is at I2C address $40 (write) & $41
(read)

        i2c_write(I2C_MASTER_0_BASE, 0x40>>1, 0x0d, 0xff ); // GPPU = FF 1=PULLUP ENABLED)
        printf("\n\r  Write $40: $0c: 0xFF = Port B Pullups");

        i2c_write(I2C_MASTER_0_BASE, 0x40>>1, 0x01, 0xFC ); // IODIRB (0x00) = 7F (0111
1111 1=inputs)
        printf("\n\r  Write $40: $00: 0x7C = IODIRB");

        i2c_write(I2C_MASTER_0_BASE, 0x40>>1, 0x13, 0xFF ); //PIOB (0x12)
        printf("\n\r  Write $40: $12: 0xFF PIOB DATA REG"); //MIC & DE HIGH

        printf("\n\r  Dump Port B=");
        val=i2c_read(I2C_MASTER_0_BASE, 0x40>>1, 0x13);  //GPIOB (0x12) $41= Read Port A
        printf("\r  $41:$12=%x MSB Should be 1111 1XXX",val);
     poll_sfp_module_io ();
}

//  ********************INIT VIDEO DECODER ******************
//  Initialize TVP5151(I)

 void init_TVP5151_video_decoder ()
 {
          unsigned char x;
          printf("\n\n\rInit_TVP5151_Video_Decoder");

// Set up Decoder

          {
        printf("\n\r   Read TVP5151 @ BA: Sub 80&81.");
        printf("\n\r   Should be 0x5151.  Read=");
        x = i2c_read(I2C_MASTER_0_BASE, 0xba>>1, 0x80);     // check for $51 at sub address
$80
        printf("%x",x);
        x = i2c_read(I2C_MASTER_0_BASE, 0xba>>1, 0x81);     // check for $51 at sub address
$81
        printf("%x",x);
          }
// At this point you should see $51 & $51 in the NIOSII Console from the read at sub
address $80 & $81
// VERIFY this is true.  The video decoder I2C initialization string below continues.

        i2c_write(I2C_MASTER_0_BASE, 0xba>>1, 0x03, 0x0d);  // Enable YCbCr and Clock as
outputs
        printf("\n\r   Write REG 03: 0d   Enable YCbCr, Sync and Clock as Outputs");
```

```
       i2c_write(I2C_MASTER_0_BASE, 0xba>>1, 0x0D,0x47);  // (Was 07h) Y ranges from 16 to
235. U and V range from 16 to 240
       printf("\n\r   Write REG 0D: 47  Y (16 to 235) & U/V (16 to 240) BT656 with
Embedded Syncs");

       x = i2c_read(I2C_MASTER_0_BASE, 0xba>>1, 0x88);     // poll SR1 1 for chroma
       printf("\n\r   Read SR1 Sub-Address 88=%x ",x);

        x = i2c_read(I2C_MASTER_0_BASE, 0xba>>1, 0x8c);     // poll SR2 video standard
        printf("\n\r   Read SR2 Sub-Address 8c=%x ",x);
        if ((x & 0x80) != 0)
          {
           printf("Auto-Switch Mode ON-> Standard= ");
          }


// Determine video standard
             switch (x & 0x0f){
               case 0x1: printf("(M) NTSC ITU-R BT.601");            break;
               case 0x3: printf("(B, G, H, I, N) PAL ITU-R BT.601"); break;
               case 0x5: printf("(M) PAL ITU-R BT.601");             break;
               case 0x7: printf("PAL-N ITU-R BT.601");               break;
               case 0x9: printf("NTSC 4.43 ITU-R BT.601");           break;
               case 0xb: printf("SECAM ITU-R BT.601");               break;
               default:  printf("Error");                           break;
             }
 }

 //  ********************INIT VIDEO ENCODER *****************
void init_ADV7393BCP_video_encoder ()
{
//     int val;
     printf("\n\r\n\rInit_Video_Encoder");  // ADV7393BCPZ-ND is at I2C address $54

// Setup Video Encoder Registers from Table 65


     i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x17, 0x02 );
     printf("\n\r  Write $54: $17: 0x02");  //Software Reset

     usleep(25000);  //encoder software reset

     i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x00, 0x12 );
       printf("\n\r  Write $54: $00: 0x12");  //DAC1 enabled.  PLL Enabled (16X) Sleep
Mode off.

       i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x01, 0x0 );
       printf("\n\r  Write $54: $01: 0x0");  //0x00 --> SD Input Mode

       i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x80, 0x10 );
       printf("\n\r  Write $54: $80: 0x10");  // 0x10 --> NTSC Standard.  SSAF Luma Filter
enabled.  1.3MHZ Chroma Filter Enabled.

       i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x82, 0xcb );
       printf("\n\r  Write $54: $82: 0xcb");  // Pixel Data Valid. CVBS/Y-C (S-Video Out).
SSAF PrPb Filter enabled.  Active video edge control enabled.  Pedistal enabled.
```

```
       i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x87, 0x00 );   // YUV, Auto, SSAF, Bright,
Hue, Luma, Scale all off (DEFAULT)
       printf("\n\r  Write $54: $87: 0x00");  // Bit7 = 9; for 10-Bit 4:2:2 YCrCb Mode
(Page 47)

       i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x88, 0x00 );   // SD Mode Register 7 Default
Value = 0x00
       printf("\n\r  Write $54: $88: 0x00");  // Pixel Data 8 bits.  Use 0x10 for 10-Bit
4:2:2 YCrCb Mode (Page 47)

// Dump Registers
/*     val=i2c_read(I2C_MASTER_0_BASE, 0x54>>1, 0x17);  //read register
       printf("\n\r  Sub_17h,%x",val);
     val=i2c_read(I2C_MASTER_0_BASE, 0x54>>1, 0x00);  //read register
       printf("\n\r  Sub_00h,%x",val);
     val=i2c_read(I2C_MASTER_0_BASE, 0x54>>1, 0x01);  //read register
       printf("\n\r  Sub_01h,%x",val);
     val=i2c_read(I2C_MASTER_0_BASE, 0x54>>1, 0x80);  //read register
       printf("\n\r  Sub_80h,%x",val);
     val=i2c_read(I2C_MASTER_0_BASE, 0x54>>1, 0x82);  //read register
       printf("\n\r  Sub_82h,%x",val);
*/
}

//  ********************75% COLOR BARS ******************
void ADV7393BCP_color_bars ()
{
      printf("\n\r\n\rColor Bars");  // ADV7393BCZ-ND is at I2C address $54

// Setup Video Encoder Registers
       i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x00, 0x12 );
       printf("\n\r  Write $54: $00: 0x12");  //

//     i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x02, 0x24 );
//     printf("\n\r  Write $54: $02: 0x24");  //Enable grey bars (black screen)

       i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x82, 0xCB);
       printf("\n\r  Write $54: $82: 0xCB");

       i2c_write(I2C_MASTER_0_BASE, 0x54>>1, 0x84, 0x40 );
       printf("\n\r  Write $54: $84: 0x40");
}
```