BeMicro SDK - BeInMotion Motor Control Design Lab

August 5, 2011

ARROW ELECTRONICS, INC.

TABLE OF CONTENTS

TABLE OF CONTENTS CONTINUED

OVERVIEW

What Does It Take To Create Your Own Motor Control System?

This lab teaches you how to create motor control systems implemented in programmable logic. You will build a processor-based hardware system and run software on it. You will see how quick and easy it is to build motor control systems using Altera's Qsys and the Nios II EDS to configure and integrate pre-verified IP blocks.

The reference demos include programming the system as an autonomous vehicle with wall and edge avoidance and synchronized movement. Simple PID control will be used to influence the DC motor's speed and position. The stepper motor will be controlled based on full step, half step and micro-stepping modes. A final demo has been included in the appendix to monitor the charge remaining in the battery system and display the value on the LCD.

Lab Notes:

Many of the names that the lab asks you to choose for files, components, and other objects in this exercise must be spelled *exactly* as directed.

This nomenclature is necessary because the pre-written software application includes variables that use the names of the hardware peripherals. Naming the components differently can cause the software application to fail.

There are also other similar dependencies within the project that require you to enter the correct names.

No IP licenses are required to run these reference demos.

For technical support or questions, please review Appendix D "Troubleshooting"

- **NOTE:** This lab guide requires an Arrow Electronics BeInMotion FPGA-based Motor Control Kit (www.arrow.com/beinmotion).

MODULE 1: GETTING STARTED

Module Objective

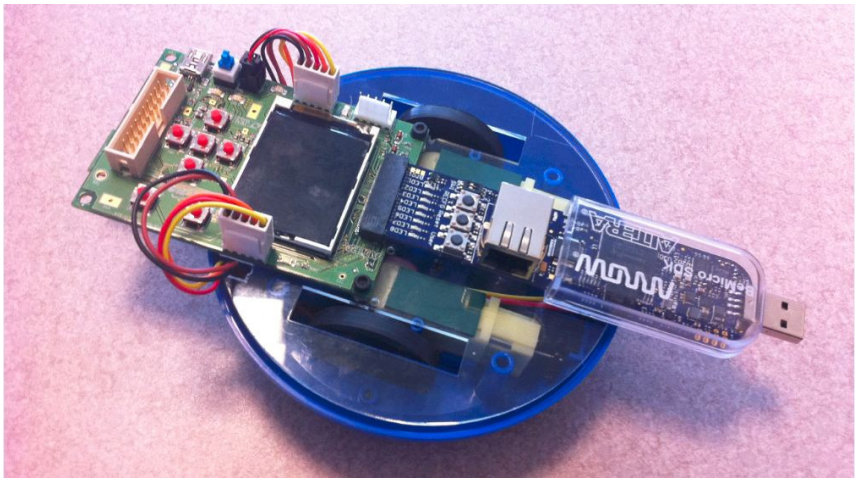
Your first objective is to ensure that you have all of the items needed and to install the tools so that you are ready to create and run your design.

List of required items

- Arrow Electronics BeInMotion Development Kit
- Design Software Quartus® II v11.0 and Nios® II EDS 11.0
- Windows PC (XP/SP2, Vista/32 or Win7), 866MHz or faster, minimum of 1GB RAM, 5GB hard disk space.
- BeInMotion Lab Design Files

1.1 ACQUIRE THE BEINMOTION DEVELOPMENT KIT

This development kit can be ordered from **http://www.arrow.com/beinmotion**. It consists of two pieces, the BeMicro SDK development kit and the BeInMotion motorized base.



1.2 INSTALL THE ALTERA DESIGN SOFTWARE

You will need to install the following design software packages:

1. **Quartus II 11.0 Web Edition design software** – FPGA synthesis and compilation tool that contains Qsys and the MegaCore IP library with the Nios II processor IP core

2. **Nios II 11.0 Embedded Design Suite** – A complete, integrated environment for software development

If you already have both Quartus II and the Nios II EDS installed on your machine, you may skip ahead to Section 1.3 to extract the lab files.

INSTALLING FROM THE DVD-ROM: Please skip ahead to step 4 of the installation instructions.

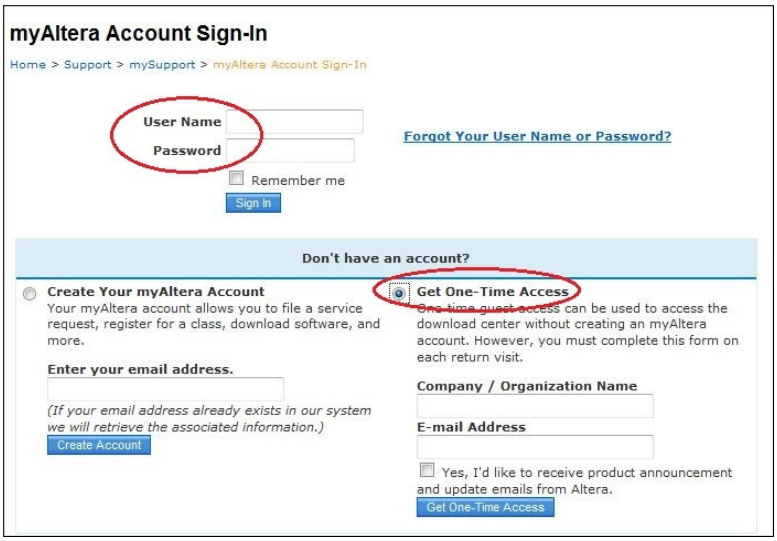
INSTALLING FROM THE WEB: Please follow steps 1 through 4 of the installation instructions.

The Web Edition can be downloaded from the Altera web site. **PLEASE CAREFULLY FOLLOW THE STEPS SHOWN BELOW**.

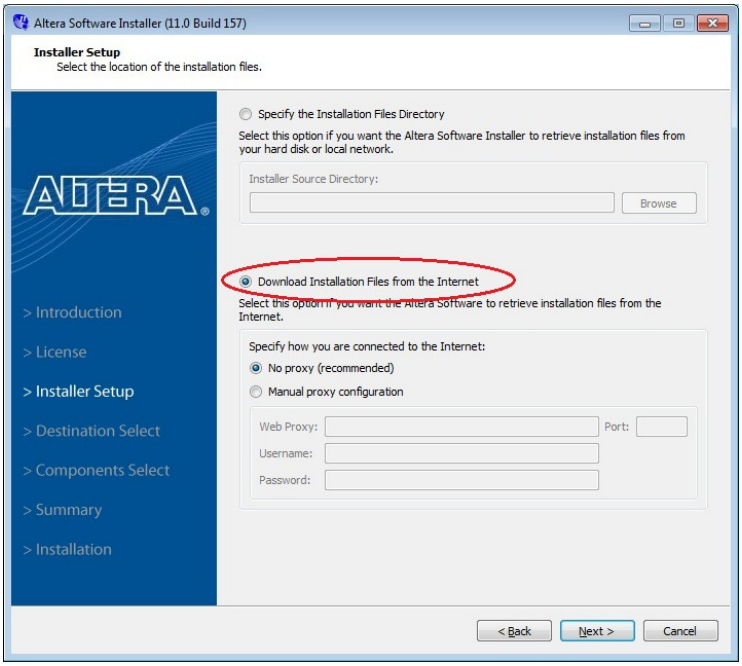
1. Go to the Altera Download web page at **https://www.altera.com/download/dnl-index.jsp**

2. Select **Download Windows Version (13 MB)**

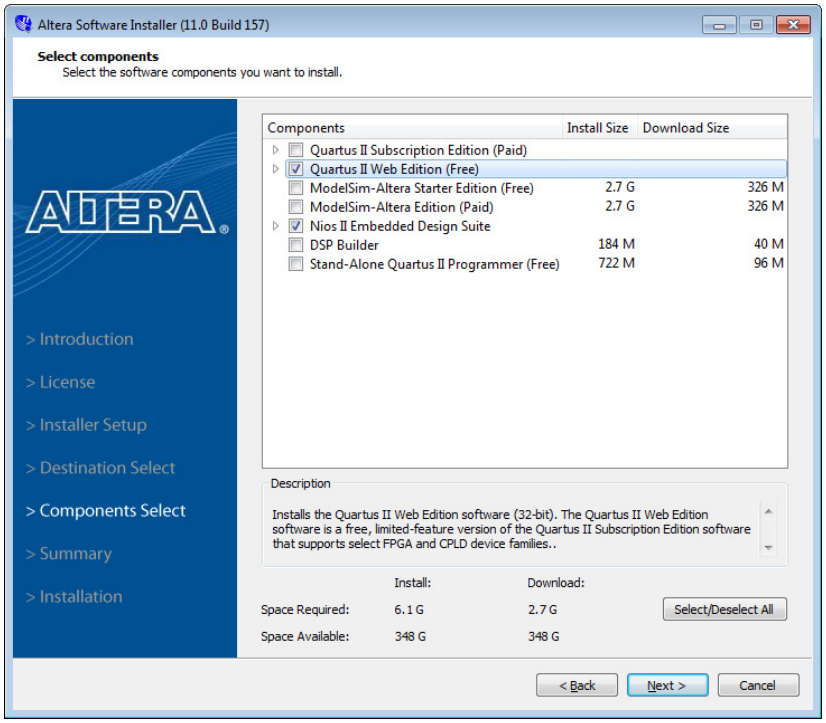
3. Log-in to **myAltera** using your login credentials OR select **Get One-Time Access**



4. Altera Installer Setup. Run the Altera Installer and Navigate to the **Installer Setup** page. Select the **Download Installation Files from the Internet** radio button



5. Select Components. Select Quartus II Web Edition and Nios II Embedded Design Suite components for download.



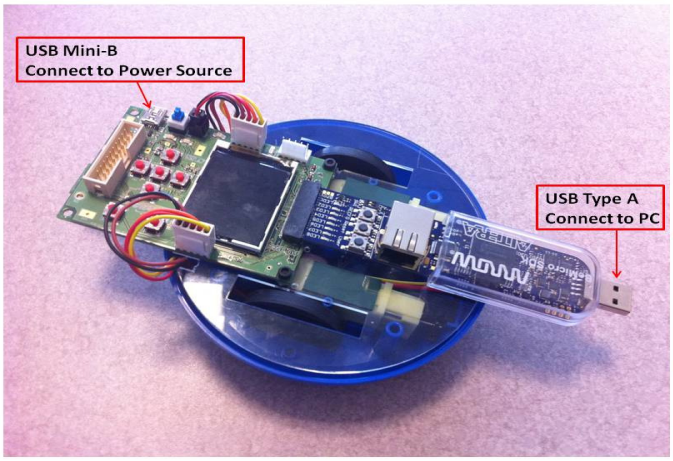
1.3 EXTRACT THE BEINMOTION LAB FILES

Download **BeInMotion.zip** from http://www.arrow.com/beinmotion

Create a folder called **altera_trn** on your PC (eg. C:\altera_trn).

Extract the contents of BeInMotion.zip to the **altera_trn** folder on your PC. Make sure there are ***NO SPACES*** in the path name. This will create a **BeInMotion_lab** folder, which will be the root folder for the labs.

1.4 INSTALL THE USB-BLASTER DEVICE DRIVER



Connect the BeMicro SDK to the 80-pin edge connector on the motor base. Plug the Type A USB port to your PC. It is recommended to use a USB extension cable to reach from the PC to kit. Once connected, your Windows PC will find the new hardware and a message will come up and request that the driver needs to be installed.

- **NOTE:** The Mini-B USB connector on the BeInMotion Motor Base is used to charge the battery. Use an AC adaptor with USB power output of 5V at 0.5A or greater. LED1, a green LED on the motor base, will illuminate when the battery is charging and turn off once charging is complete. The battery will charge with the power switch on or off. Power supplied through the USB Type A port will power the kit, but will not charge the battery.

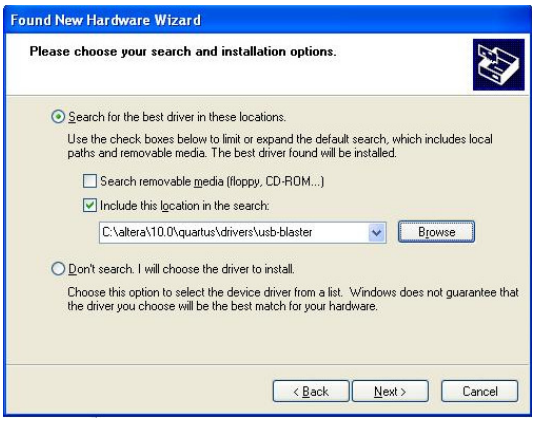
Windows XP

Windows 7

Windows XP USB-Blaster Driver Installation

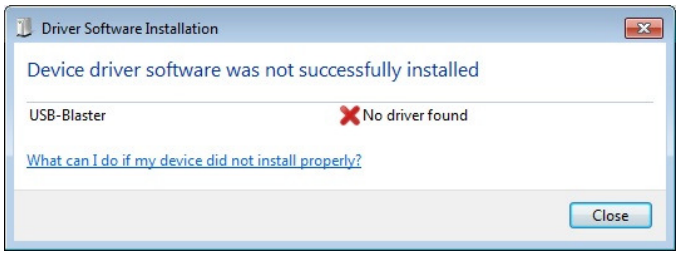
Select “Install from a list or specific location (Advanced)” and continue through the wizard.

In the next dialogue box point the wizard to the drivers which can be found in your Quartus II installation directory under “<install directory>\11.0\quartus\drivers\usb-blaster”. If Windows presents you with a message that the drivers have not passed Windows Logo testing, please click “Continue Anyway”.

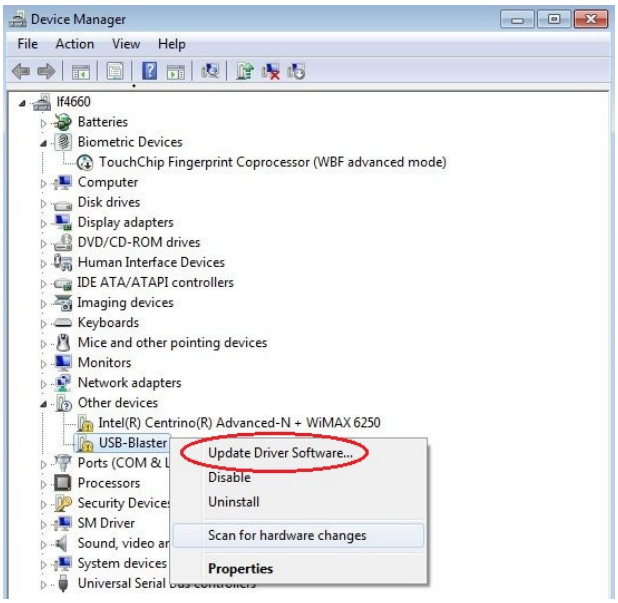


Windows 7 USB-Blaster Driver Installation

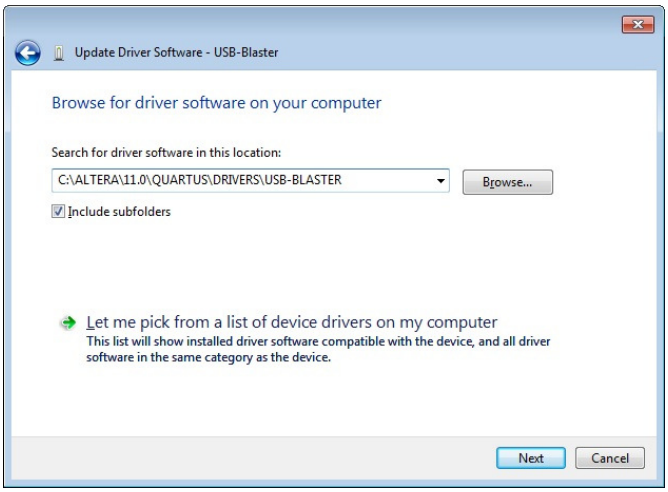
If Windows 7 does not detect the driver, it can be installed manually through the Device Manager.

