NIOS-V I/O

Last updated 11/1/26

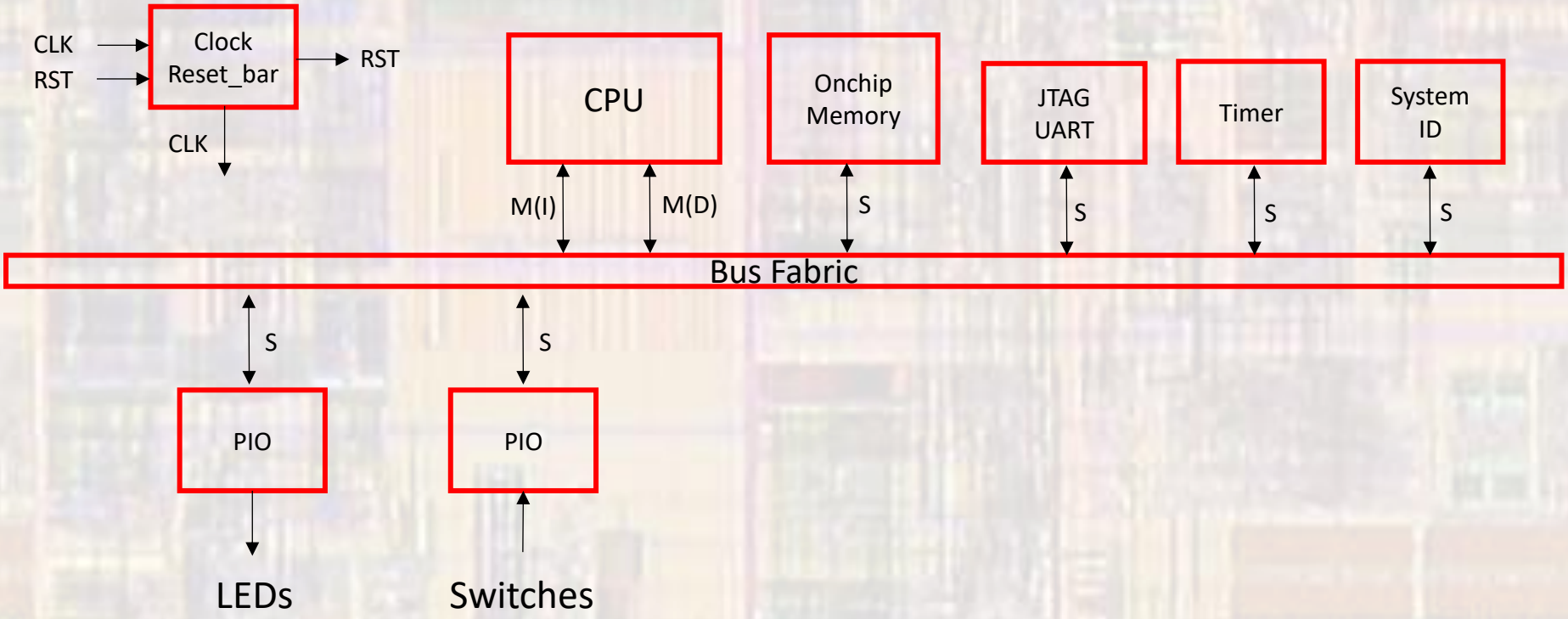
NIOS-V I/O

- NIOS V Embedded Design Suite
  - Configurable Processor
  - Selection of Peripherals
  - Eclipse based Board Support Package (BSP) for SW development (Ashling)

© tj

NIOS-V I/O

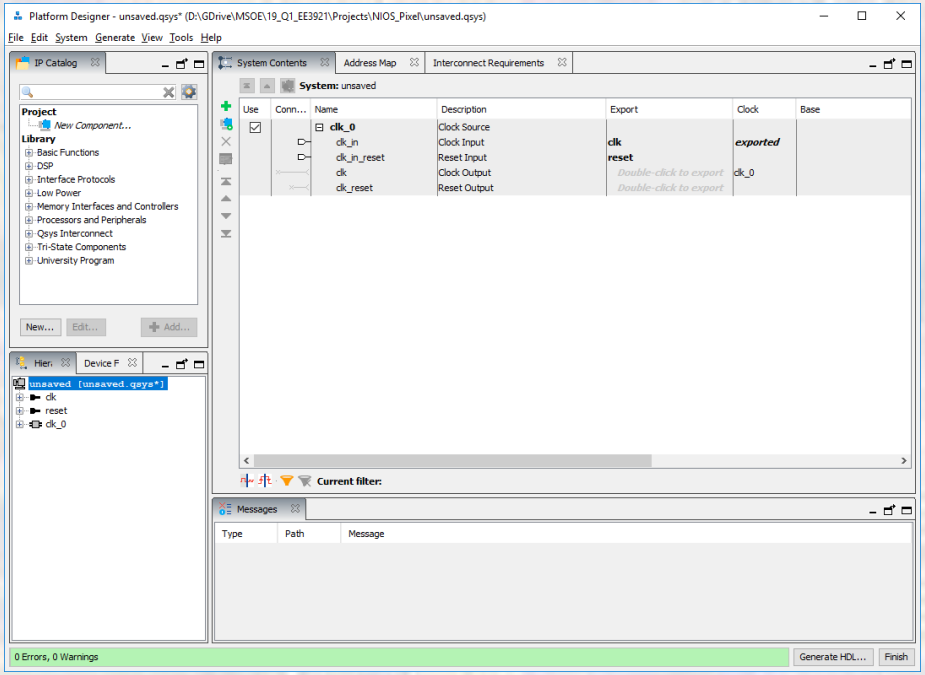
- NIOS V System
  - Create a processor system to display a count on LEDs with switch control for up/dn, pause



© tj

NIOS-V – Hardware Setup

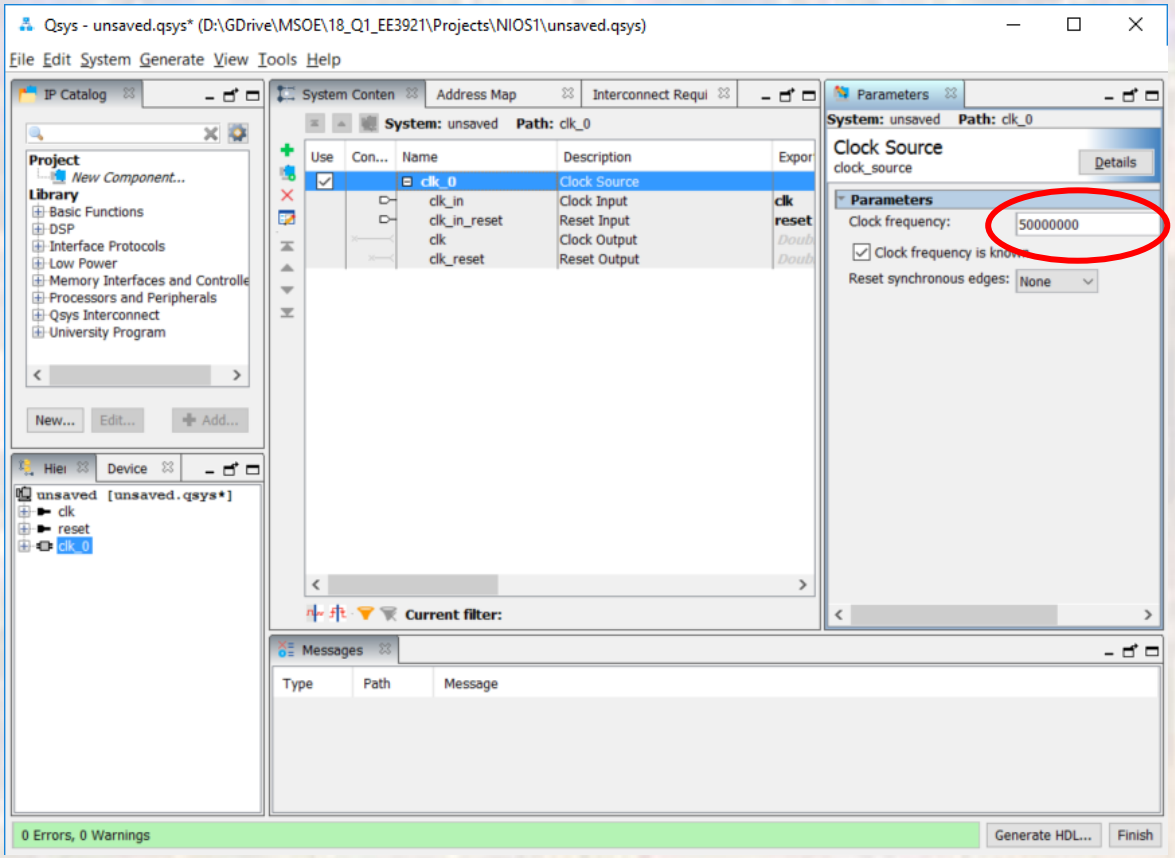
- Create a new Quartus project
  - Do not select a Simulation Tool in EDA Tool Settings
  - Be sure to follow all the steps in the Quartus Project Setup slides, NIOS-V requires all steps to be executed
- Open Tools → Platform Designer