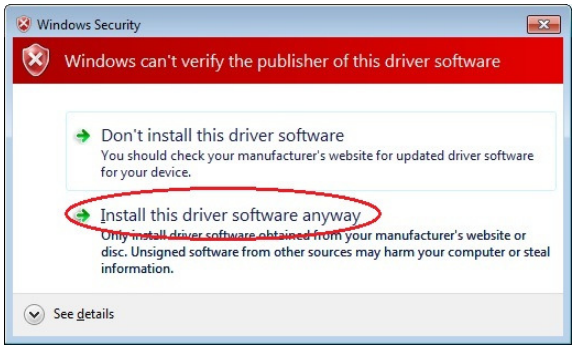


Open the Device Manager and Right Click on **USB-Blaster**. Select **Update Driver Software...**



In the next dialogue box select **Browse my computer for driver software.** Point the wizard to the drivers which can be found in your Quartus II installation directory under "<install directory>\11.0\quartus\drivers\usb-blaster". Check the box to **Include Subfolders**. If Windows presents you with a message that the publisher can't be verified, please click "**Install this driver software anyway**".





CONGRATULATIONS!!

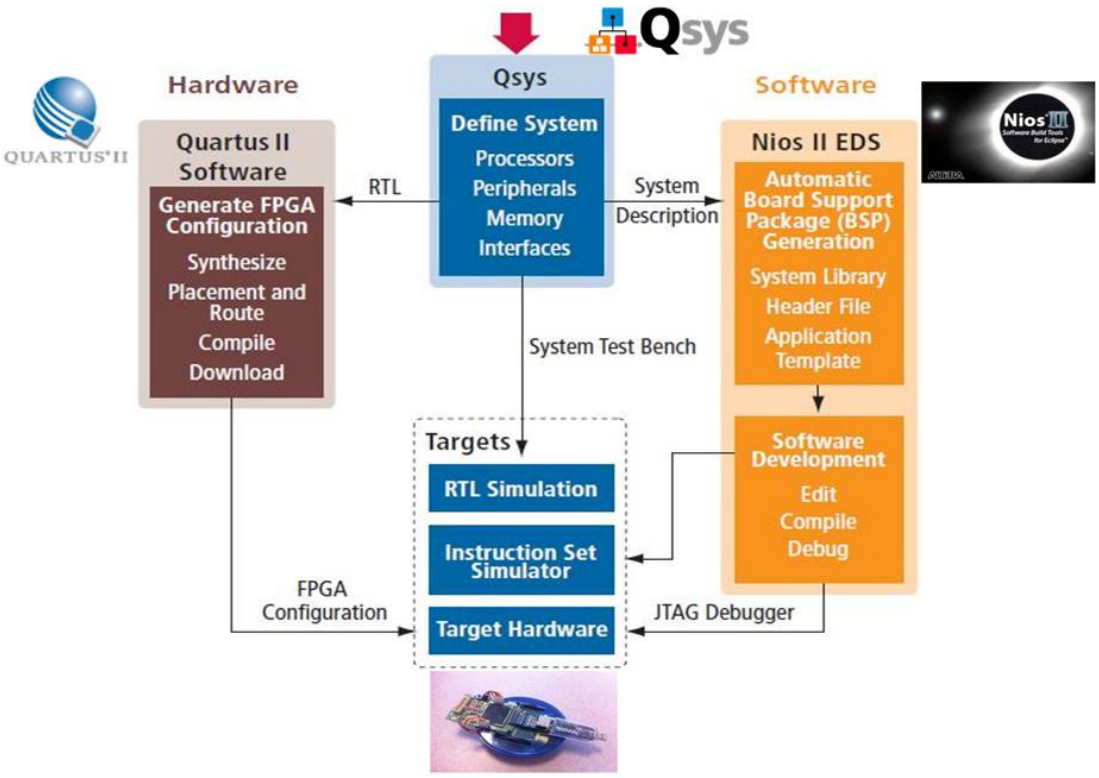
You have just completed all the setup and installation requirements and are now ready to examine the system-level design.

MODULE 2: EXAMINE THE SYSTEM DESIGN

Module Objective

Developing software for an Altera System on a Programmable Chip (SOPC) requires an understanding of the design flow between the Qsys system tool and the Nios II Embedded Design Suite (EDS). Typically, design requirements begin with customer requirements and become inputs to the system definition. System definition is hence the first step in the design flow process. For this lab, we will generate the system definition and design is complete and an FPGA image derived from that has been flashed into the BeMicro SDK kit. Our objective is to learn how to use the Nios II EDS to build software projects for this system. The objective of this module is to examine the system architecture and development tools that you will be using today.

2.1 EXAMINE THE SYSTEM TOOL FLOW



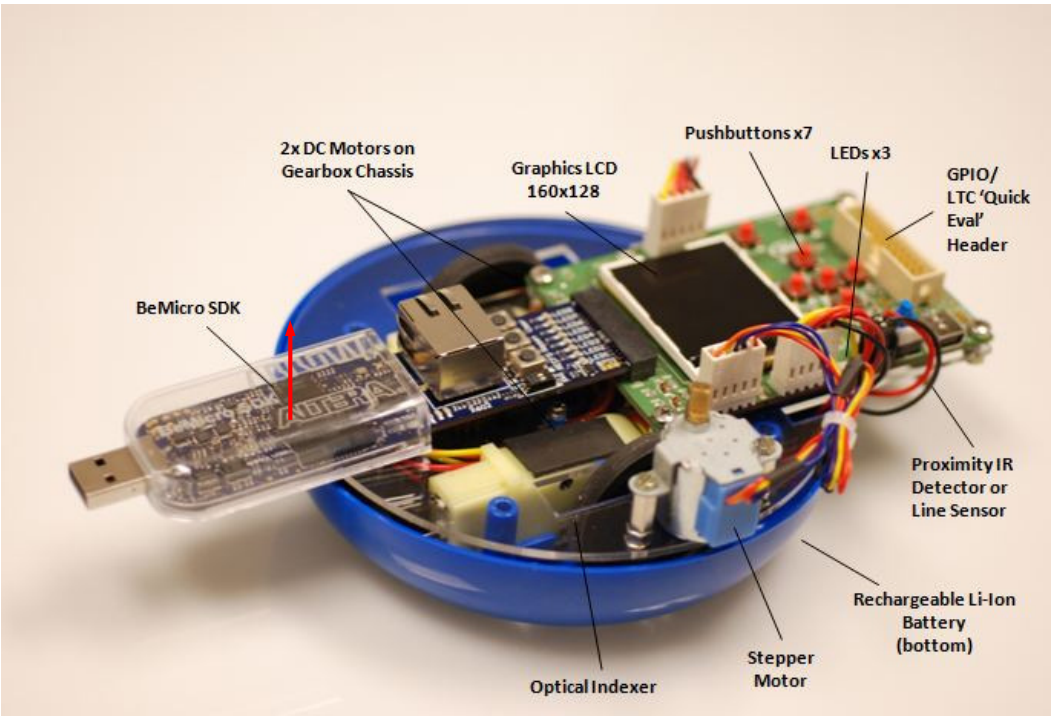
The above diagram depicts the typical flow for system design. System definition is performed using Qsys. The results are two-fold:

System description that the Nios II Embedded Design Suite, the software design tool, uses to create a new project for the software application.

- HDL files for the system that are used by the Quartus II FPGA design software to compile and generate the hardware system.

The output of the Hardware Flow is an FPGA image that is used to configure the FPGA. The output of the Software Flow is an executable from which the Nios II processor executes instructions.

2.2 EXAMINE THE BEINMOTION KIT



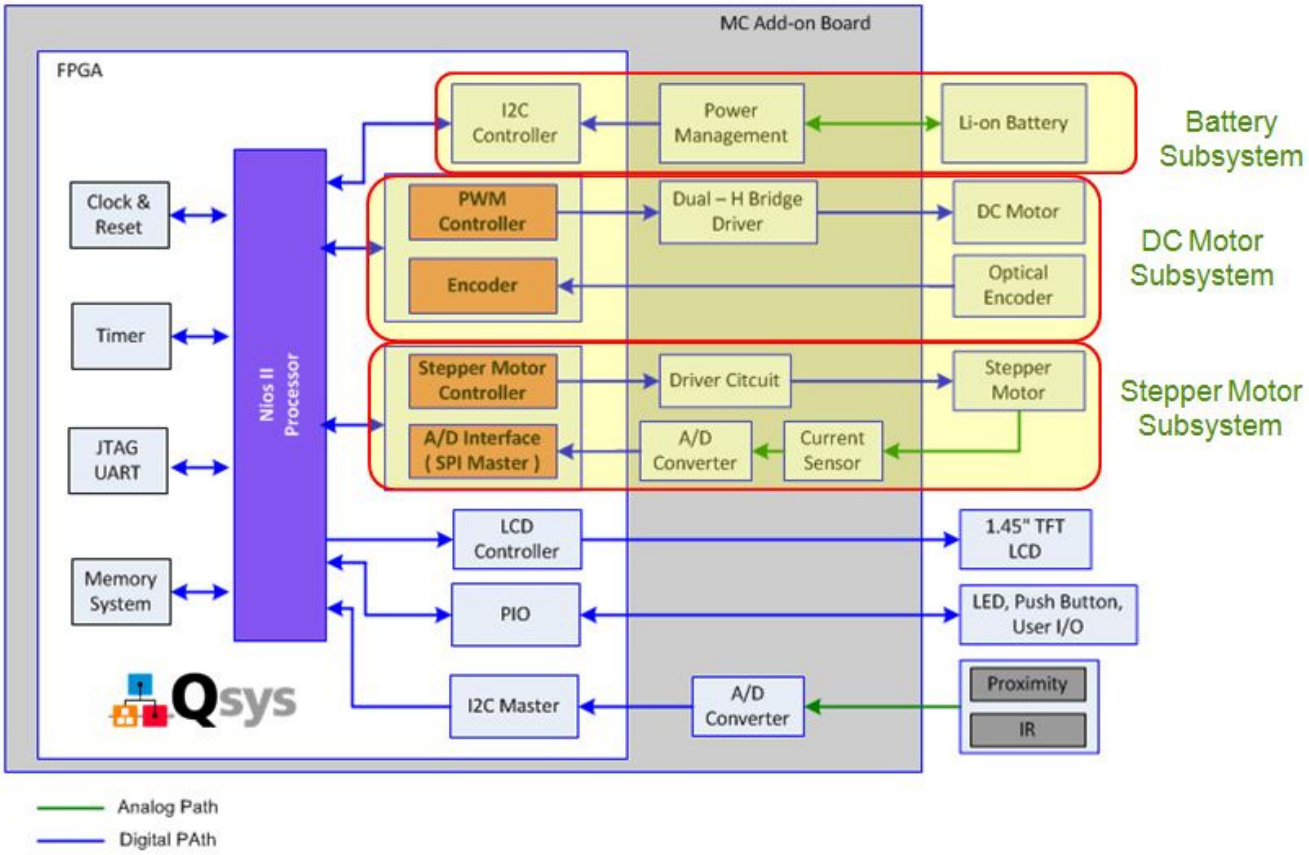
This shows the major components of the BeInMotion kit. Featured components include Arrow Electronics' BeMicro SDK development kit with Altera Cyclone IV-E FPGA and the BeInMotion Motor Base designed by System Level Solutions.

Other featured components include:

- Proximity IR Detector — Avago APDS9800
- Proximity A/D Converter — Linear Technology LTC2485
- Battery Gas Gauge — Linear Technology LTC2942-1
- Stepper Feedback Current Sensor — Linear Technology LTC1999
- Stepper Feedback A/D Converter — Linear Technology LTC1865

2.3 SYSTEM ARCHITECTURE

The BeInMotion kit is architected as shown in this sketch:



The system above can be created in Qsys using a standard library of re-useable IP blocks. The System Interconnect Fabric is automatically generated by Qsys and binds the blocks together. The system interconnect manages dynamic bus-width matching, interrupt priorities, arbitration and address mapping. This system is a full-featured processor system capable of running operating systems such as uC-OSII or Linux.

The following pages will guide you through the process of building the embedded system shown above.

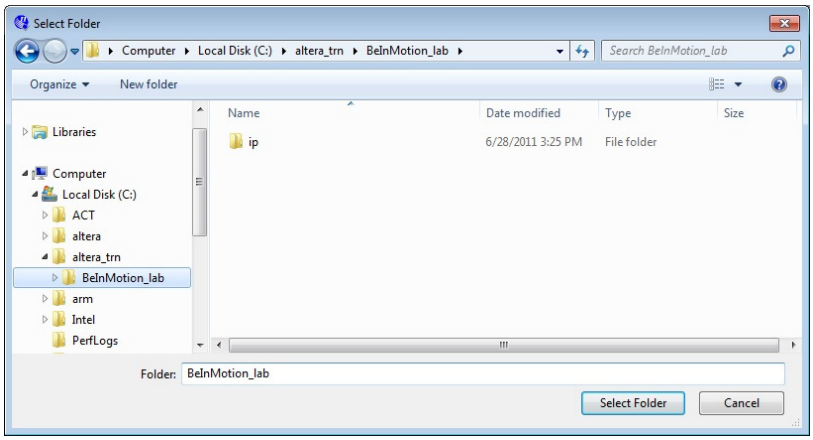
MODULE 3: CREATE A NEW QUARTUS II PROJECT

Module Objective

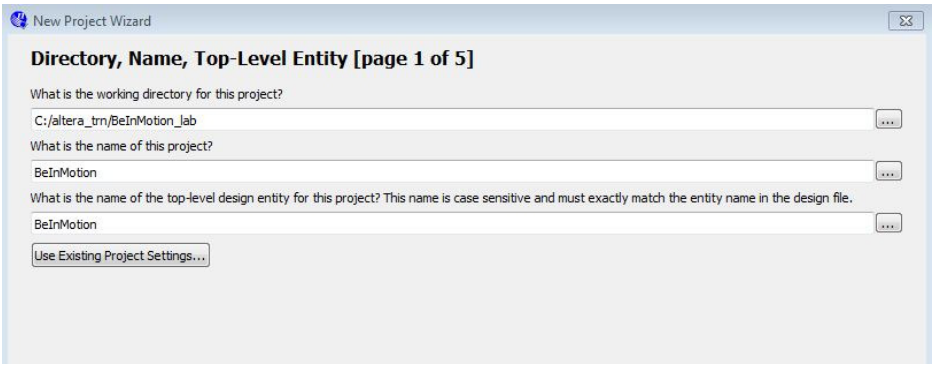
In this module, you will create a new Quartus II project that will contain the Qsys system. The top level is a pre-made Verilog file which contains some reset logic and the port connections to the Qsys system. The following steps will guide you through creating a new Quartus II project and executing a Tcl script to automate design assignment settings.

3.1 CREATE NEW QUARTUS II PROJECT

- Launch the Quartus II 11.0 software from Start -> All Programs -> Altera -> **Quartus II 11.0**.
- Click on **File -> New Project Wizard**. This will launch the New Project Wizard. An "Introduction" dialogue box may appear. If so, click Next to move to the dialogue box for the Name, Directory and Top-Level Entity.
- For the project's working directory, either type in **C:\altera_trn\BeInMotion_lab**, or click the Browse button indicated by the "**…**" symbol and navigate to the folder **BeInMotion_lab** located where BeInMotion.zip was extracted in Module 1. This will be the working directory for your project.



- Name the project **"BeInMotion"**.
  - **NOTE:** Verilog is case sensitive. Match the case exactly to match the top-level Verilog file.
- Accept the default entity name **BeInMotion**

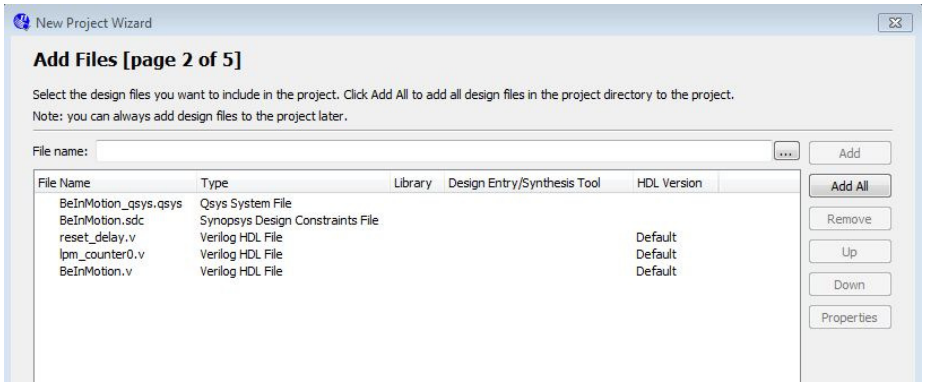


- Click **Next**.

3.2 ADD FILES TO THE PROJECT

On page 2, you will add pre-made source files to the new project.