© tj

NIOS-V – Hardware Setup

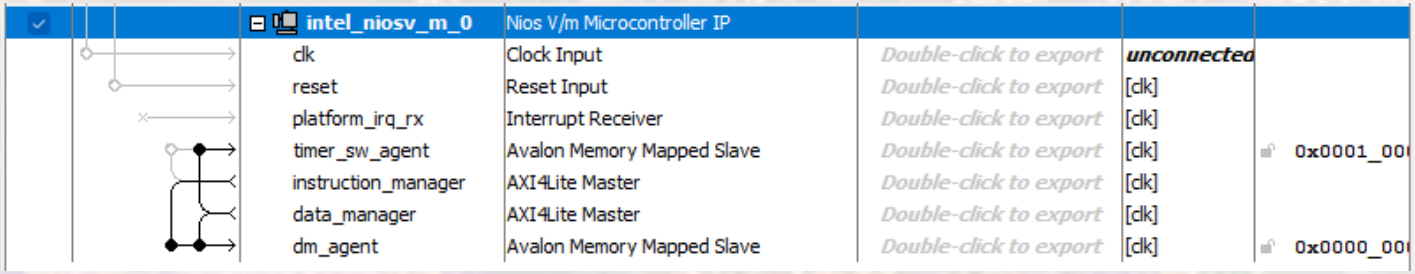
- Create NIOS-V System
  - Double Click on clk_0 - verify clk frequency = 50MHz



© tj

NIOS-V – Hardware Setup

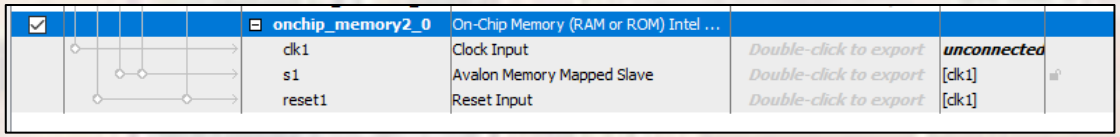
- Add NIOS-V
  - Processors and Peripherals → Embedded Processors → NIOS V/m Microcontroller
  - No other changes for now



- Add On-chip Memory
  - Basic Functions→ On Chip Memory → On Chip Memory (RAM or ROM)…

    RAM

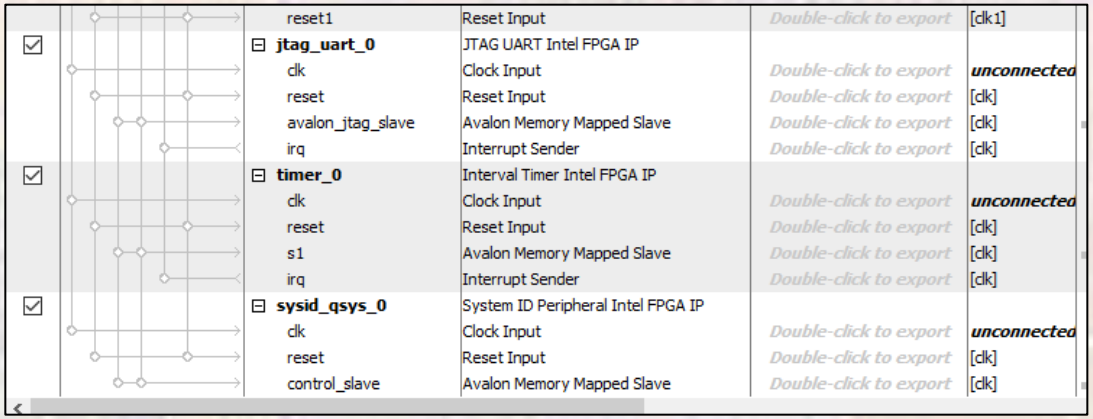
    Size = 100,000 bytes



© tj

NIOS-V – Hardware Setup

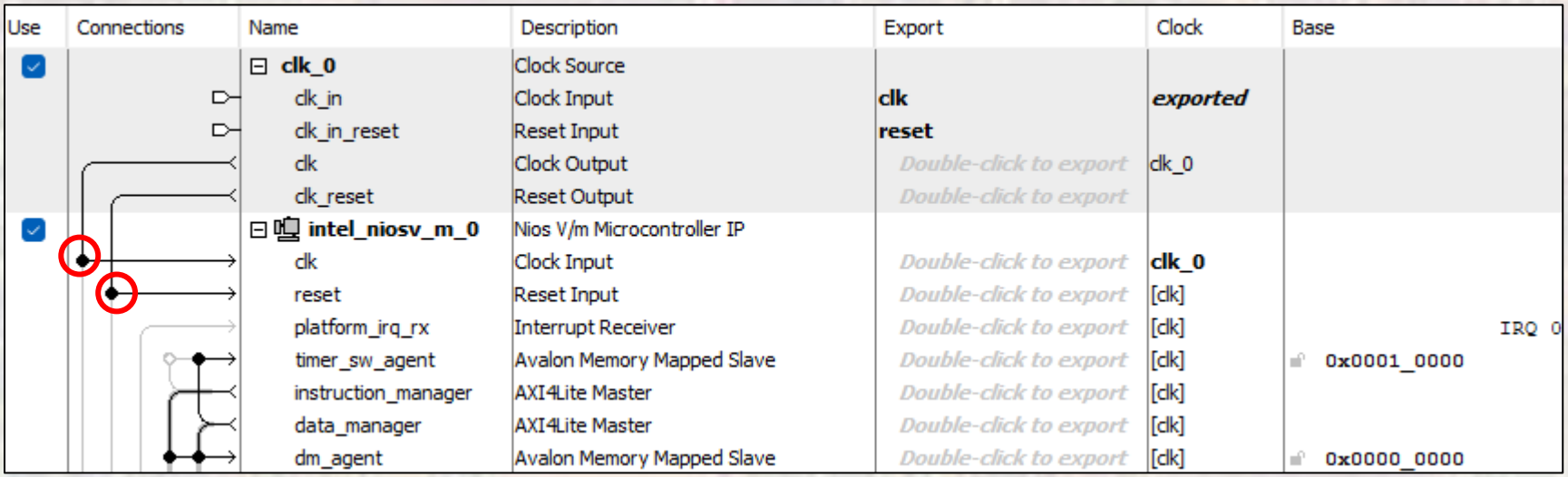
- Add JTAG
  - Interface Protocols → Serial→ JTAG Uart Intel FPGA IP
- Add Timer
  - Processors and Peripherals → Peripherals → Interval Timer Intel FPGA IP
- Add System ID
  - Basic Functions → Simulation; Debug and Verification → Debug and Performance → System ID Peripheral Intel FPGA IP



© tj

NIOS-V – Hardware Setup

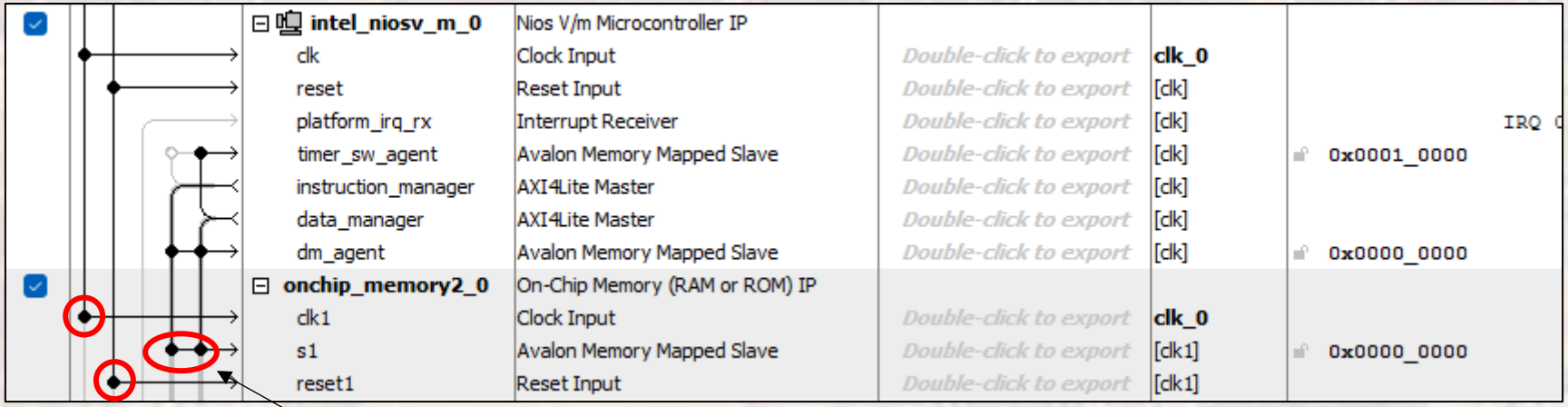
- Connect up basic NIOS system
  - NIOS-V Inputs
    - Clock and Reset



© tj

NIOS-V – Hardware Setup

- Connect up basic NIOS system
  - On-chip Memory
    - clk, reset, instruction manager, data manager

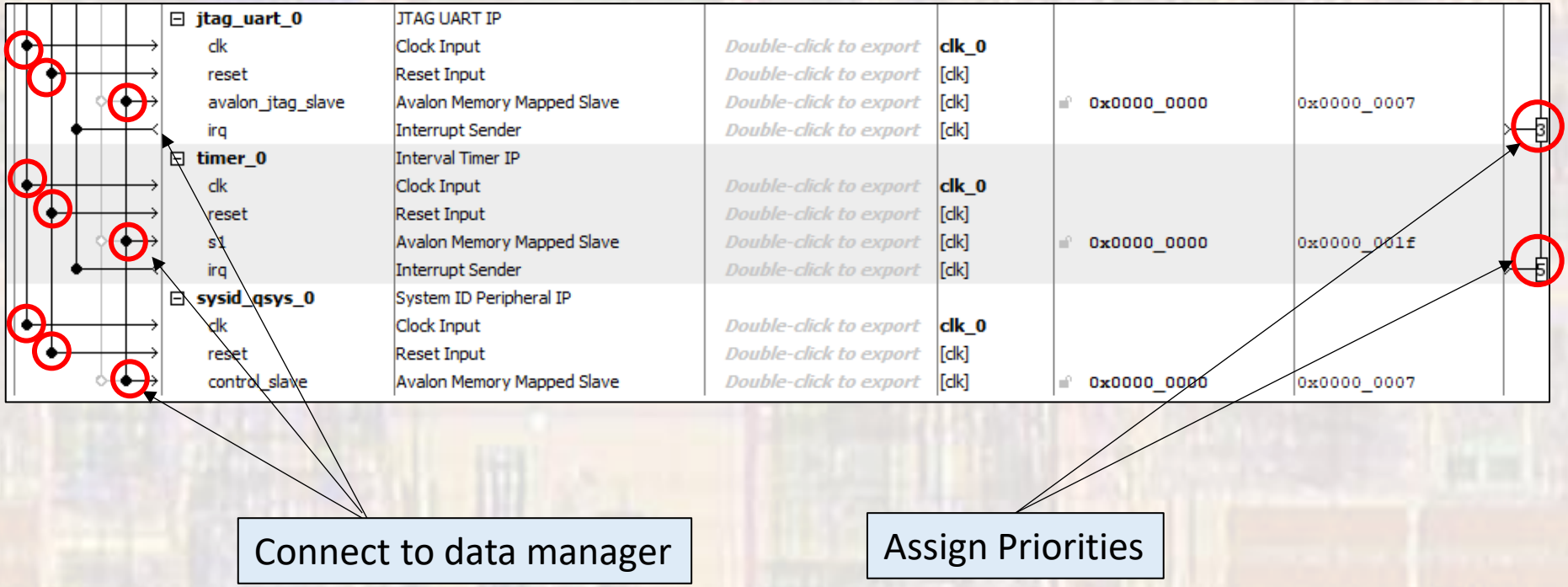


Connect to data and instruction masters

© tj

NIOS-V – Hardware Setup

- Connect up basic NIOS system
  - JTAG, Timer, SysID
  - 16 interrupt priority values 1-16



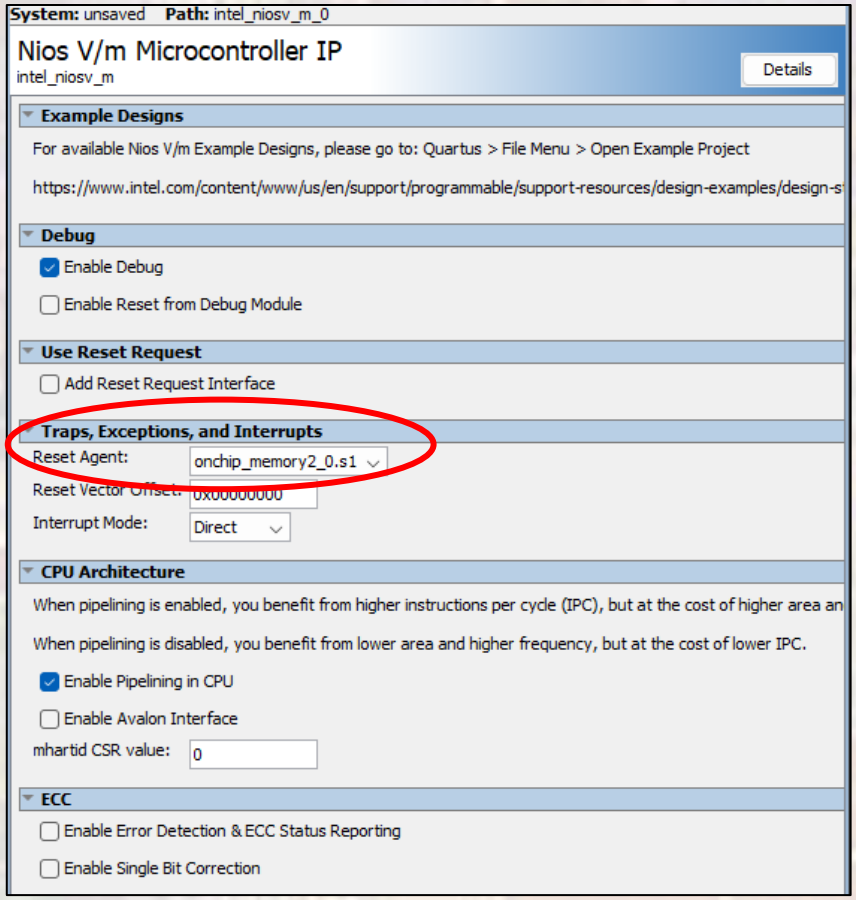
Connect to data manager

Assign Priorities

 © tj

NIOS-V – Hardware Setup

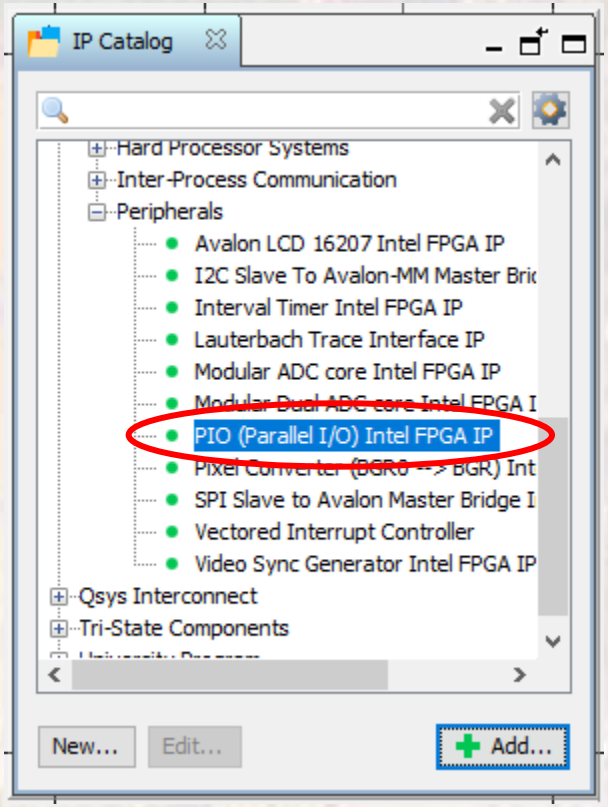
- Connect up basic NIOS system
  - Assign the NIOS-V Reset and Exception vectors
    - Open the NIOS Processor
    - Select Traps, Exceptions, …
    - Select on-chip memory for Reset Agent