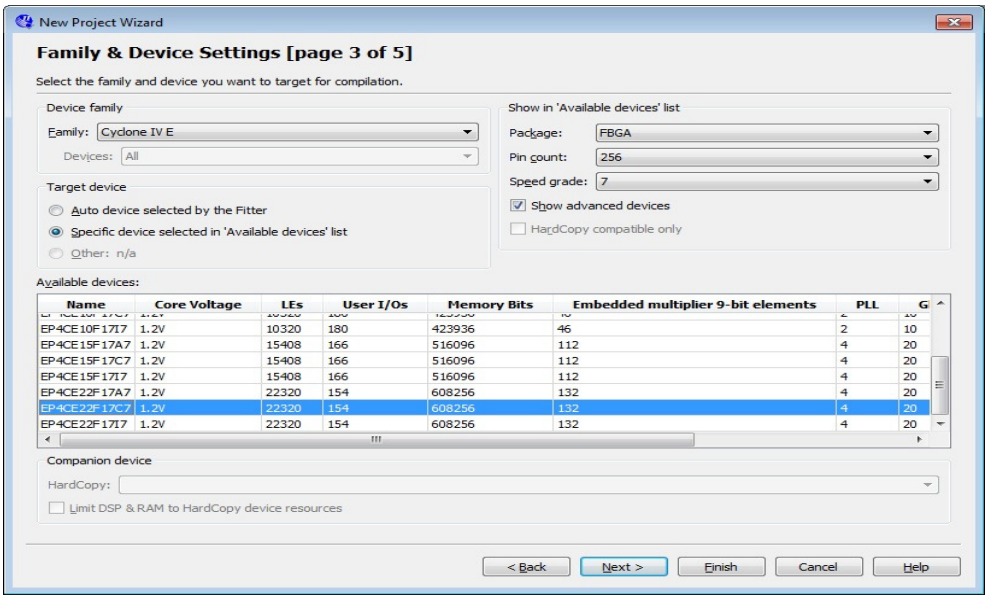
- Click the **Add All** button to include all design files located in the project folder.



- Click **Next**

3.3 SPECIFY FAMILY AND DEVICE SETTINGS

- Select Cyclone® IV **EP4CE22F17C7** device, which is the device mounted on the BeMicro SDK circuit board.
- Select **Cyclone IV E** from the Family pull-down list.
- Use the "Show in 'Available devices' list" option to filter the list of available devices to make selection easier. Select **FBGA** for the package type, **256** for the pin count and **7** for the speed grade. This will give you a shorter list of devices to choose from.
- Select **EP4CE22F17C7** from the "Available devices" list as shown below.



- Click **Next**.

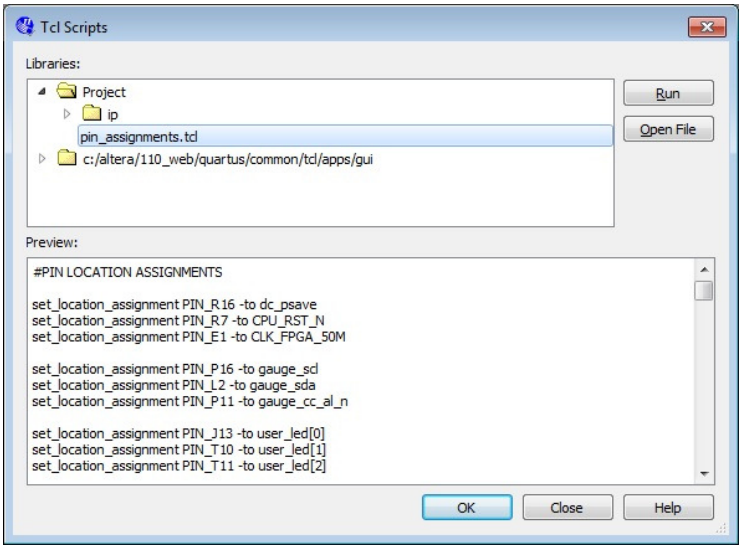
3.4 SELECT EDA TOOL SETTINGS

- Leave <None> selected for all of the options as we will not be using any third party EDA tools. Click **Next**.
- You will see a **Summary** page. Click **Finish**.

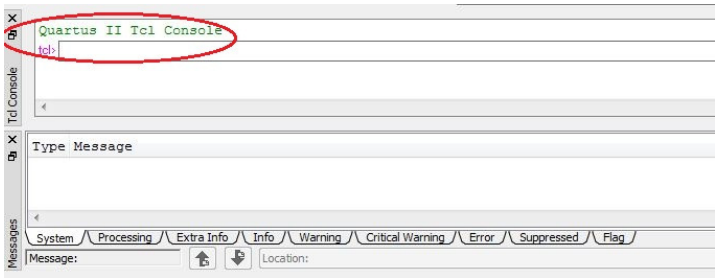
3.5 EXECUTE SETUP SCRIPT

The I/O pin constraints have been included in a Tcl script in order to set up the Quartus II project properly.

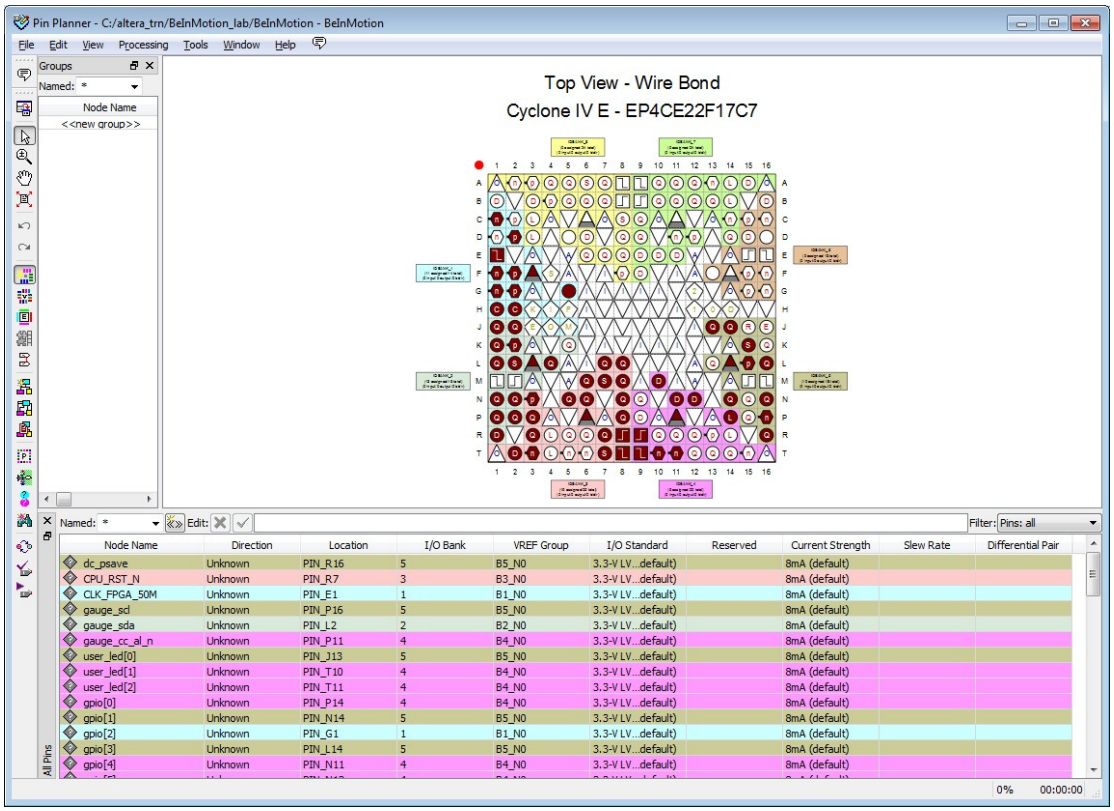
- Under the **Tools** menu, select "**Tcl Scripts…**".
- In the **Tcl Scripts** dialog box, scroll down if needed to choose the "**pin_assignments.tcl**" script.
- Click **Run**.



- The Tcl Console window will open when this script is run, it may be closed.



- Under the **Quartus Assignments** menu, select **Pin Planner.** Observe the new pin assignments that have been added to the project. For now, the direction says Unknown. These will update to the correct direction (Input, Output or Bidir) when the project is compiled later on.



- Close Pin Planner

CONGRATULATIONS!!

Your Quartus II project is set up. You are ready to start building your Qsys system.

MODULE 4: BUILD THE QSYS SYSTEM

Module Objective

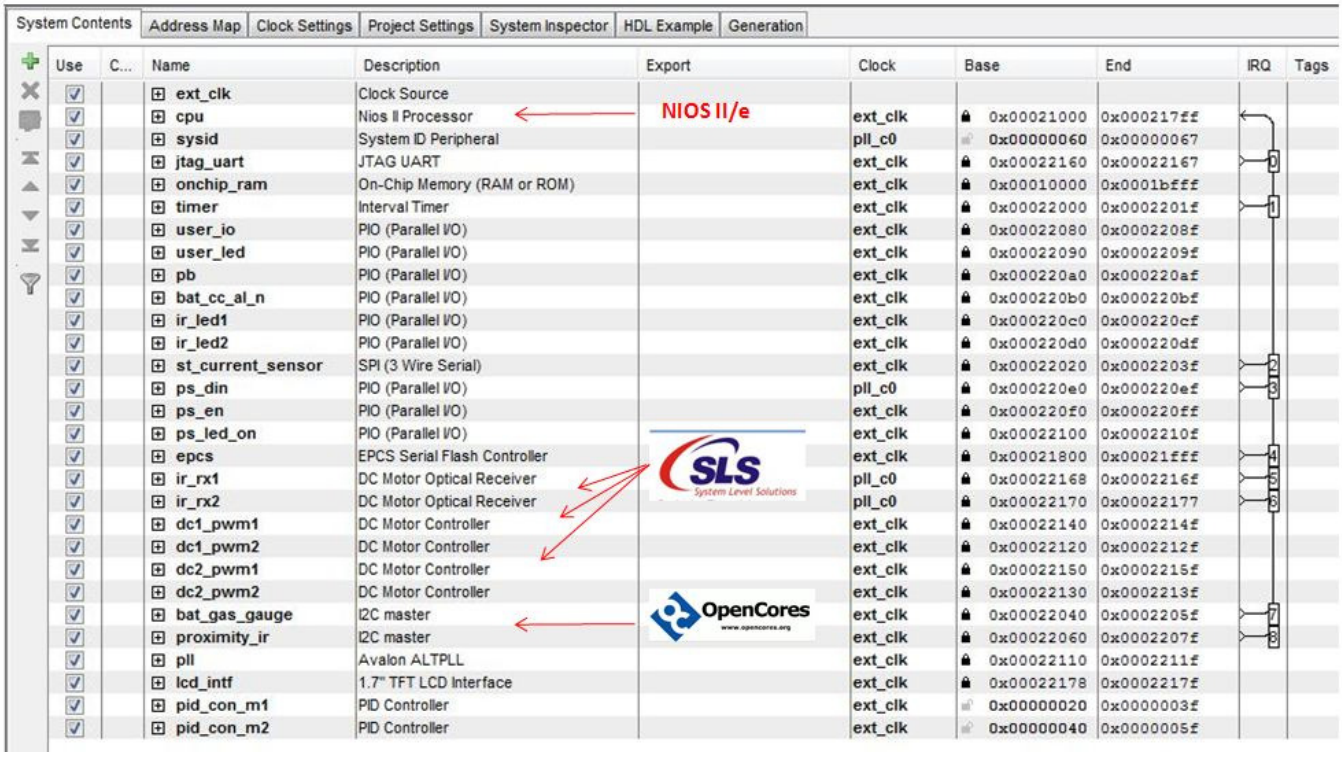
In this module, you will examine the components of the Qsys system. It is nearly complete with only the stepper motor module to be added. The following steps will guide you through creating a new Qsys component, connecting it and generating the Qsys system.

Qsys will generate RTL source code to be compiled by Quartus II, which will become the hardware platform of our motor control system. Qsys will also generate an SOPCINFO file which contains the settings needed by the Nios II EDS to generate the Board Support Package (BSP). Optionally, Qsys can create simulation files with test benches included.

4.1 LAUNCH QSYS

- From the **Tools** menu, select "**Qsys**", or click the Qsys button on the toolbar. There may be a slight delay while the Qsys application launches.

- In the "**Open**" dialog, select the system name "**BeInMotion_qsys.qsys**".
- Click **Open**.
- Examine the Qsys GUI. Each component listed in the System Contents window has a Description, Clock Source, Memory Address, and IRQ, if required. The Component Library contains all available components to add to the system.
- The components used in this system come from a variety of sources. Most are Altera components that ship with Qsys such as the Nios II/e processor, Parallel I/O controller, Interval Timer, SPI Controller and the EPCS Serial Flash Controller. SLS created the motor controllers and DC motor optical receiver. The I2C masters are from OpenCores.org. Qsys integrates these components together and generates all files required for synthesis This lab requires a stepper motor controller, which is not in the Component Library. A new component will be created and added to the library.

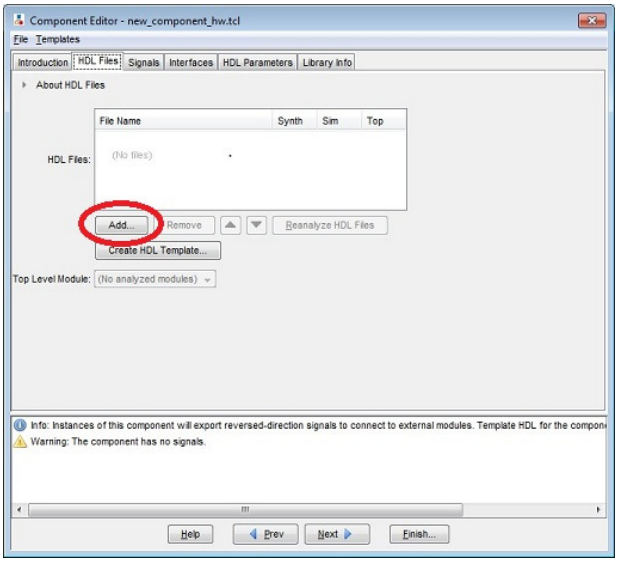


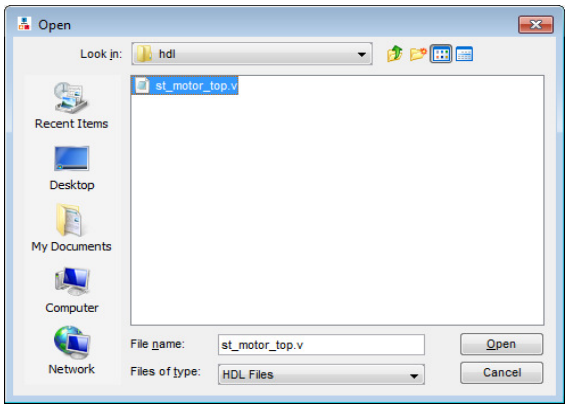
Use	C...	Name	Description	Export	Clock	Base	End	IRQ	Tags
☑		ext_clk	Clock Source						
☑		cpu	Nios II Processor		ext_clk	0x00021000	0x000217ff		
☑		sysid	System ID Peripheral		pll_c0	0x00000060	0x00000067		
☑		jtag_uart	JTAG UART		ext_clk	0x00022160	0x00022167	0	
☑		onchip_ram	On-Chip Memory (RAM or ROM)		ext_clk	0x00010000	0x0001bfff		
☑		timer	Interval Timer		ext_clk	0x00022000	0x0002201f	1	
☑		user_io	PIO (Parallel I/O)		ext_clk	0x00022080	0x0002208f		
☑		user_led	PIO (Parallel I/O)		ext_clk	0x00022090	0x0002209f		
☑		pb	PIO (Parallel I/O)		ext_clk	0x000220a0	0x000220af		
☑		bat_cc_al_n	PIO (Parallel I/O)		ext_clk	0x000220b0	0x000220bf		
☑		ir_led1	PIO (Parallel I/O)		ext_clk	0x000220c0	0x000220cf		
☑		ir_led2	PIO (Parallel I/O)		ext_clk	0x000220d0	0x000220df		
☑		st_current_sensor	SPI (3 Wire Serial)		ext_clk	0x00022020	0x0002203f	2	
☑		ps_din	PIO (Parallel I/O)		pll_c0	0x000220e0	0x000220ef	3	
☑		ps_en	PIO (Parallel I/O)		ext_clk	0x000220f0	0x000220ff		
☑		ps_led_on	PIO (Parallel I/O)		ext_clk	0x00022100	0x0002210f		
☑		epcs	EPCS Serial Flash Controller		ext_clk	0x00021800	0x00021fff	4	
☑		ir_rx1	DC Motor Optical Receiver		pll_c0	0x00022168	0x0002216f	5	
☑		ir_rx2	DC Motor Optical Receiver		pll_c0	0x00022170	0x00022177	6	
☑		dc1_pwm1	DC Motor Controller		ext_clk	0x00022140	0x0002214f		
☑		dc1_pwm2	DC Motor Controller		ext_clk	0x00022120	0x0002212f		
☑		dc2_pwm1	DC Motor Controller		ext_clk	0x00022150	0x0002215f		
☑		dc2_pwm2	DC Motor Controller		ext_clk	0x00022130	0x0002213f		
☑		bat_gas_gauge	I2C master		ext_clk	0x00022040	0x0002205f	7	
☑		proximity_ir	I2C master		ext_clk	0x00022060	0x0002207f	8	
☑		pll	Avalon ALTPLL		ext_clk	0x00022110	0x0002211f		
☑		lcd_intf	1.7" TFT LCD Interface		ext_clk	0x00022178	0x0002217f		
☑		pid_con_m1	PID Controller		ext_clk	0x00000020	0x0000003f		
☑		pid_con_m2	PID Controller		ext_clk	0x00000040	0x0000005f		

4.2 CREATE NEW COMPONENT

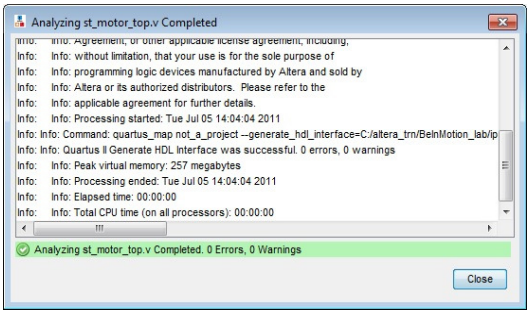
Create the stepper motor controller to add to the Qsys Component Library

- Click **File, New Component...**
- Click **Next** to advance past the New Component Introduction.
- Click **Add...** to select the source HDL



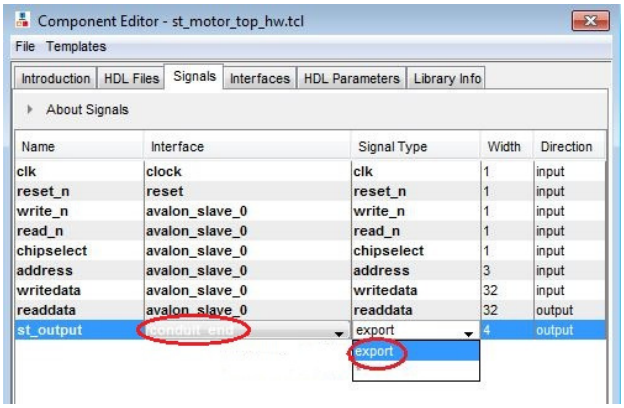


- Browse to **C:\altera_trn\BeInMotion_lab\ip\stepper_motor\hdl**
- Select **st_motor_top.v**, Click **Open**



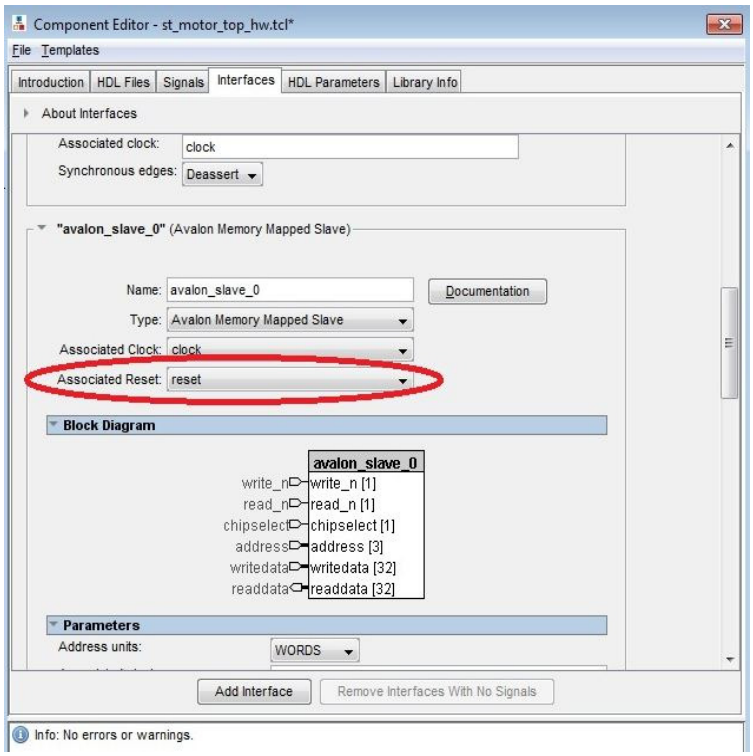
- The file is analyzed for syntax errors, click **Close**
  - **NOTE:** The two errors listed in the Messages window are corrected in the following steps.
- Click **Next**

The ports found in the HDL top level entity are listed in the Signals tab. Qsys automatically detects the signal types. However, signals that route to the system top level must be exported through a conduit as follows:



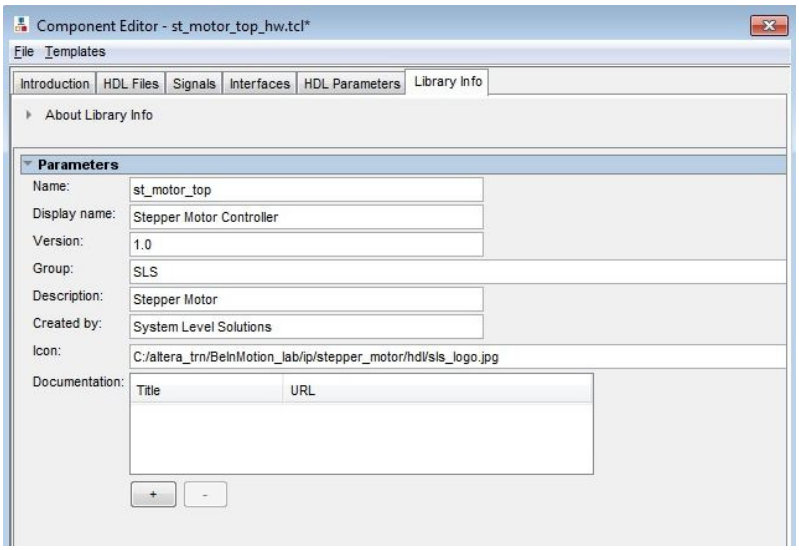
- Change the Interface for signal **st_output**. Click the Interface column, choose **new Conduit...** This will change to "conduit_end".
- Change the Signal Type for **st_output**. Click the Signal Type column, choose **export**
  - **NOTE:** Make sure you click on the word **export.** The GUI will appear to automatically select export, but it must be clicked in the pull-down box.
- Click **Next**

- The Interfaces tab lists parameters for each signal's interface used by the component. Scroll down to view the parameters for **avalon_slave_0**.
- In the **Associated Reset** pull-down box, select **reset**.



- Click **Next**
- Accept the default HDL parameters, click **Next**
- Enter the following for **Library Info:**

  Name: **st_motor_top**
  Display name: **Stepper Motor Controller**
  Version: **1.0**
  Group: **SLS**
  Description: **Stepper Motor**
  Created by: **System Level Solutions**
  Icon: Browse to **C:\altera_trn\BeInMotion_lab\ip\stepper_motor\hdl** and select **sls_logo.jpg**

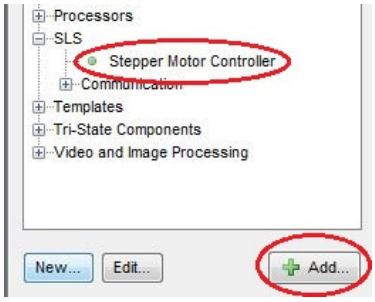


- Click **Finish**
- Click **Yes, Save**

4.3 ADD COMPONENT TO SYSTEM

The new Stepper Motor component is now contained in the library, add it to the system

- Expand **SLS** in the Component Library

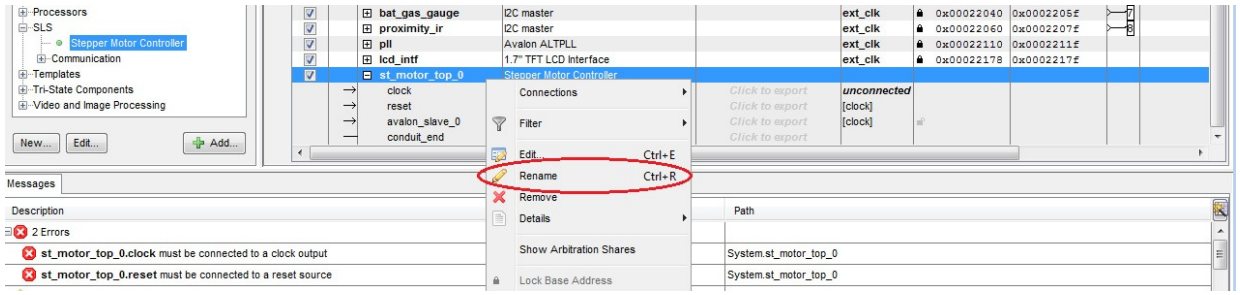


- Select **Stepper Motor Controller**
- Click **+ Add...**
- The Qsys symbol for the Stepper Motor is displayed. Click **Finish**

4.4 CONNECT COMPONENT TO SYSTEM

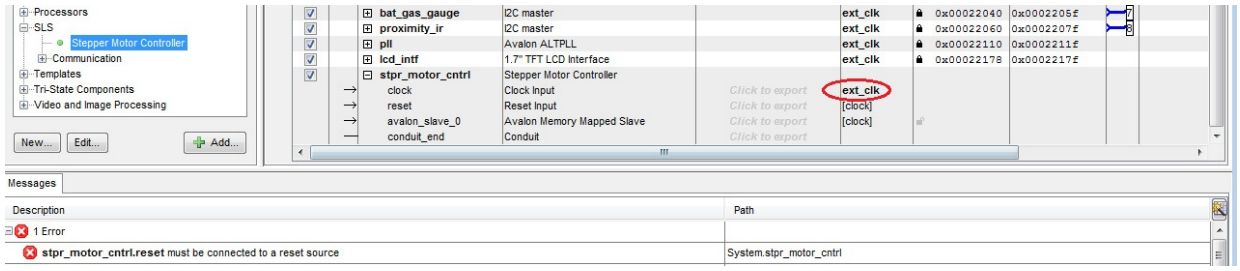
The component's interface ports must be connected to the rest of the system. Start by renaming the component to match the name used in the top-level Verilog file.

- Right-click on the new stepper motor component in the System Contents window and click **Rename**

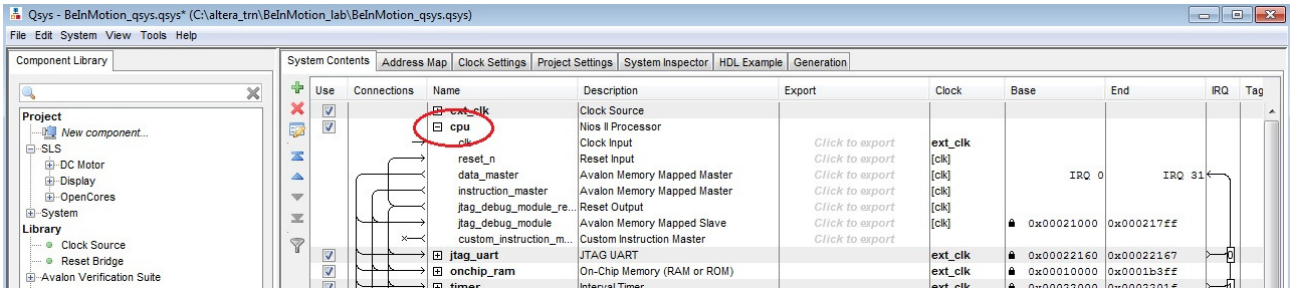


- Enter **stpr_motor_cntrl**

- Connect the external 50 MHz clock to the stepper controller. Click the Clock selection to change from "unconnected" to **ext_clk.**

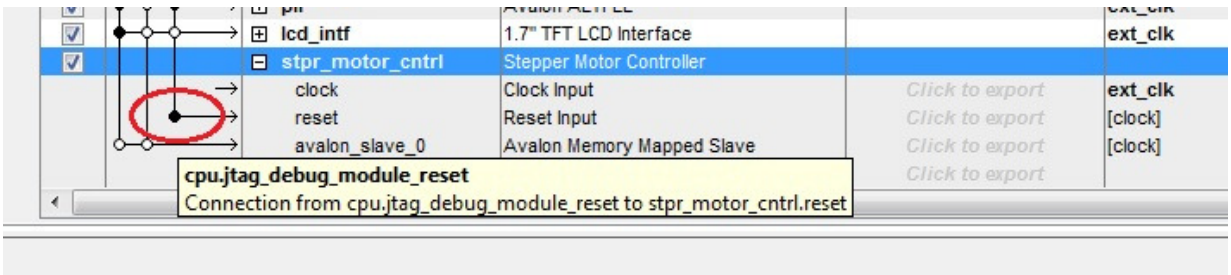


- Expand the **cpu** component to reveal the Connections panel. This will allow us to make the necessary connections between the Stepper Motor Controller and the Nios II Processor.

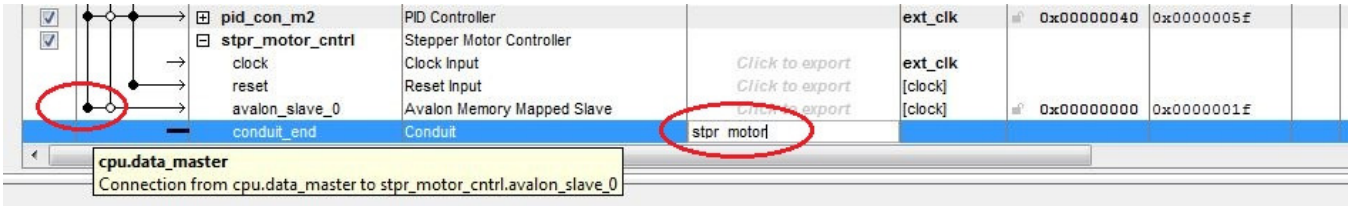


- Hold the mouse pointer over the Connection panel to view the connection options for **stpr_motor_cntrl**

- Connect the **reset** port by clicking the connection to **cpu.jtag_debug_module_reset**

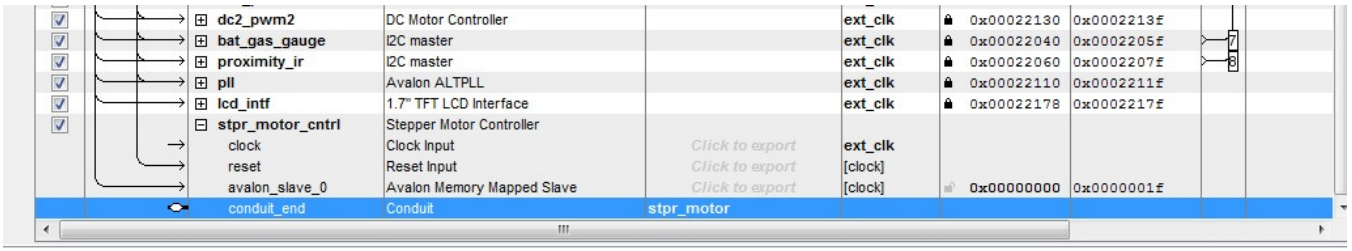


- Connect the **avalon_slave_0** port by clicking the connection to **cpu.data_master**



- Connect the **conduit_end** port to the top level by clicking in the Export column "**Click to export**"
- Enter **stpr_motor**

The Base address will remain **0x00000000** and there is no IRQ for the stepper motor controller. The completed component connections will look like this:

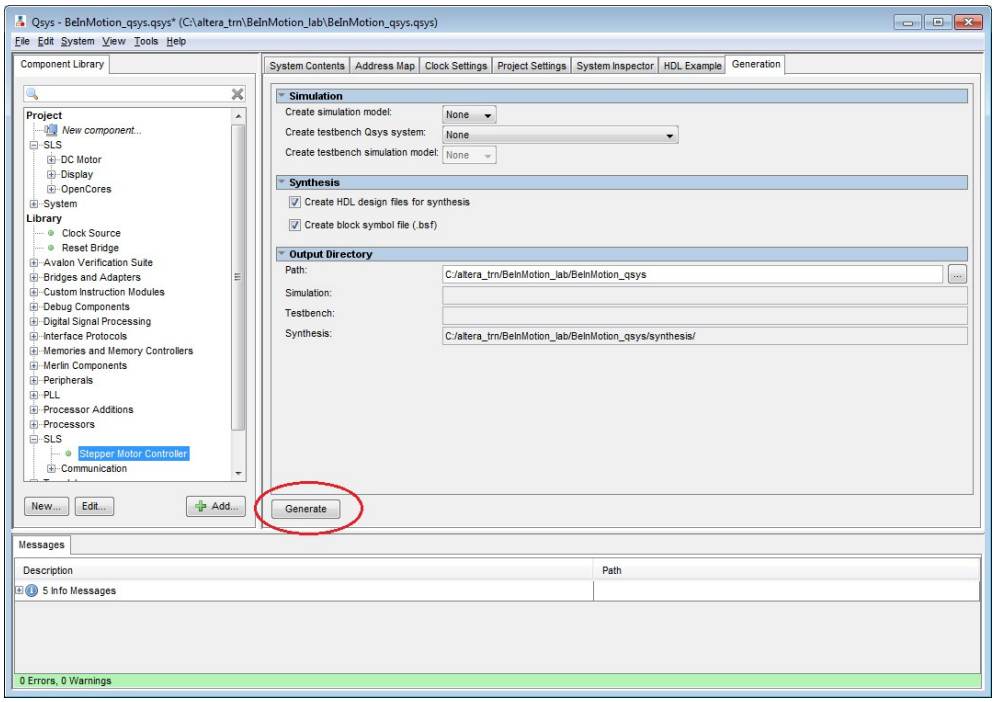


4.5 GENERATE QSYS SYSTEM

The Qsys system is ready to generate. First take a look at some of the project settings. Each of the following are tabs at the top of the screen. Click on each one to examine the settings.