 © tj

NIOS-V I/O

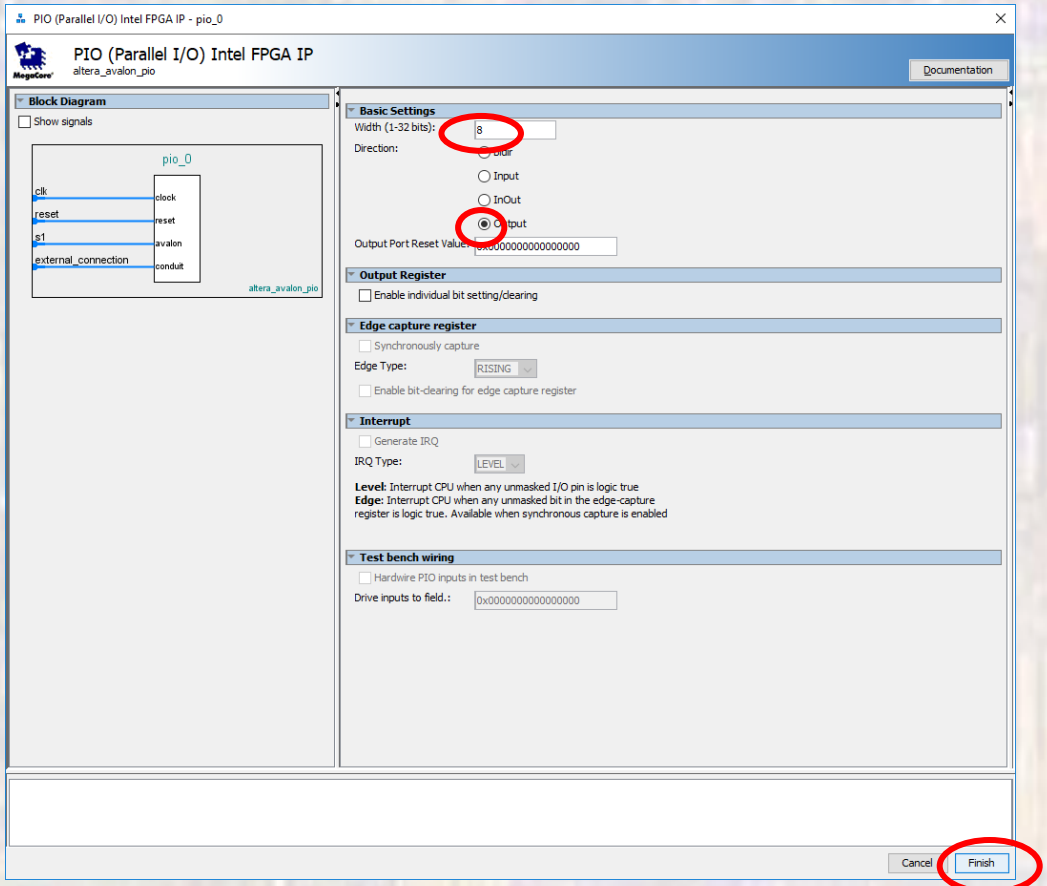
- Customize System
  - Create LED output
    - IP Catalog → Library → Processors and Peripherals→ Peripherals → PIO (Parallel I/O)



 © tj

NIOS-V I/O

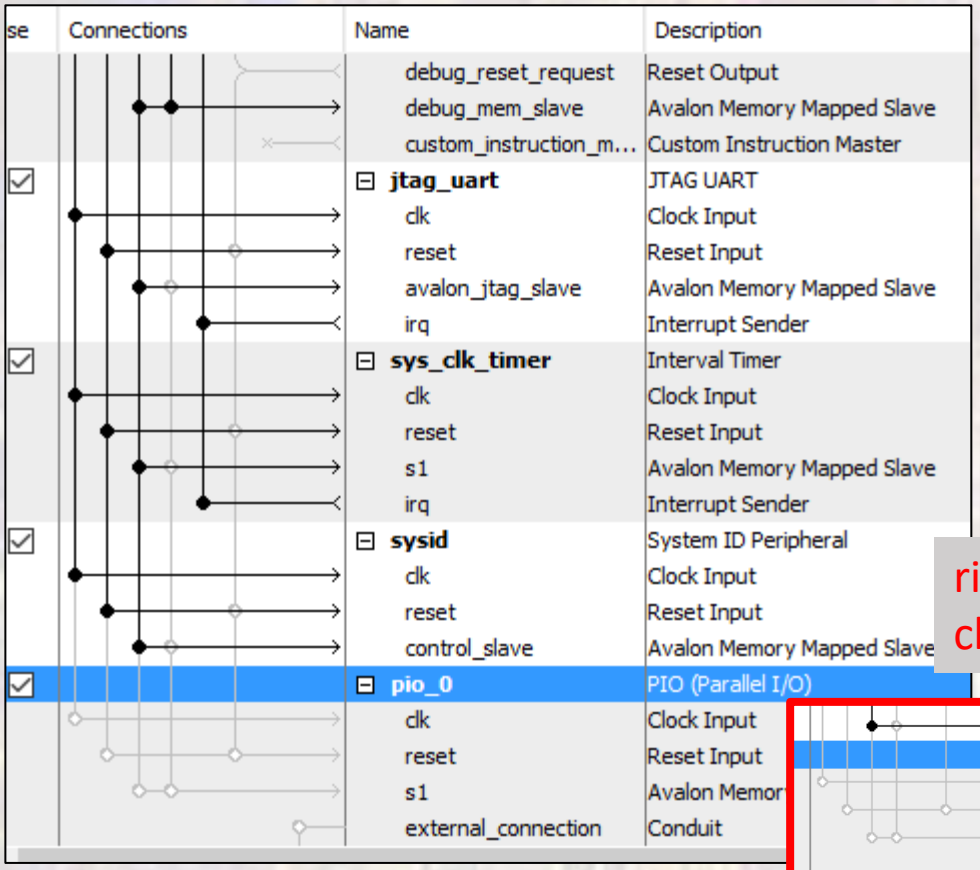
- Customize System
  - Create LED output
    - Set 8 bit, select output



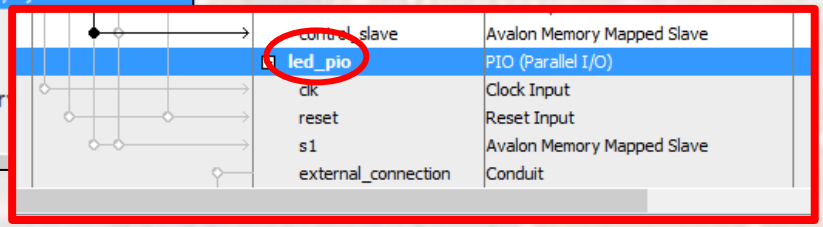
© tj

NIOS-V I/O

- Customize System
  - Hookup LED output



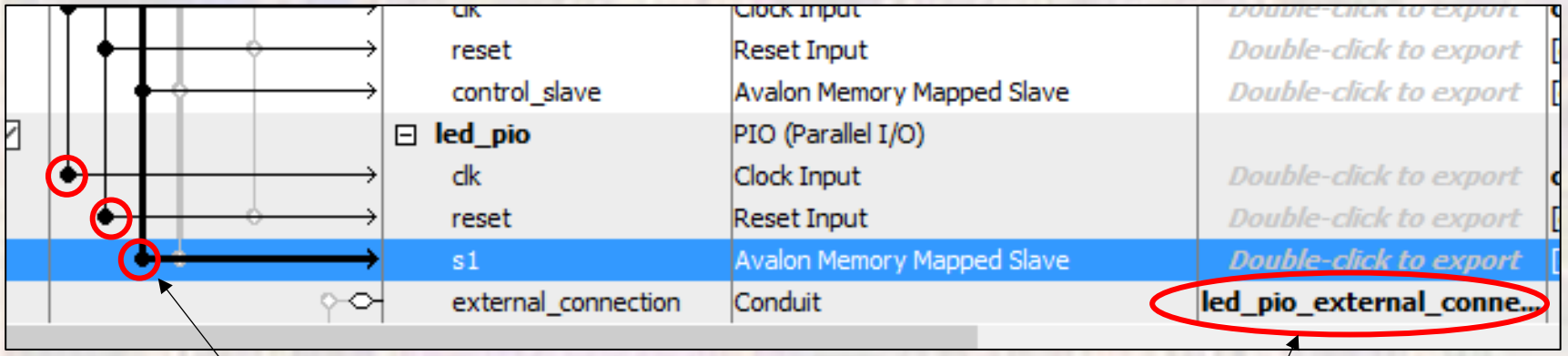
right click the PIO name and change it to led_pio



© tj

NIOS-V I/O

- Customize System
  - Hookup LED output



Connect to data master

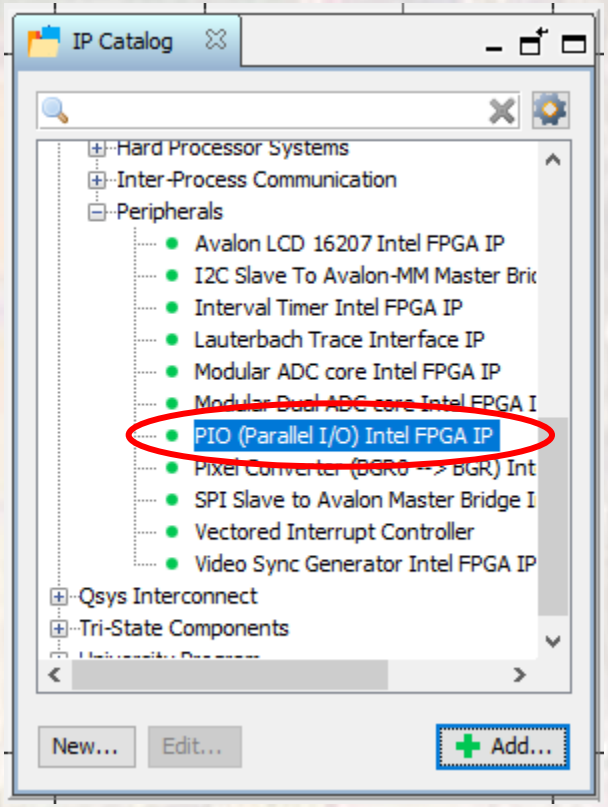
Double click to export

Makes the connection visible outside the system

© tj

NIOS-V I/O

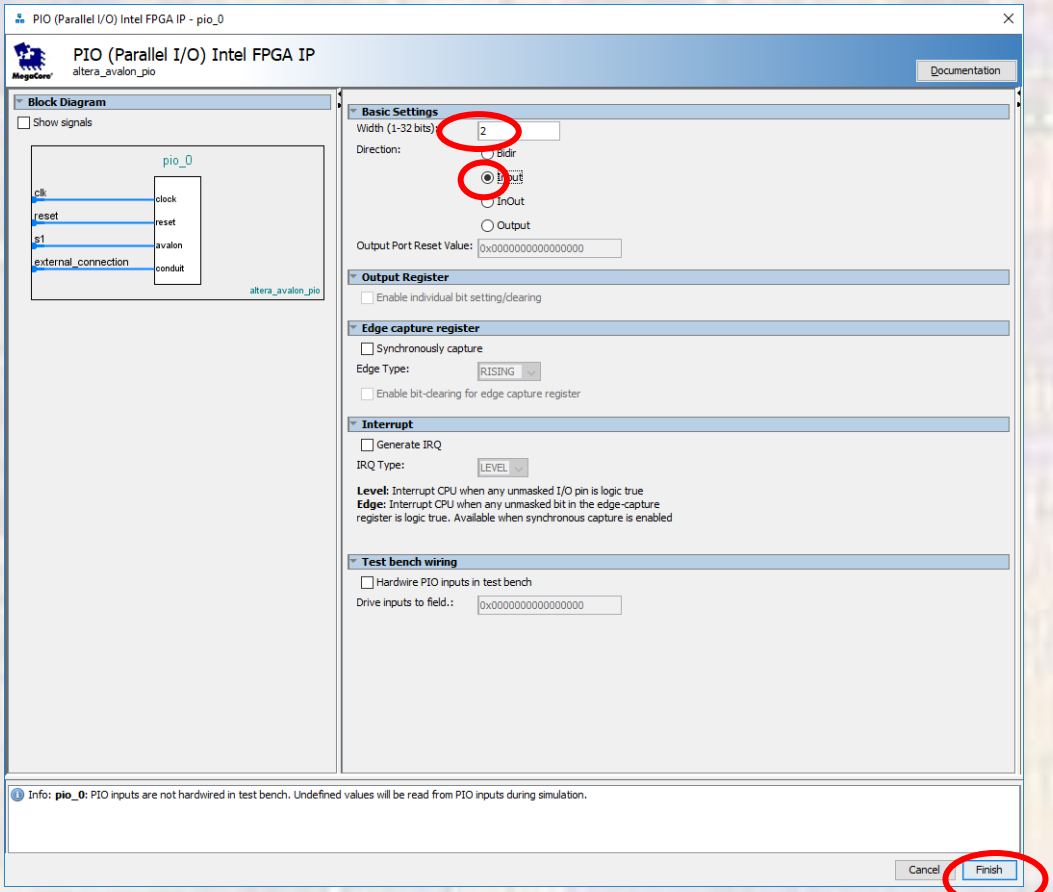
- Customize System
  - Create switch inputs
    - IP Catalog → Library → Processors and Peripherals→ Peripherals → PIO (Parallel I/O)



 © tj

NIOS-V I/O

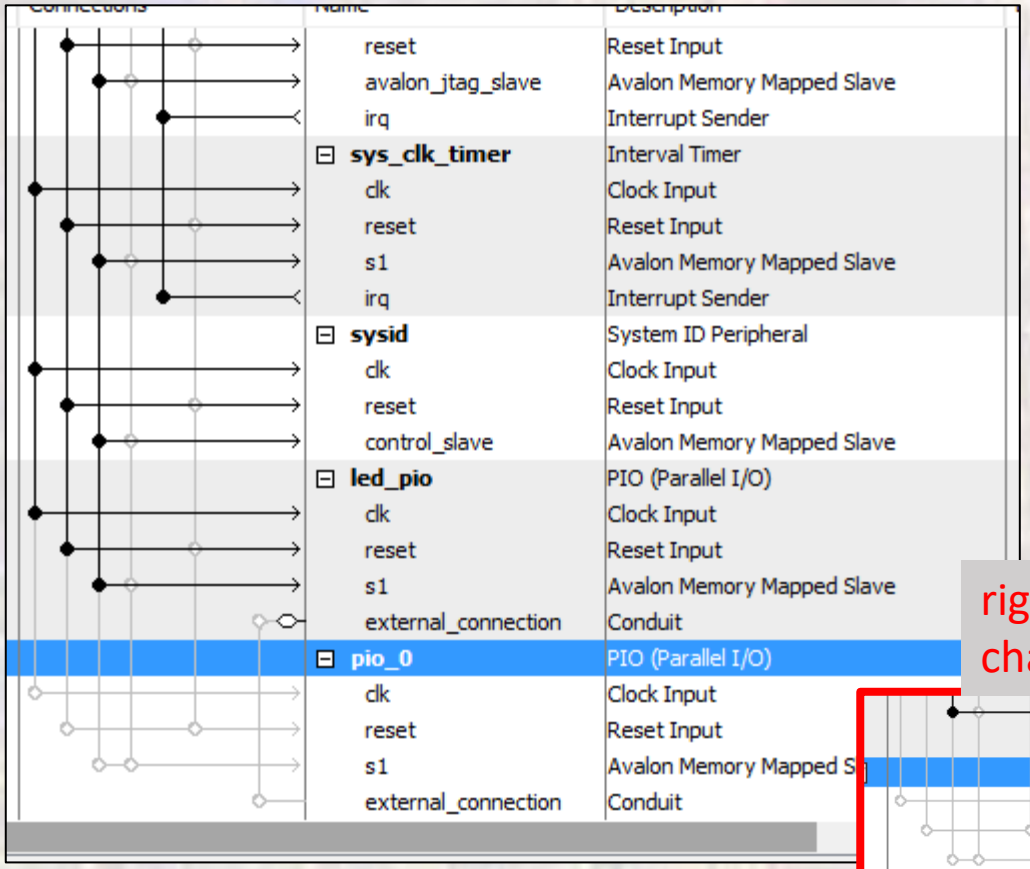
- Customize System
  - Create switch inputs
    - Set 2 bit, select input



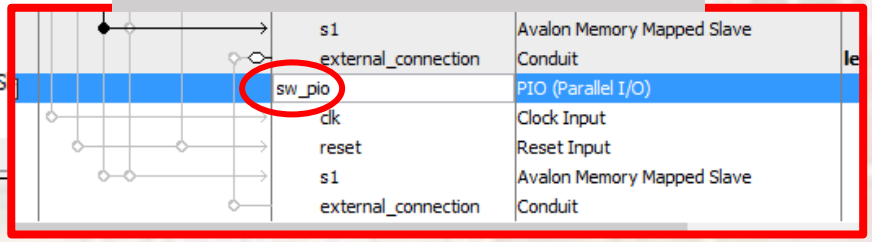
 © tj

NIOS-V I/O

- Customize System
  - Hookup switch input



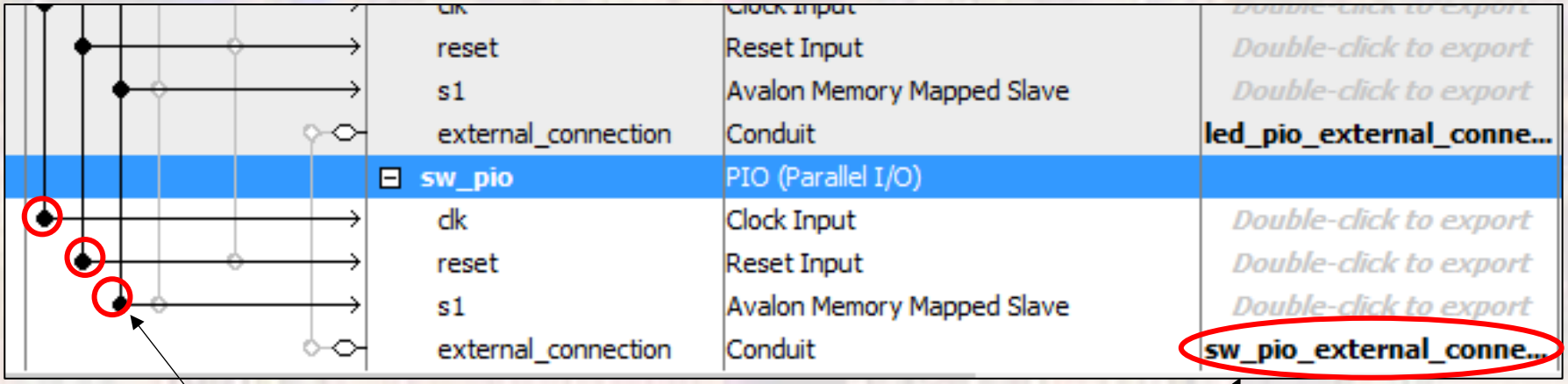
right click the PIO name and change it to sw_pio