Address Map: List of all components with memory address ranges for each one
Clock Settings: List of clocks available to use in the system. This includes external clocks and PLL generated clocks
Project Settings: Block diagram symbol of all interfaces to the Qsys system. Scroll down to see the project level settings, leave these at the default.
System Inspector: Provides detailed information of all connections and interfaces used in the system.
HDL Example: VHDL and Verilog HDL code examples that can be used to instantiate the Qsys system.
Generation: Includes options to create simulation models and test benches during generation.

- On the Generation page, Accept the defaults, click **Generate.**



- Click **Save**

Qsys will generate the system, allow several minutes for this to complete.

- Click **Close** Generate Complete message
- Close Qsys, click **File, Exit**

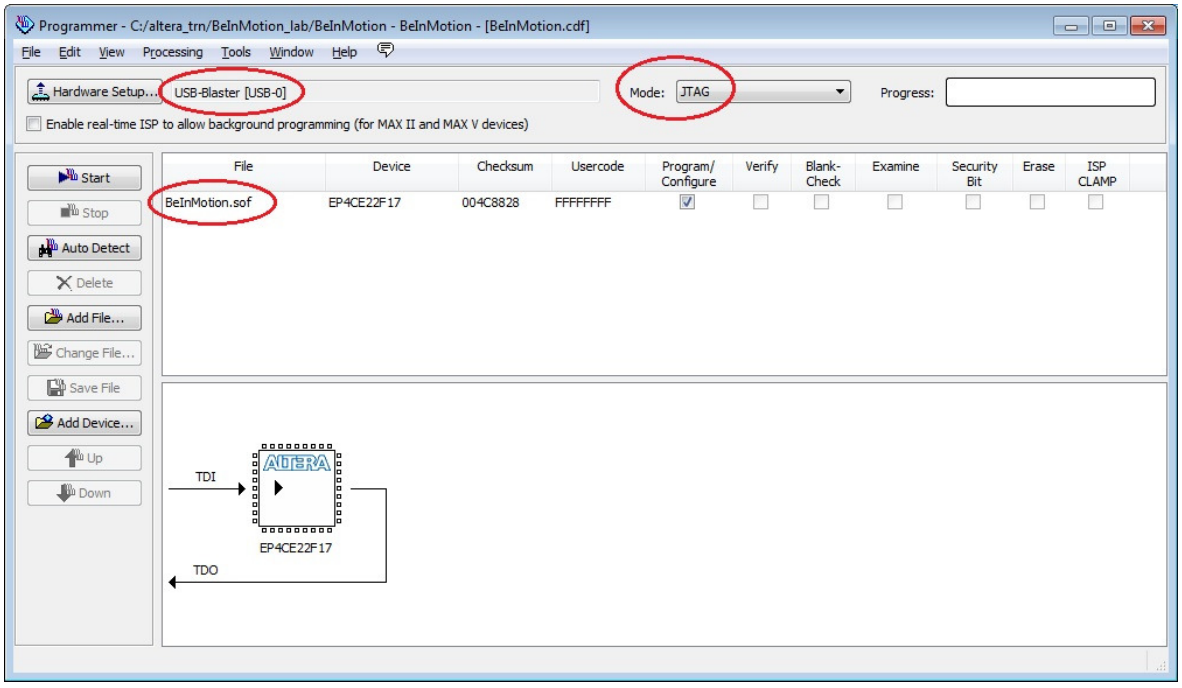
4.6 COMPILE THE QUARTUS II PROJECT

Quartus II will compile the Qsys generated code and other source files included in the design. Quartus II creates an SRAM Object File (*.SOF) which is the configuration file for the FPGA. The .SOF contains the Nios II processor, which must be in the FPGA prior to downloading software, so the Programmer is run right after compilation.

- Click the **Start Compilation** button to begin. Allow 7-12 minutes for this to complete. The initial compilation will take longer than subsequent compiles since the database and other support files are created for the first time.

- Once compilation is complete, click the **Programmer** button to load the FPGA

- Click **Hardware Setup...** to select the USB-Blaster if it is not already selected. Verify the programming mode is set to **JTAG**



- Verify that **BeInMotion.sof** is listed and the **Program/Configure box is checked**. If BeInMotion.sof is not listed as the current File, **Delete** the current file, click **Add File...,** and browse to the project folder **C:\altera_trn\BeInMotion_lab** to select **BeInMotion.sof**

- Click **Start**

- Once Progress reaches 100%, close the Programmer Window

CONGRATULATIONS!!

You have completed the Qsys system and compiled the Quartus II project. The Nios II processor has been loaded into the FPGA, so you are ready to start building your software application for motor control.

MODULE 5: IMPORT AND COMPILE THE SOFTWARE DESIGN USING NIOS II EDS

Module Objective

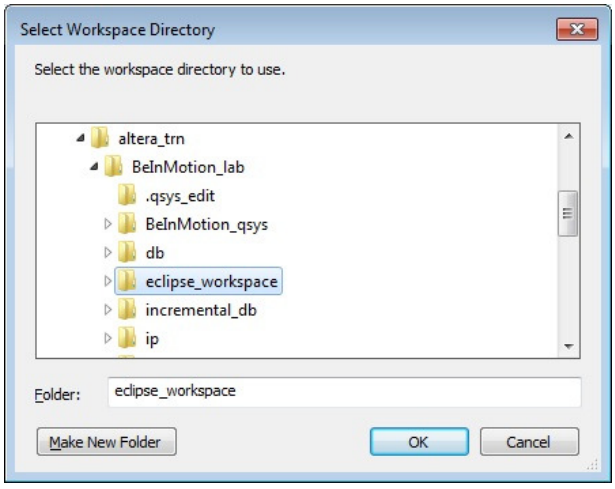
The Nios II EDS compiles the software applications that run on the Nios II processor. The Nios II EDS also creates the Board Support Package (BSP), which is the software's interface to the hardware generated by Qsys. The application in this module will compute the PID for each motor and drive each motor with PWM while taking feedback from an optical encoder. The following steps will guide you through creating a new BSP and Nios II application, downloading software to the BeInMotion and viewing PID parameter changes directly on the BeInMotion kit.

5.1 CREATE ECLIPSE WORKSPACE

The Nios II EDS requires a workspace to store project files.

Open the Nios II EDS

- Start->Programs->Altera->Nios II EDS 11.0->**Nios II Software Build Tools for Eclipse**.
  - **NOTE:** If using Windows 7, right-click on this shortcut and select **Run as administrator.**
- Select location for workspace, click **Browse...**
- Navigate to project folder **C:\altera_trn\BeInMotion_lab**

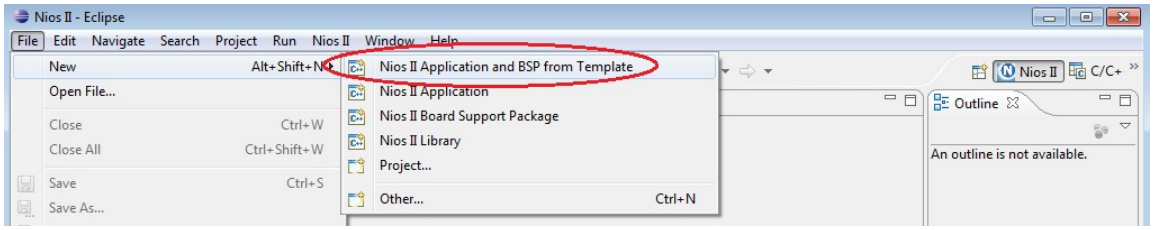


- Click **Make New Folder**
- Enter name **eclipse_workspace**
- Click **OK**
- Click **OK**

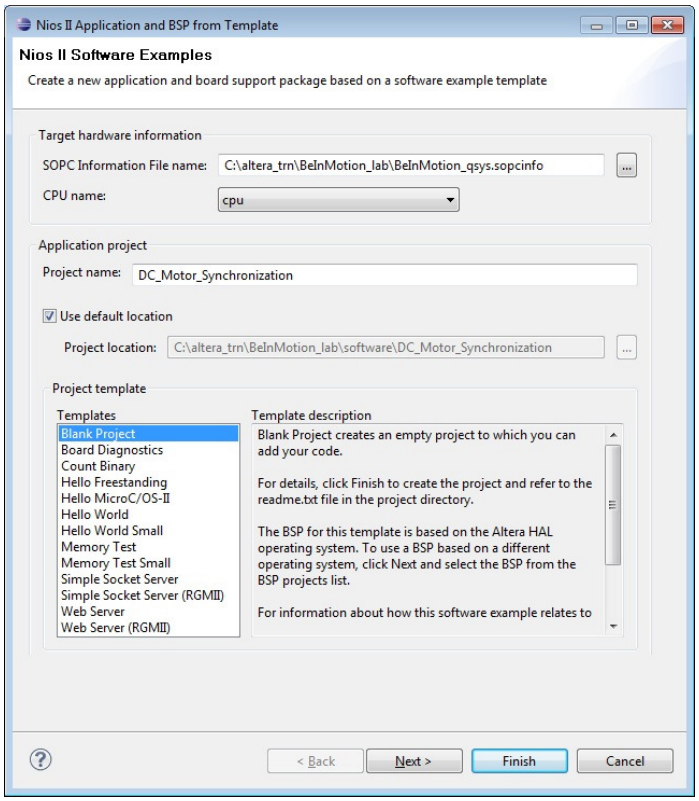
5.2 CREATE NEW NIOS II APPLICATION AND BSP

Nios II EDS opens with an empty Project Explorer. Run the wizard to create the BSP and Application folders. The BSP uses the SOPCINFO file generated by Qsys to configure the Hardware Abstraction Layer and device drivers. This lab starts with a single Application folder, more will be added later to simplify switching software applications while using a common Nios II hardware platform. Nios II EDS has Project Templates that serve as software reference designs. We will use the Blank-Project Template and import the source code for this design.

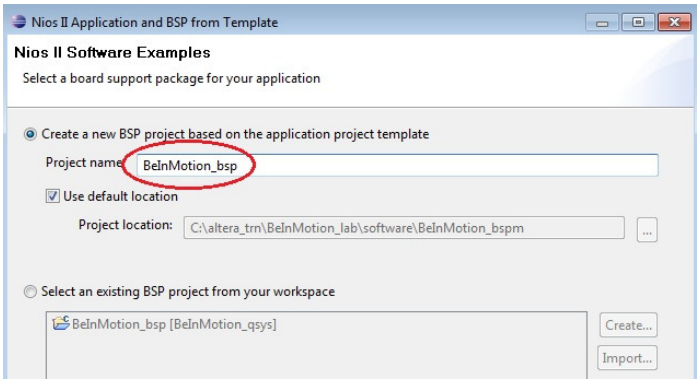
- Click File, New --> **Nios II Application and BSP from Template**



- Select SOPC Information File name: **Click Browse ...**
- Navigate to the project folder **C:\altera_trn\BeInMotion_lab** and select **BeInMotion_qsys.sopcinfo**
- Enter Application Project name: **DC_Motor_Synchronization**
- Check **Use default location**



- Select template **Blank Project**
- Click **Next**
- Select **Create a new BSP project based on the application project template**
- Delete the default BSP name and enter Project name **BeInMotion_bsp.** This BSP will be used by several applications during this workshop**.**

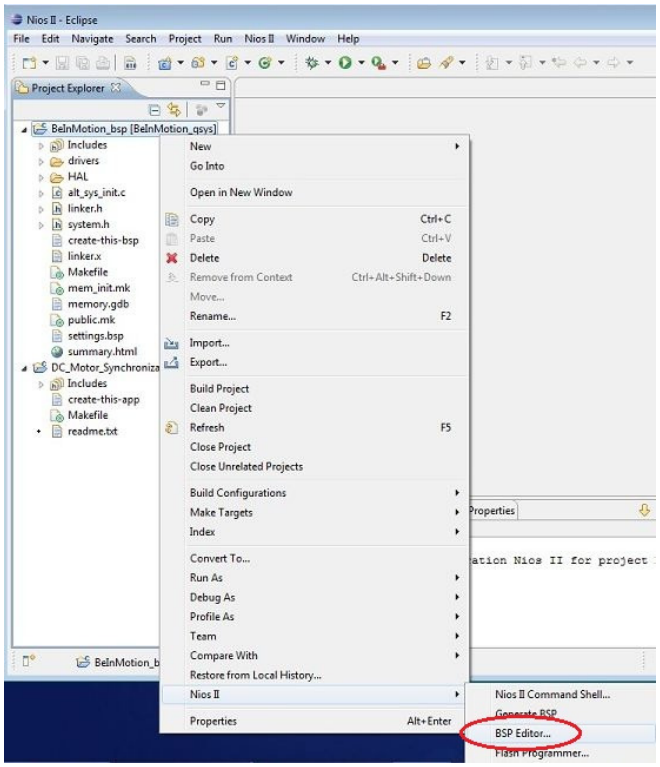


- Click **Finish** to complete the wizard

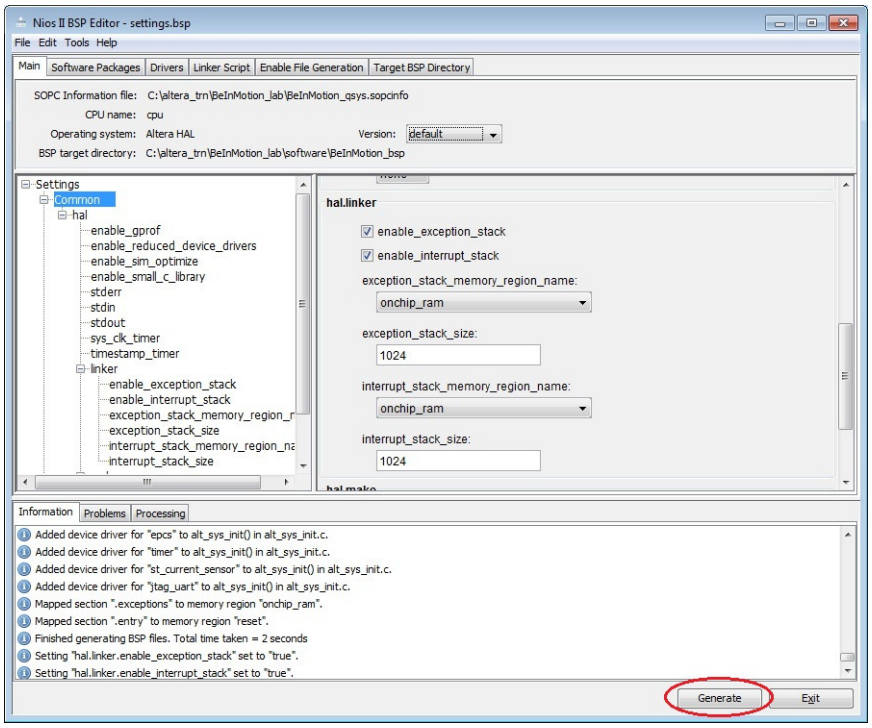
5.3 EDIT AND GENERATE BSP

Nios II EDS has created two folders, one for the BSP, and one for the application. These are located in the project folder's software directory, C:\altera_trn\BeInMotion_lab\software. The BSP folder contains system.h, Hardware Abstraction Layer (HAL) and device drivers. The following steps will guide you through editing the BSP properties and generating the BSP files.