© tj

NIOS-V I/O

- Customize System
  - Hookup switch input



Connect to data master

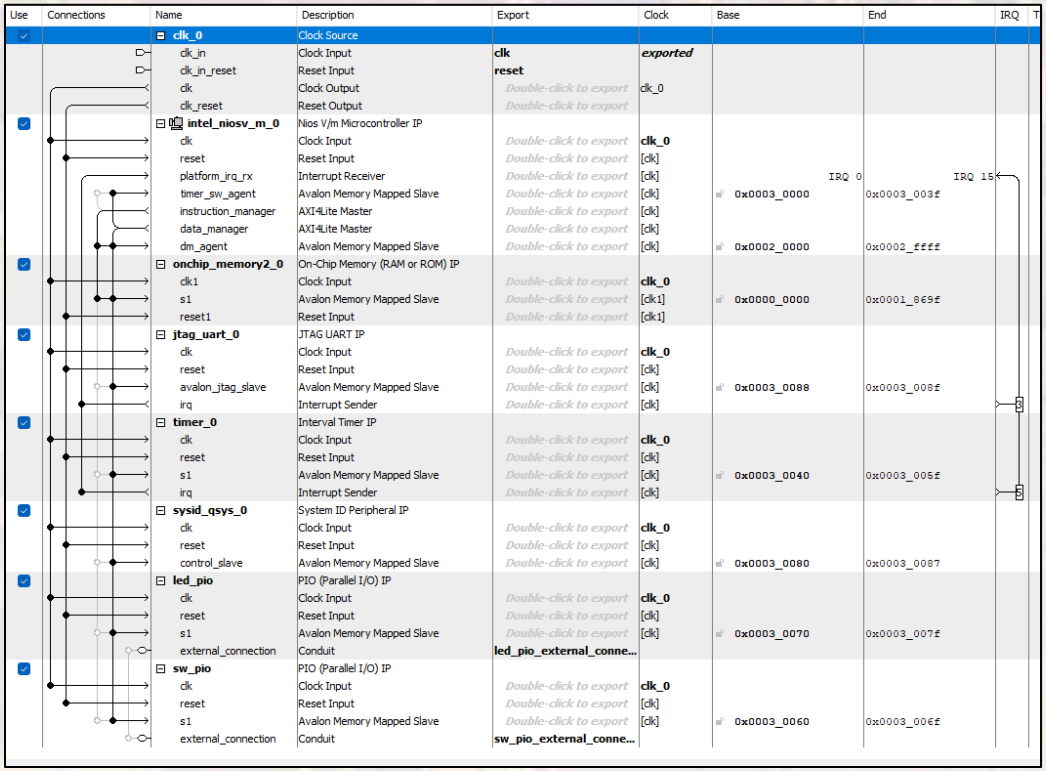
Double click to export

Makes the connection visible outside the system

© tj

NIOS-V I/O

- Complete Custom System
  - Assign base addresses
  - System → Assign Base Addresses