- Right-click on the BSP folder **BeInMotion_bsp** and choose **Nios II --> BSP Editor...**



- Check the boxes for **enable_reduced_device_drivers** and **enable_small_c_library.** This will reduce the code size so it will fit in the FPGA's on-chip memory.
- Scroll down and check the boxes for **enable_exception_stack** and **enable_interrupt_stack**
- The default settings will be used for the remainder of the BSP. Click on the tabs at the top of the BSP Editor to review the settings on each tab. Here you will see settings for selecting Software Packages, Drivers, Linker Script, Enable File Generation and Target BSP Directory.

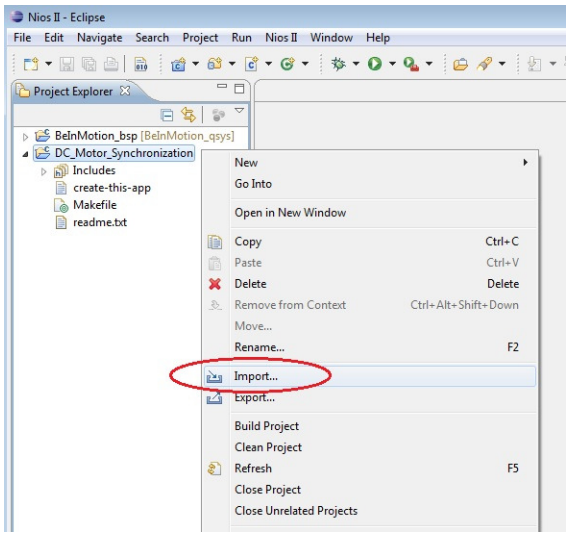


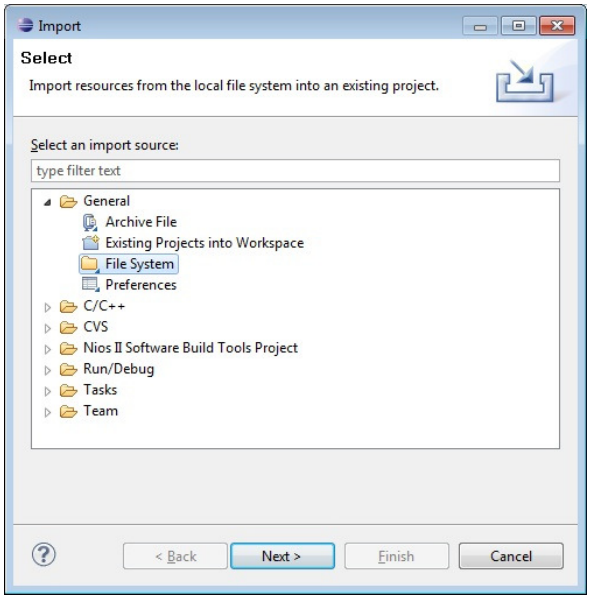
- Click **Generate**
- Click **Exit**
- Right-click on the BSP folder, **BeInMotion_bsp,** and choose **Build Project** to complete the BSP.

5.4 IMPORT SOFTWARE

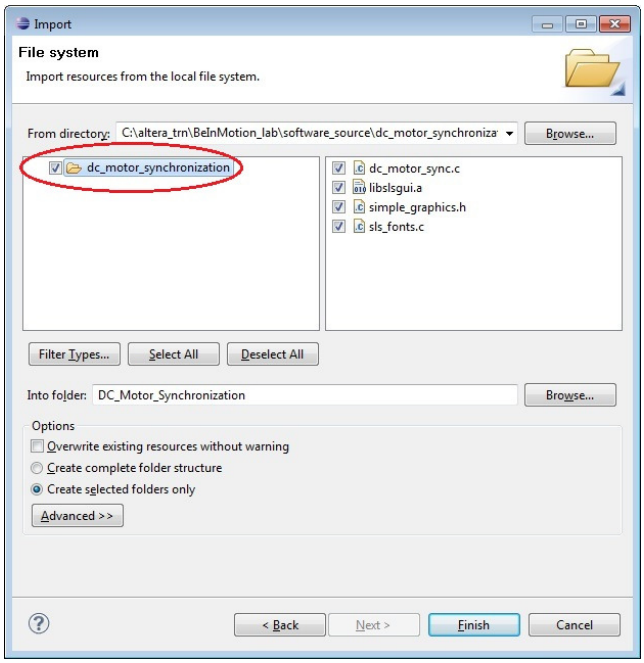
The application folder in the Nios II EDS contains the C source code. The software application will run on the target hardware previously defined in the BSP. The following steps will guide you through importing software source files and compilation.

- Right click on the Application folder, **DC_Motor_Synchronization**, choose **Import**





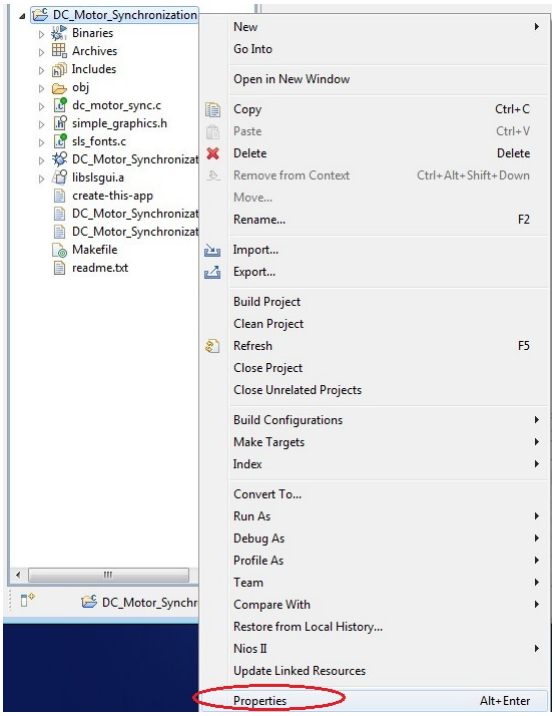
- Expand **General** and choose **File System**
- Click **Next**
- Click From directory: **Browse...**
- Navigate to **C:\altera_trn\beinmotion_lab\ import_code\dc_motor_synchronization**
- Click **OK**
- Click the check box next to the folder **dc_motor_synchronization** to select all files in the folder



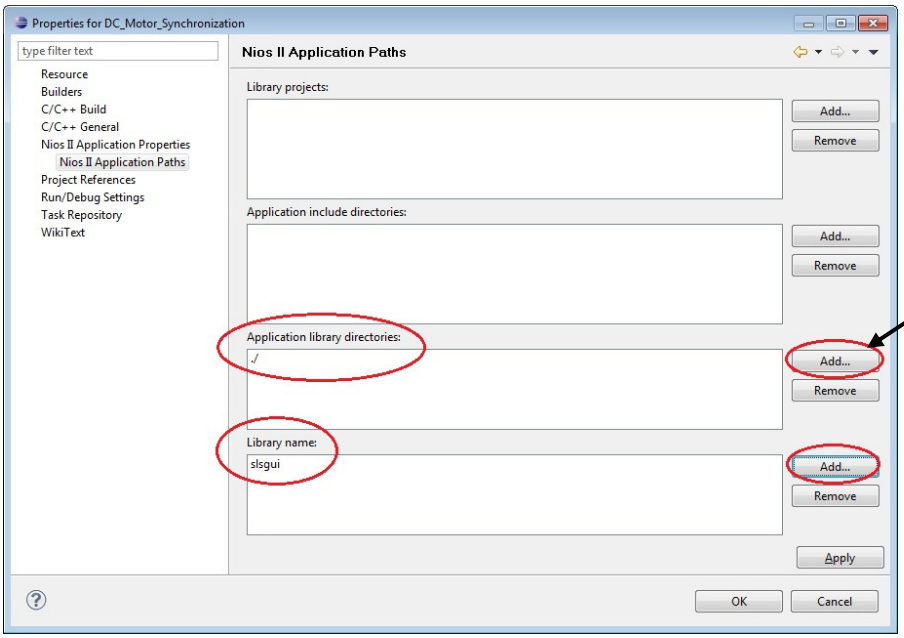
- Accept the rest of the defaults and click **Finish**

The design includes LCD support files which are in library format, set the library path in Properties

- Right-click on the Application folder, **DC_Motor_Synchronization**, and choose **Properties**



- Expand **Nios II Application Properties** and choose **Nios II Application Paths**



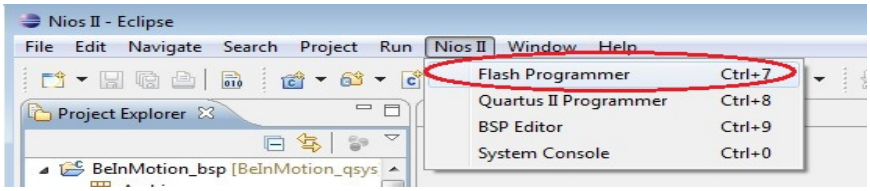
- Click **Add...** next to **Application library directories:**

- Select the **dc_motor_synchronization** application directory located at **C:\altera_trn\beinmotion_lab\software\dc_motor_synchronization** and click **OK**
- Click **Yes** to convert the path to a relative path
- Click **Add...** next to **Library name:**
- Enter **slsgui**
- Click **OK**
- The application is ready to be compiled. Right-click on the Application folder, **DC_Motor_Synchronization**, and choose **Build Project**.

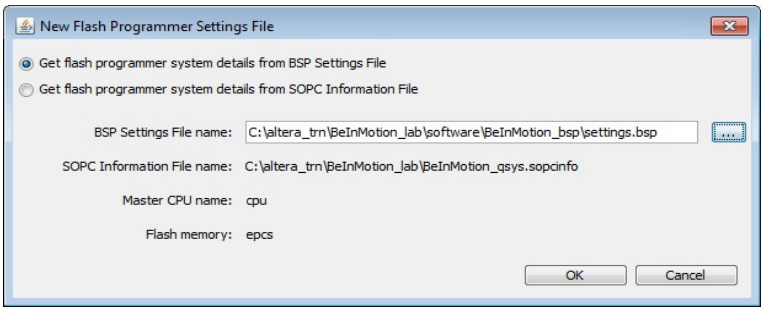
5.5 LOAD DESIGN INTO TOOLSTICK USING THE NIOS II FLASH PROGRAMMER

Now that the software application is compiled, it can be downloaded directly to the FPGA via JTAG, or stored in serial flash memory located on the BeMicro SDK tool stick. The Nios II Flash Programmer can program to memory Nios II software executable files, FPGA configuration code and other data files such as coefficient tables or sine lookup tables. The following steps will guide you through storing the Quartus II *.SOF and the Nios II executable *.ELF to flash using the Nios II Flash Programmer.

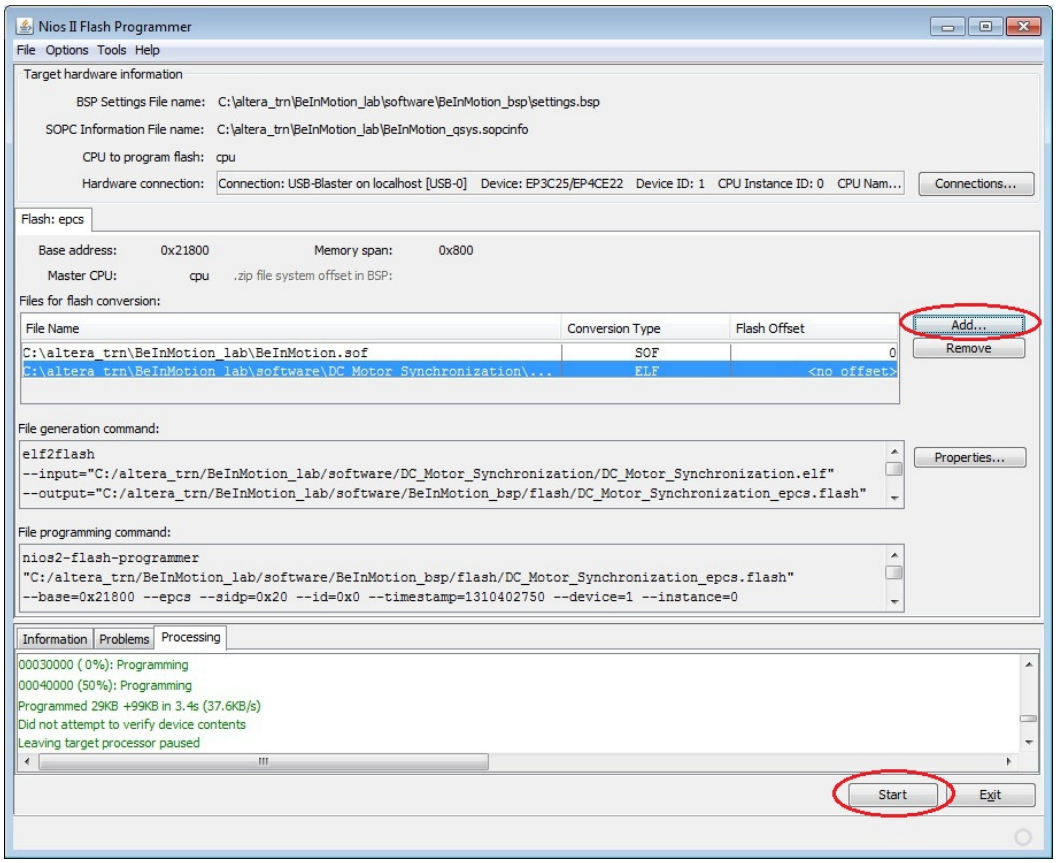
- Select **Nios II** from the menu and choose **Flash Programmer**



- Click **File, New** to create a New Flash Programmer Settings File
- Click **Get flash programmer system details from BSP Settings File**
- Click **Browse ...** to locate the BSP Settings File name
- Navigate to the BSP location **C:\altera_trn\BeInMotion_lab\software\BeInMotion_bsp** and select **settings.bsp**



- Click **OK**
- Click **Add...** to include the hardware image file
- Select Files of type**: Quartus II SOF File**
- Navigate to the project folder **C:\altera_trn\BeInMotion_lab** and select B**eInMotion.sof**



- Click **Add...** again, this time to include the software ELF
- Select Files of type: **Nios II ELF File**
- Navigate to the project application folder **C:\altera_trn\BeInMotion_lab\software\DC_Motor_Synchronization** and select **DC_Motor_Synchronization.elf**
- Click **Start**
- The flash is programmed when the message "Leaving target processor paused" appears. Click **Exit**

CONGRATULATIONS!!

You have created and compiled the software application to control the DC motors in the BeInMotion kit. The flash memory is programmed, it's time to drive the kit!

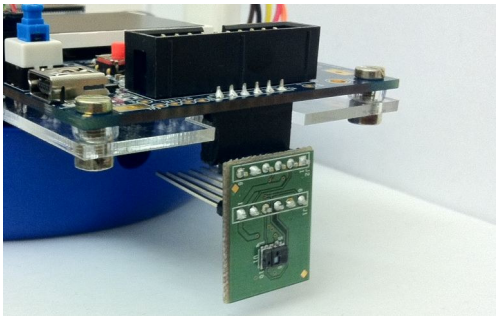
MODULE 6: DC MOTOR SYNCHRONIZATION

Module Objective

The BeInMotion kit is ready to go! In this module, we will take the kit for a drive, and see the effects of modifying the PID motor control parameters.