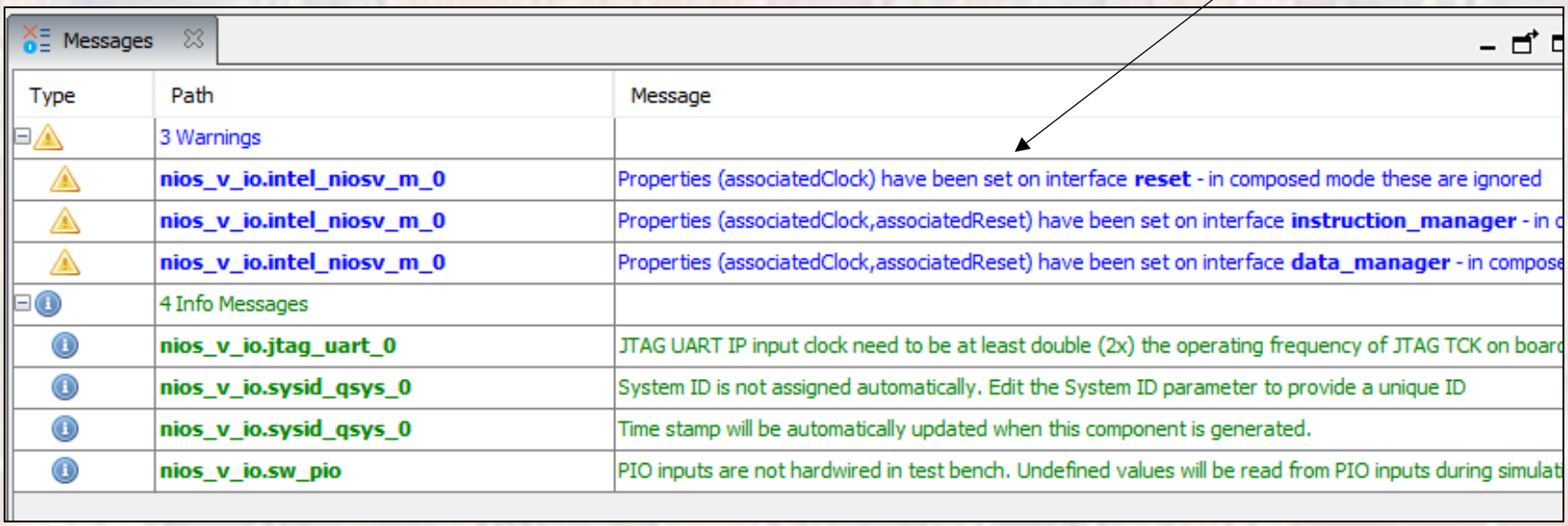


 © tj

NIOS-V I/O

- Create Custom System
  - Check for errors

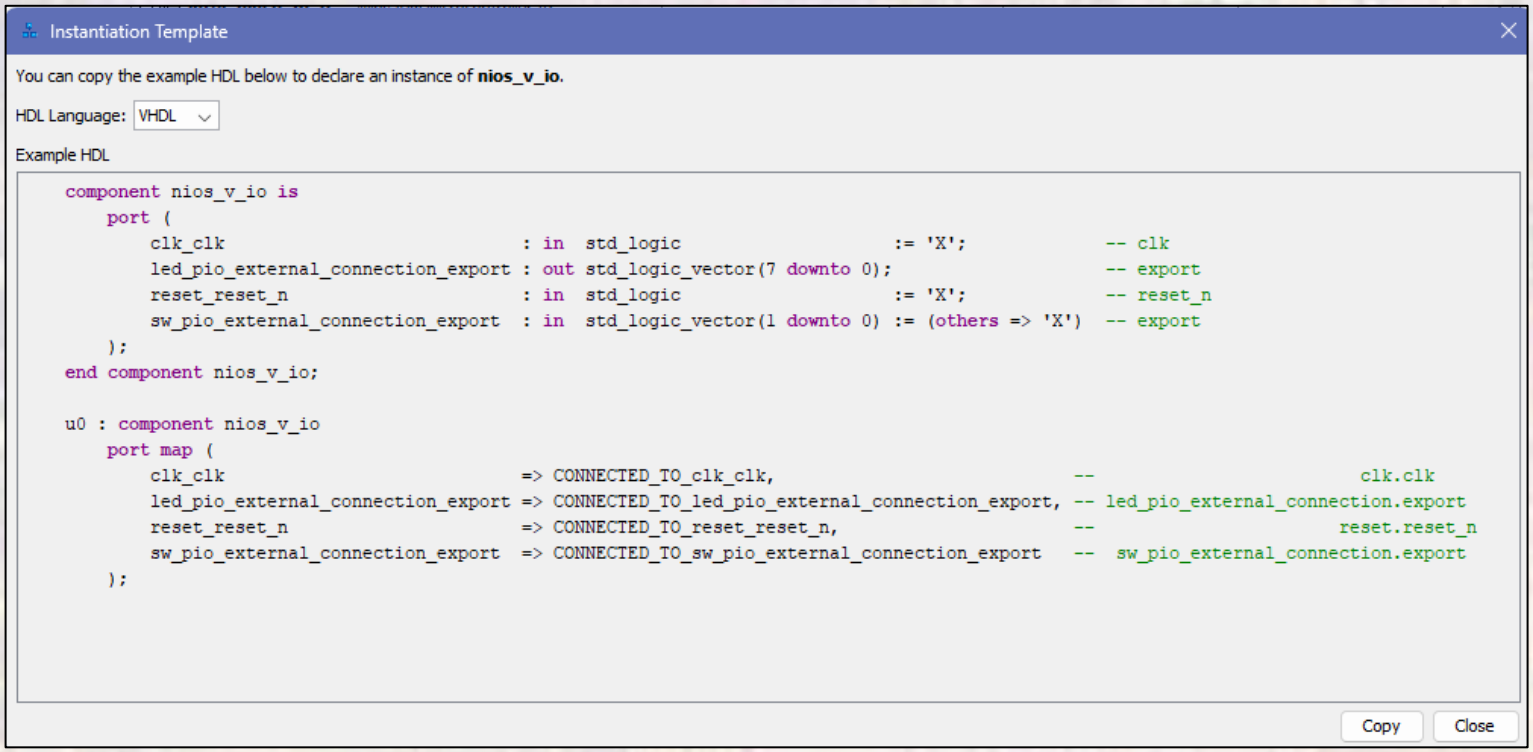
unavoidable



 © tj

NIOS-V I/O

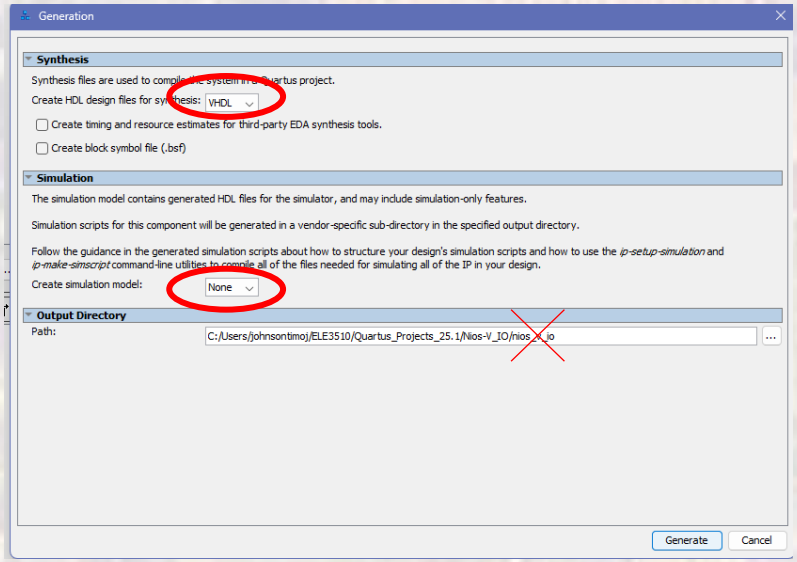
- Create Custom System
  - Create component and instantiation code
    - In Platform Designer: Generate → Show Instantiation Template
    - Copy and Paste into the new design where appropriate



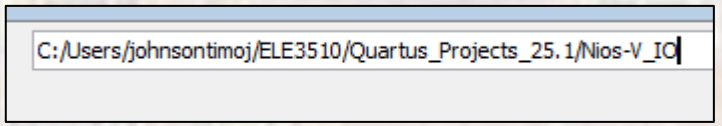
 © tj

NIOS-V I/O

- Create Custom System
  - Save the Platform Designer system
  - Generate the Platform Designer system
    - Generate → Generate HDL
    - The first time you generate you must delete the last directory in the path – don't use the '…'
    - Finish



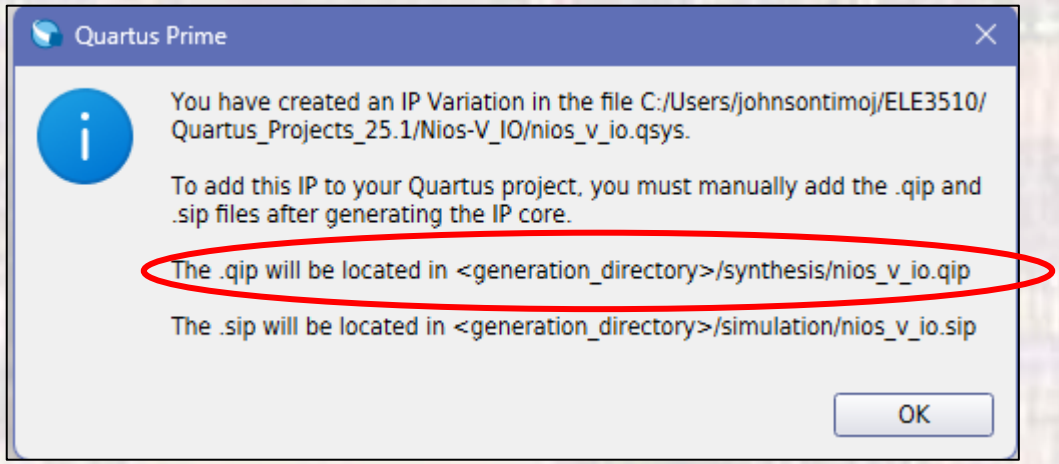
Should point to your project directory



© tj

NIOS-V I/O

- Create Custom System
  - Add the .qip file to the project



 © tj

NIOS-V I/O

- Create DE10 Design
  - Instantiate into a VHDL file

```
-- my nios
library ieee;
use ieee.std_logic_1164.all;

entity nios_v_io_de10 is
   port(
      CLOCK_50:   in std_logic;
      LEDR:       out std_logic_vector(9 downto 0);
      SW:         in std_logic_vector(9 downto 0)
   );
end entity;


   architecture hardware of nios_v_io_de10 is

   component nios_v_io is
     port (
         clk_clk                           : in  std_logic                     := 'X';             -- clk
         led_pio_external_connection_export : out std_logic_vector(7 downto 0);                    -- export
         reset_reset_n                     : in  std_logic                     := 'X';             -- reset_n
         sw_pio_external_connection_export  : in  std_logic_vector(1 downto 0) := (others => 'X')  -- export
     );
   end component nios_v_io;

begin

   u0 : component nios_v_io
     port map (
         clk_clk                           => CLOCK_50,          -- clk.clk
         led_pio_external_connection_export => LEDR(7 downto 0), -- led_pio_external_connection.export
         reset_reset_n                     => '1',               -- reset.reset_n
         sw_pio_external_connection_export  => SW(1 downto 0)     -- sw_pio_external_connection.export
     );

end architecture;
```


© tj

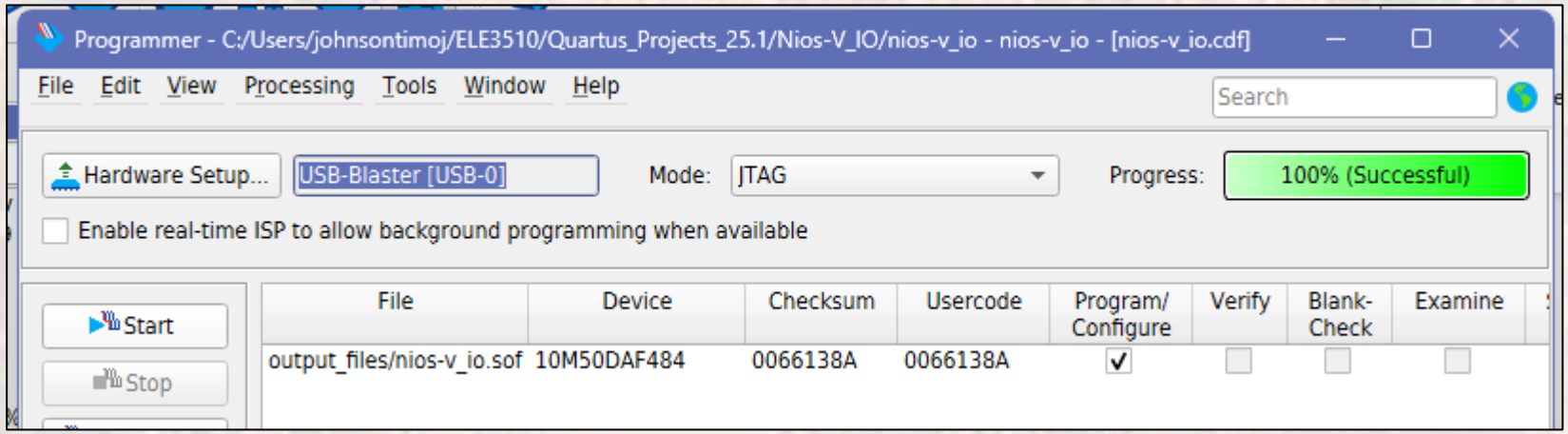
NIOS-V I/O

- Create DE10 Design
  - Prepare to synthesize
    - If you did not do these when you created the project be sure to do them now
      - assignments → device → device and Pin options
        - Single Uncompressed with memory initialization
      - Import the pin aliases (qsf file)
      - Setup the SDF file
  - Be sure to set your top level entity
  - Start Compilation

© tj

NIOS-V I/O

- Create DE10 Design
  - Complete the HW setup
    - Download the HW project onto the board



 © tj

NIOS I/O

- Create the BSP System
  - Follow the notes on NIOS V Software Setup

 © tj

NIOS-V I/O

- BSP Features
  - Open system.h in the hal_bsp folder
    - scroll down to the PIOs
    - These define software parameters for the PIOs

```
/*
 * sw_pio configuration
 *
 */

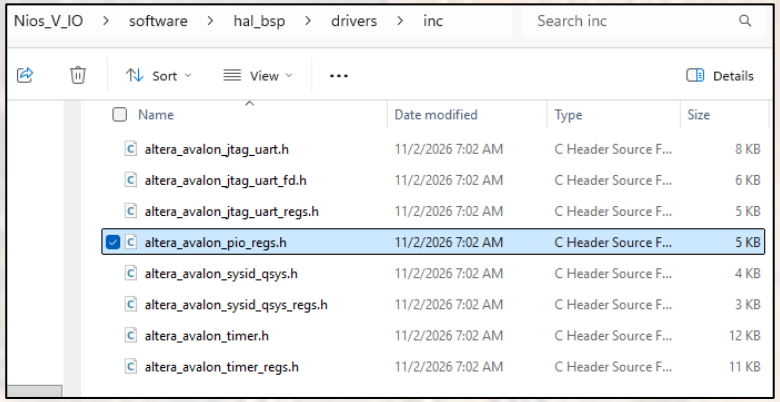
#define ALT_MODULE_CLASS_sw_pio altera_avalon_pio
#define SW_PIO_BASE 0x30060
#define SW_PIO_BIT_CLEARING_EDGE_REGISTER 0
#define SW_PIO_BIT_MODIFYING_OUTPUT_REGISTER 0
#define SW_PIO_CAPTURE 0
#define SW_PIO_DATA_WIDTH 2
#define SW_PIO_DO_TEST_BENCH_WIRING 0
#define SW_PIO_DRIVEN_SIM_VALUE 0
#define SW_PIO_EDGE_TYPE "NONE"
#define SW_PIO_FREQ 50000000
#define SW_PIO_HAS_IN 1
#define SW_PIO_HAS_OUT 0
#define SW_PIO_HAS_TRI 0
#define SW_PIO_IRQ -1
#define SW_PIO_IRQ_INTERRUPT_CONTROLLER_ID -1
#define SW_PIO_IRQ_TYPE "NONE"
#define SW_PIO_NAME "/dev/sw_pio"
#define SW_PIO_RESET_VALUE 0
#define SW_PIO_SPAN 16
#define SW_PIO_TYPE "altera_avalon_pio"
```

```
/*
 * led_pio configuration
 *
 */

#define ALT_MODULE_CLASS_led_pio altera_avalon_pio
#define LED_PIO_BASE 0x30070
#define LED_PIO_BIT_CLEARING_EDGE_REGISTER 0
#define LED_PIO_BIT_MODIFYING_OUTPUT_REGISTER 0
#define LED_PIO_CAPTURE 0
#define LED_PIO_DATA_WIDTH 8
#define LED_PIO_DO_TEST_BENCH_WIRING 0
#define LED_PIO_DRIVEN_SIM_VALUE 0
#define LED_PIO_EDGE_TYPE "NONE"
#define LED_PIO_FREQ 50000000
#define LED_PIO_HAS_IN 0
#define LED_PIO_HAS_OUT 1
#define LED_PIO_HAS_TRI 0
#define LED_PIO_IRQ -1
#define LED_PIO_IRQ_INTERRUPT_CONTROLLER_ID -1
#define LED_PIO_IRQ_TYPE "NONE"
#define LED_PIO_NAME "/dev/led_pio"
#define LED_PIO_RESET_VALUE 0
#define LED_PIO_SPAN 16
#define LED_PIO_TYPE "altera_avalon_pio"
```

NIOS-V I/O

- BSP Features
  - Expand drivers → inc in the hal_bsp folder
    - Open altera_Avalon_pio_regs.h
    - PIO command are defined here




```
#ifndef __ALTERA_AVALON_PIO_REGS_H__
#define __ALTERA_AVALON_PIO_REGS_H__

#include <io.h>

#define IOADDR_ALTERA_AVALON_PIO_DATA(base)             __IO_CALC_ADDRESS_NATIVE(base, 0)
#define IORD_ALTERA_AVALON_PIO_DATA(base)               IORD(base, 0)
#define IOWR_ALTERA_AVALON_PIO_DATA(base, data)         IOWR(base, 0, data)

#define IOADDR_ALTERA_AVALON_PIO_DIRECTION(base)        __IO_CALC_ADDRESS_NATIVE(base, 1)
#define IORD_ALTERA_AVALON_PIO_DIRECTION(base)          IORD(base, 1)
#define IOWR_ALTERA_AVALON_PIO_DIRECTION(base, data)    IOWR(base, 1, data)

#define IOADDR_ALTERA_AVALON_PIO_IRQ_MASK(base)         __IO_CALC_ADDRESS_NATIVE(base, 2)
#define IORD_ALTERA_AVALON_PIO_IRQ_MASK(base)           IORD(base, 2)
#define IOWR_ALTERA_AVALON_PIO_IRQ_MASK(base, data)     IOWR(base, 2, data)

#define IOADDR_ALTERA_AVALON_PIO_EDGE_CAP(base)         __IO_CALC_ADDRESS_NATIVE(base, 3)
#define IORD_ALTERA_AVALON_PIO_EDGE_CAP(base)           IORD(base, 3)
#define IOWR_ALTERA_AVALON_PIO_EDGE_CAP(base, data)     IOWR(base, 3, data)


#define IOADDR_ALTERA_AVALON_PIO_SET_BIT(base)         __IO_CALC_ADDRESS_NATIVE(base, 4)
#define IORD_ALTERA_AVALON_PIO_SET_BITS(base)           IORD(base, 4)
#define IOWR_ALTERA_AVALON_PIO_SET_BITS(base, data)     IOWR(base, 4, data)

#define IOADDR_ALTERA_AVALON_PIO_CLEAR_BITS(base)       __IO_CALC_ADDRESS_NATIVE(base, 5)
#define IORD_ALTERA_AVALON_PIO_CLEAR_BITS(base)         IORD(base, 5)
#define IOWR_ALTERA_AVALON_PIO_CLEAR_BITS(base, data)   IOWR(base, 5, data)


/* Defintions for direction-register operation with bi-directional PIOs */
#define ALTERA_AVALON_PIO_DIRECTION_INPUT  0
#define ALTERA_AVALON_PIO_DIRECTION_OUTPUT 1

#endif /* __ALTERA_AVALON_PIO_REGS_H__ */
```


© tj

NIOS-V I/O

- Verify the system
  - Follow the notes on NIOS V Software Setup
  - Verify “hello” message is printed
  - terminate the console and the debugger (red boxes or open the debug tab and rt-click → terminate om both of them)

© tj

NIOS-V I/O

- Applications Program
  - Edit your verification code

    or
  - Create a new code file

    - Uses SW0 as a pause
    - Uses SW1 as up/dn

```
/*
/*
 * count.c
 *
 *  Created on: Sep 19, 2018
 *      Author: johnsontimoj
 */
//////////////////////////////
//
// Example file using
// NIOS with LEDs and switches
//
//////////////////////////////

#include "altera_avalon_pio_regs.h"
#include "system.h"
#include <stdio.h>
#include <unistd.h>

int main(){
  printf("My count program!\n");
  alt_u8 count = 0;
  alt_u8 sw;

  while(1){
        // output the count to the LEDs
        IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, count);

        // read the switches
        sw = IORD_ALTERA_AVALON_PIO_DATA(SW_PIO_BASE);

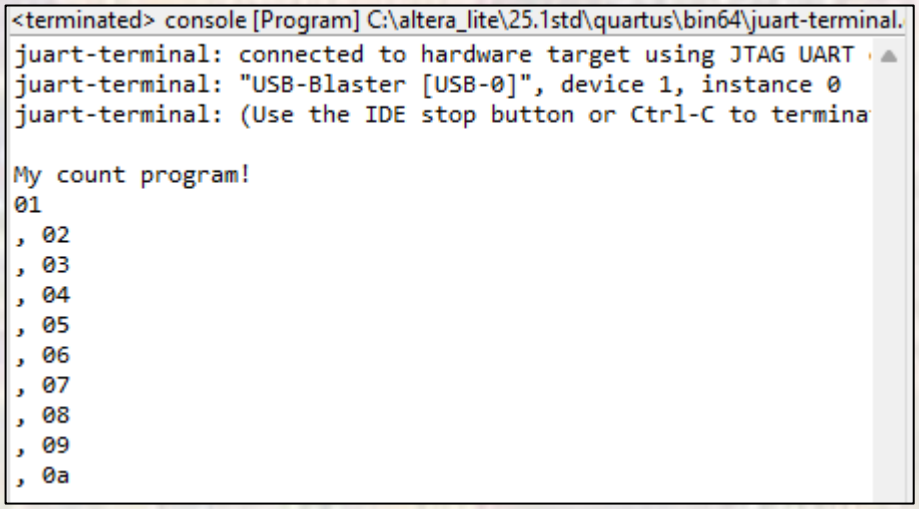
        // count up/dn/pause
        if(sw & 0x01){        // pause
              if(!(sw & 0x02))      // up/dn
                    count++;
              else
                    count--;
              printf("%02x,  ", count);
              usleep(500000);
        }// end if

        // delay before restarting
        if( count == 0xff ){
              printf("\nWaiting...");
              int i;
              for (i = 0; i<6; ++i){
                  usleep(500000); /* Sleep for 3s. */
              }
        }// end if
  }// end while
  return 0;
}// end main
```


© tj

NIOS I/O

- Run
  - Re-build the project
    - Rt-click on the app project → Build Project
  - Follow the directions in NIOS V Software Setup to debug the project in the Ashling console or the juart-terminal



 © tj