6.1 CONNECT THE PROXIMITY SENSOR BOARD

- Disconnect the BeInMotion USB cable and turn the power switch to the OFF position.
- Connect the proximity sensor board to the front of the BeInMotion motor base for either wall or table edge detection. The BeInMotion in Wall Detect mode will drive up to a wall, back up, turn, and drive until another wall is detected. The BeInMotion in Table Edge mode will drive up to a table edge, back up, turn, and drive until another table edge is detected.



Wall Detect Position

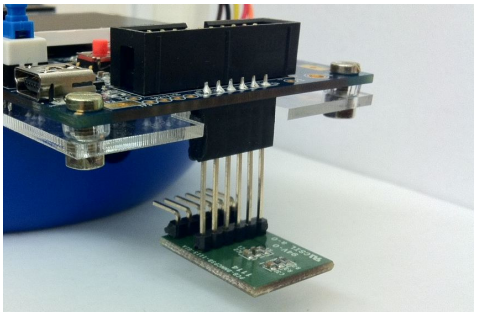
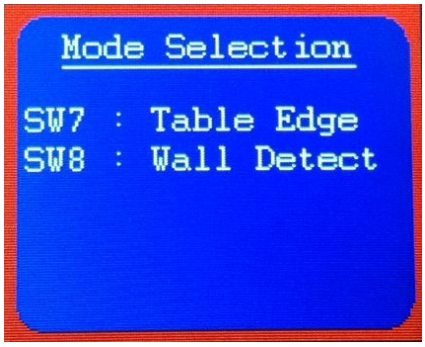


Table Edge Detect Position

6.2 DRIVE THE BEINMOTION

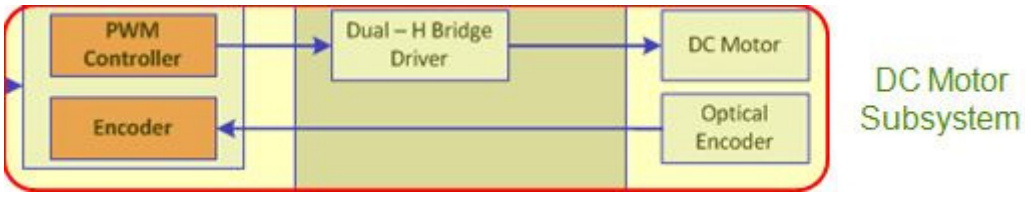
- Power on the BeInMotion kit, you will be presented with the Mode Selection menu on the LCD



- Press either SW7 or SW8 corresponding to the position of the proximity sensor board and watch it go!
- Turn off the BeInMotion kit to reset or change modes.

6.3 CHANGE MOTOR PARAMETERS

Recall the DC Motor Subsystem from Module 2:



In this closed loop control system, the encoder count values are fed into the PID algorithm which detects any differences between the two motors. Based on the error detected, the PWM duty cycle is changed to each motor accordingly to keep the motors synchronous.

The three PID terms are Proportional Term (Present error), Integral Term (Accumulation of Past Errors) and Derivative Term (Prediction of future errors). The initial values have been chosen to properly tune the motors and keep them synchronous. If the constant parameters of these terms are chosen incorrectly, the system can become unstable.

- Turn off the BeInMotion kit and reconnect the kit to the PC USB cable
- Open **dc_motor_sync.c** located in the **DC_Motor_Synchronization** Application folder
- Make the following modifications to PID parameters for DC motor 1:

  #define dc_m1_Kp 12
  #define dc_m1_Ki 4
  #define dc_m1_Kd 4

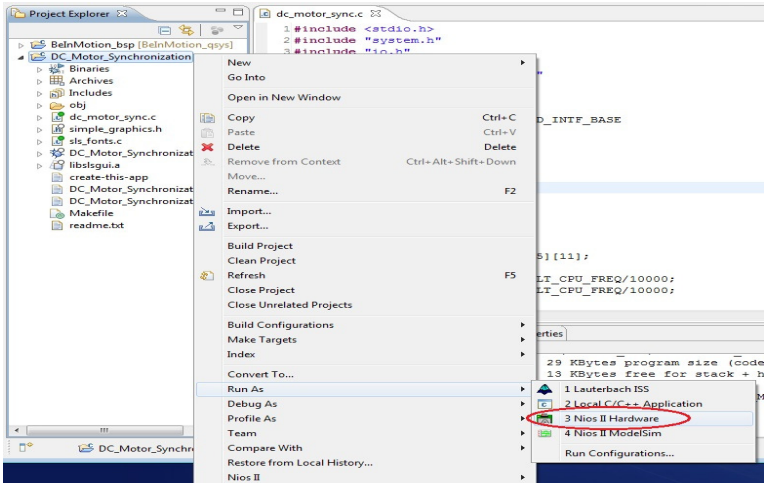
- Click **Save**

dc_motor_sync.c

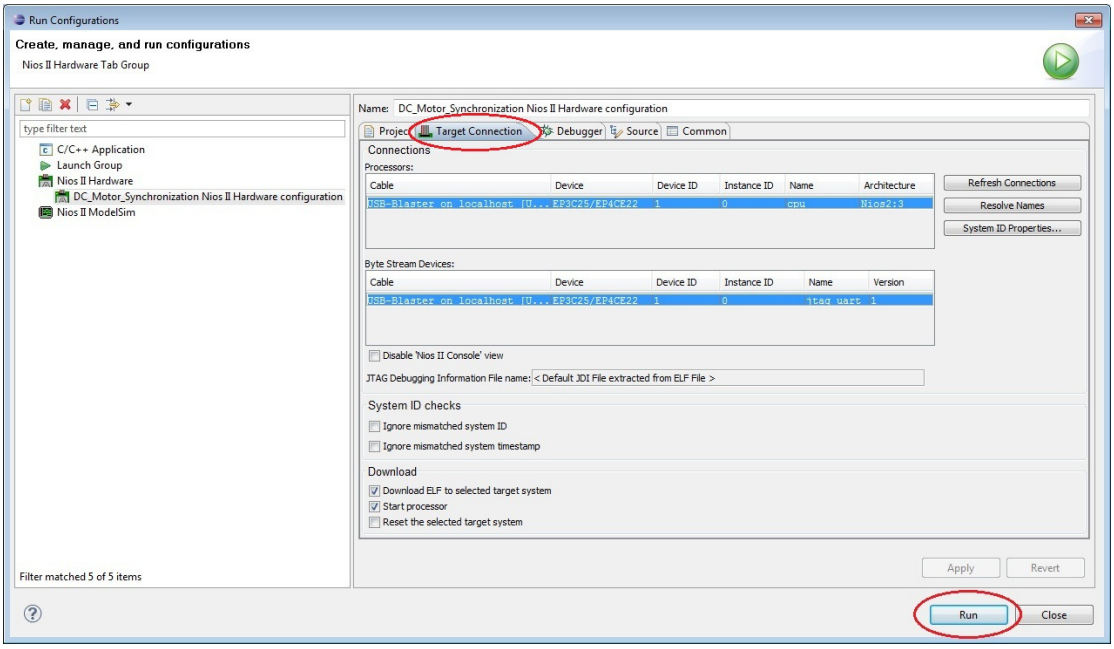
```
#define LCD_INTERFACE_0_BASE LCD_INTF_BASE

#define TABLE_EDGE  1
#define WALL_DETECT 0
#define OutMax 0xFFFF
#define OutMin 0x0000
#define DC_M1_COUNT_REF 100
#define DC_M2_COUNT_REF 100

#define PID_MODE_SEL 0    // "1" for HW  "0" for SW

//dc motor 1 PID constant parameters
#define dc_m1_Kp 12
#define dc_m1_Ki 4
#define dc_m1_Kd 4

//dc motor 2 PID constant parameters
#define dc_m2_Kp 6
#define dc_m2_Ki 0.5
#define dc_m2_Kd 0

extern char cour10_font_array[95][11];
unsigned int  edge;
unsigned int  dc_m1,dc_m2;
unsigned int  dc_m1_pwm_freq = ALT_CPU_FREQ/10000;
unsigned int  dc_m2_pwm_freq = ALT_CPU_FREQ/10000;
unsigned int  dc1_cmp,dc2_cmp;
unsigned char table_edge,wall_detect;
```

- Right click on the Application folder, **DC_Motor_Synchronization**, and choose **Build Project**.

- Right Click on the Application folder, **DC_Motor_Synchronization,** and choose **Run As -> Nios II Hardware**



➢ **NOTE:** The Nios II Run Configurations window will appear the first time Nios II- Run As Hardware is launched. Subsequent runs will download directly without opening this window. Changes to the Run Configurations can be made in Run -> Run Configurations.

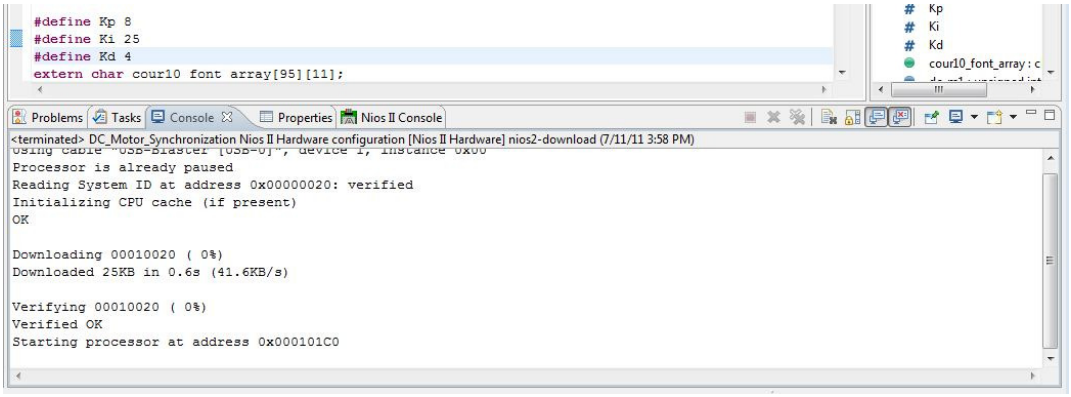


- Click the **Target Connection** tab to view the detected Processors and Byte Stream Devices.

- Click **Run**

  - **NOTE:** If the Run button is grayed out, click the **Refresh Connections** button.

Nios II Run As Hardware will download the new software image directly to the FPGA without overwriting the flash contents. This is a quick way to test out new code. Cycling power to the FPGA will restore the original code in flash memory. The system is ready when the Console reports "Starting processor"



- Push the power switch to the **ON** position.
- Disconnect the USB cable, the kit will switch to battery power and stay on.
- Press SW7 or SW8 corresponding to the position of the proximity sensor board.

Notice that DC motor 1 is no longer stable since the PID calculation is no longer tuned to the motor. Reconnect the kit and experiment with different PID values.

CONGRATULATIONS!!

You have run the DC motor control applications on the BeInMotion kit and modified the PID parameters controlling the motor synchronization.

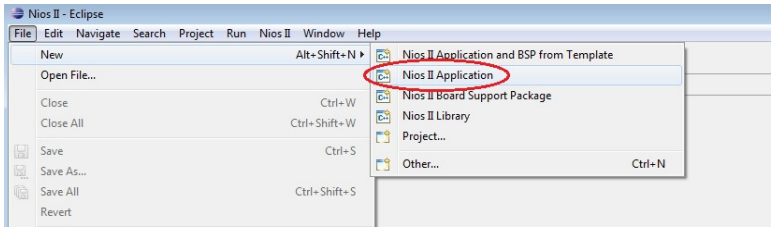
MODULE 7: STEPPER MOTOR

Module Objective:

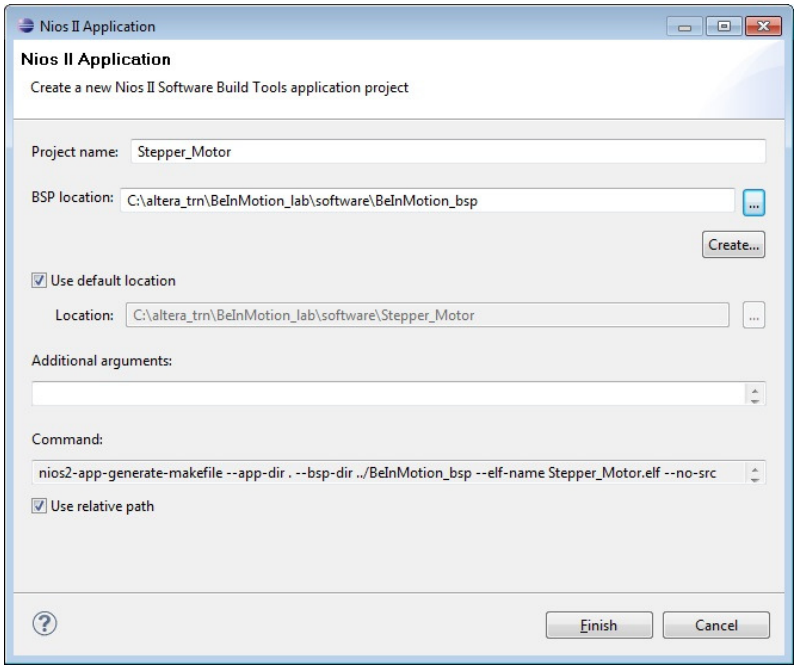
Stepper Motor drivers often have different modes of operation. These modes determine the sequence the coils are energized to make the motor shaft move appropriately. This module offers a selection between the four stepping modes, Wave Mode, Full Step, Half Step and Micro-Stepping, both clockwise and counter-clockwise. The following steps will guide you through creating a new Nios II application and modifying the source code to switch between the modes on the BeInMotion kit.

7.1 CREATE NEW NIOS II APPLICATION

- Click **File, New --> Nios II Application**

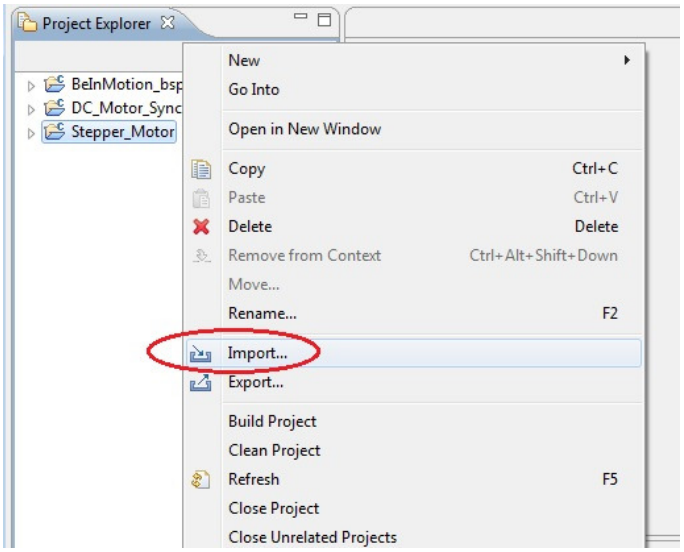


- Enter Project Name**: Stepper_Motor**
- BSP location: click **Browse ...**
- Select the BSP, **BeInMotion_bsp**
- Accept the defaults for the remaining options, Click **Finish**

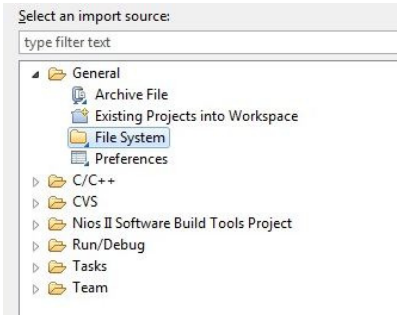


Nios II EDS has generated a new Application folder, Stepper_Motor. Import source code into the folder.

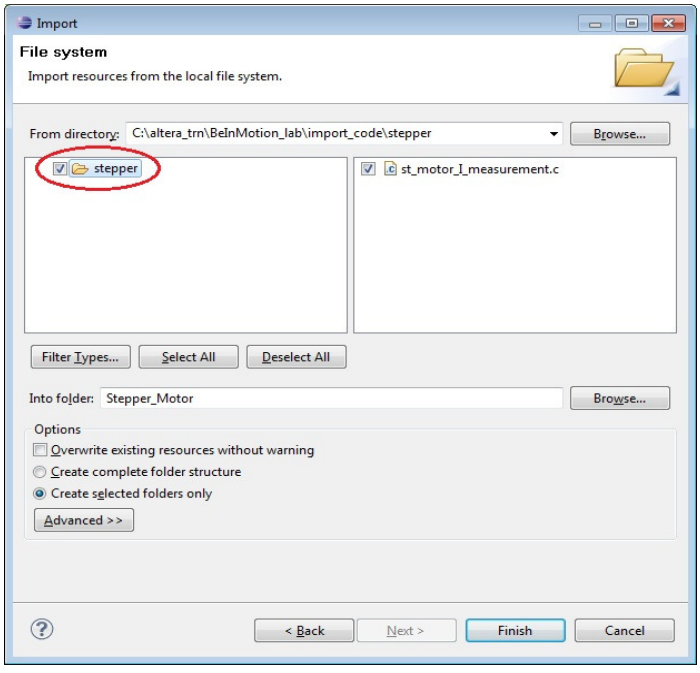
- Right click on **Stepper_Motor**, choose **Import**



- Expand **General** and choose **File System**
- Click **Next**



- Click **From directory: Browse...**
- Navigate to **C:\altera_trn\beinmotion_lab\ import_code\stepper**
- Click **OK**
- Click the check box next to the **Stepper** folder
- Accept the rest of the defaults and click **Finish**



7.2 STEPPER MODE PARAMETERS

- The Stepper_Motor application defines the following stepper modes for clockwise (forwards) and counter-clockwise (backwards) in **st_motor_I_measurement.c**

#define WAVE_STEP_FWD
#define WAVE_STEP_BWD
#define HALF_STEP_FWD
#define HALF_STEP_BWD
#define FULL_STEP_FWD
#define FULL_STEP_BWD
#define MICRO_STEP_FWD
#define MICRO_STEP_BWD

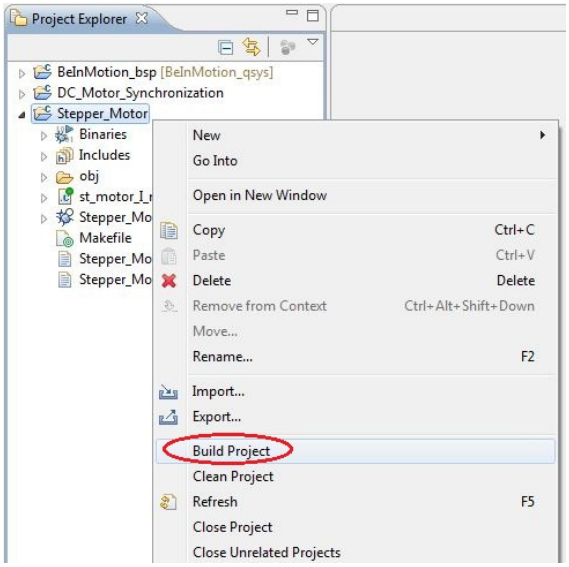
st_motor_I_measurement.c

```
#include "stdio.h"
#include "system.h"
#include "unistd.h"


#define WAVE_STEP_FWD  0x00
#define WAVE_STEP_BWD  0x01
#define HALF_STEP_FWD  0x02
#define HALF_STEP_BWD  0x03
#define FULL_STEP_FWD  0x04
#define FULL_STEP_BWD  0x05
#define MICRO_STEP_FWD 0x06
#define MICRO_STEP_BWD 0x07



#define CPU_FREQ               50000000
#define ST_MOTOR_FREQ_HZ       350
#define STEP_TO_ROTATE         20000
#define ST_MOTOR_MODE          FULL_STEP_FWD
#define ST_MOTOR_CONTOL_REG    (((CPU_FREQ/ST_MOTOR_FREQ_HZ) << 4 ) | ST_MOTOR_MODE)
```

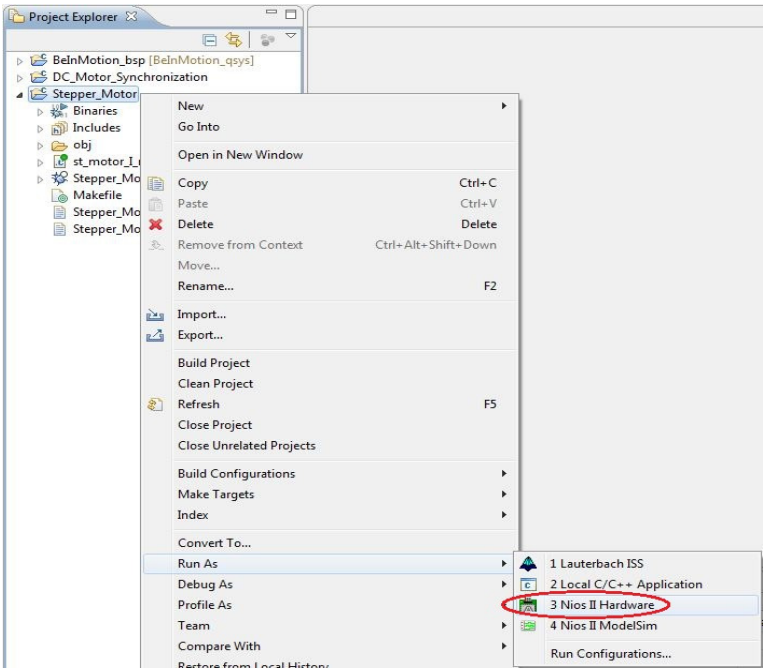
- Open **st_motor_I_measurement.c** located in the **Stepper_Motor** Application folder. Observe the stepper mode is set to FULL_STEP_FWD.

7.3 RUN THE STEPPER CONTROLLER ON THE BEINMOTION KIT

- Connect the BeInMotion kit to the PC USB cable
- Compile the application. Right-click on the Application folder, **Stepper_Motor,** click **Build Project.**

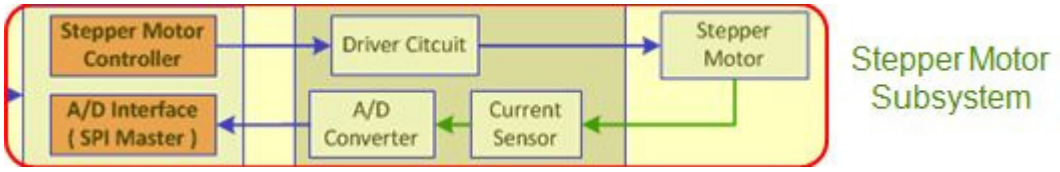


- Right-click on the Application folder **Stepper_Motor,** click **Run As -> Nios II Hardware**
- If the Run Configurations window appears, click **Run**

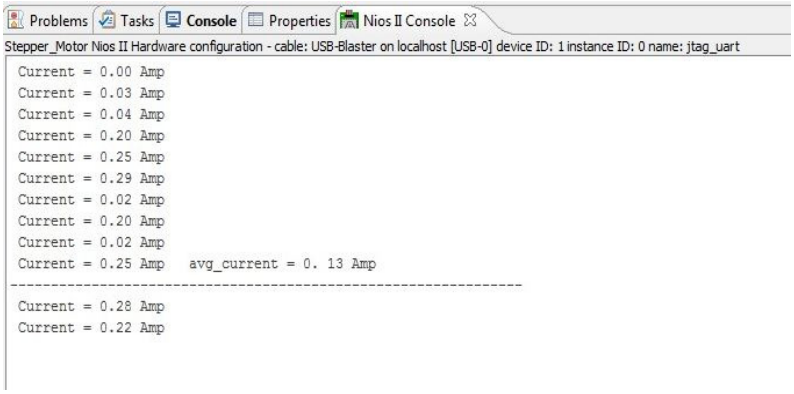


Once the software has loaded, the stepper motor will complete a few rotations using the Full Step stepper mode.

- Recall the Stepper subsystem from Module 2.



- The feedback from the current sensor is displayed in the Nios II Console



7.4 CHANGE STEPPER MODES

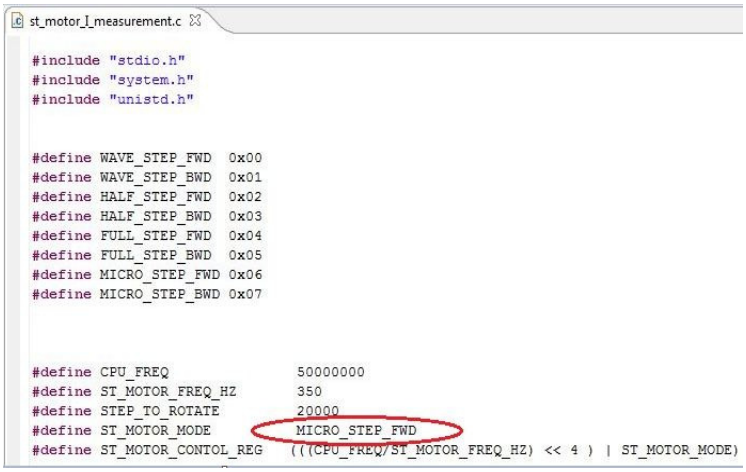
In the Wave Drive mode only a single phase is activated at a time. It has the same number of steps as the Full Step mode, but the motor will have significantly less torque.

Full Step is the usual method for driving a stepper motor. Two phases are always on and the motor will have full rated torque.

When Half stepping, the drive alternates between two phases on and a single phase on.

Micro Stepping is smoother compared to Full and Half Step Mode. The controller has to apply input pulses such that current flowing through the winding follows a sine/cosine pattern. The controller uses PWM to generate the sine waves.

- Change the Stepper Mode to **MICRO_STEP_FWD**
- Click **Save.**



- Rebuild the project. Right-click **Stepper_Motor**, select **Build Project**
- Reload the software on the BeInMotion kit. Right-click on **Stepper_Motor**, select **Run As -> Nios II Hardware**
- Notice the change in speed of the motor and the current sensor feedback reported in the Nios II console during operation. Full Step Mode will have the greatest current consumption, Micro Step Mode will have the least.

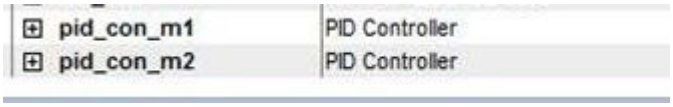
CONGRATULATIONS!!

You have created the stepper motor controller application and run it on the BeInMotion kit! The current consumption is reported in the Nios II Console during stepper mode operation.

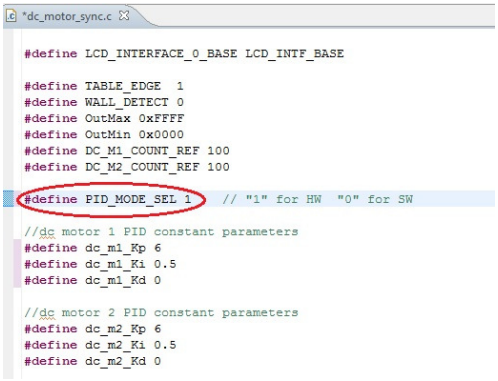
APPENDIX A: HARDWARE PID AND SOFTWARE PID

Control loops can be accelerated by implementing them in hardware rather than using the processor to calculate the algorithm. A good example is implementing the PID calculation in hardware rather than software. An example has been provided in this lab. There are two PID controllers supplied with this lab, a hardware PID written in Verilog and a software PID written in C.

The hardware PID controllers are implemented as Qsys components.



To switch to hardware calculated PID, change **#define PID_MODE_SEL** to 1 in **dc_motor_sync.c**



Rebuild the DC_Motor_Synchronization application and Run As->Nios II Hardware.

NOTE: This hardware PID implementation was not designed to use floating point multiplication or addition. Make sure only integers are used when selecting the PID parameters for hardware PID mode.

Altera offers these resources if you would like to learn more about floating point capabilities:

Taking Advantage of Advances in FPGA Floating-Point IP Cores
http://www.altera.com/literature/wp/wp-01116-floating-point.pdf

Floating Point Megafunctions User Guide
http://www.altera.com/literature/ug/ug_altfp_mfug.pdf

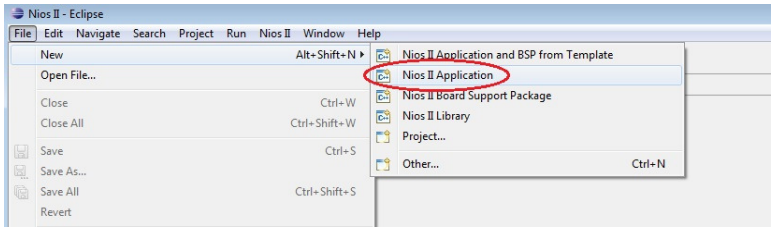
APPENDIX B: BATTERY GAS GAUGE

Objective:

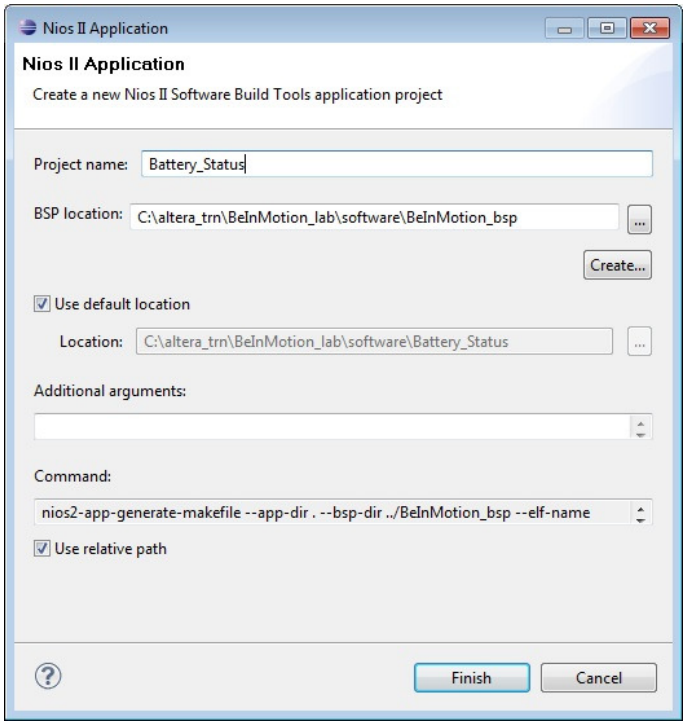
The BeInMotion kit includes the LTC2942-1 I2C Battery Gas Gauge. This software application implements an algorithm to translate Coulomb measurements resulting in a very accurate indication of battery charge and discharge. The following steps will guide you through creating this application and running it on the BeInMotion kit.

B.1 CREATE NEW NIOS II APPLICATION

- Click **File, New --> Nios II Application**

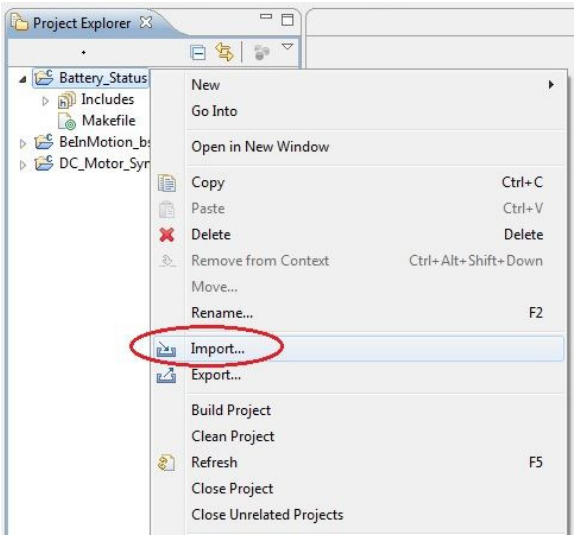


- Enter Project Name: **Battery_Status**
- BSP location: click **Browse ...**
- Select the BSP **BeInMotion_bsp**
- Accept the defaults for the remaining options, Click **Finish**

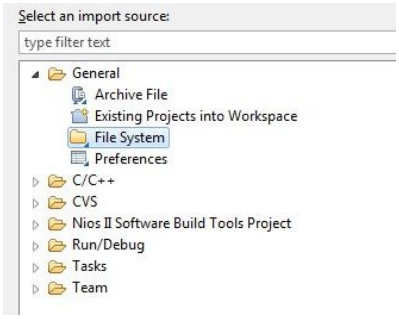


Nios II EDS has generated a new Application folder, Battery_Status. Import source code into the folder.

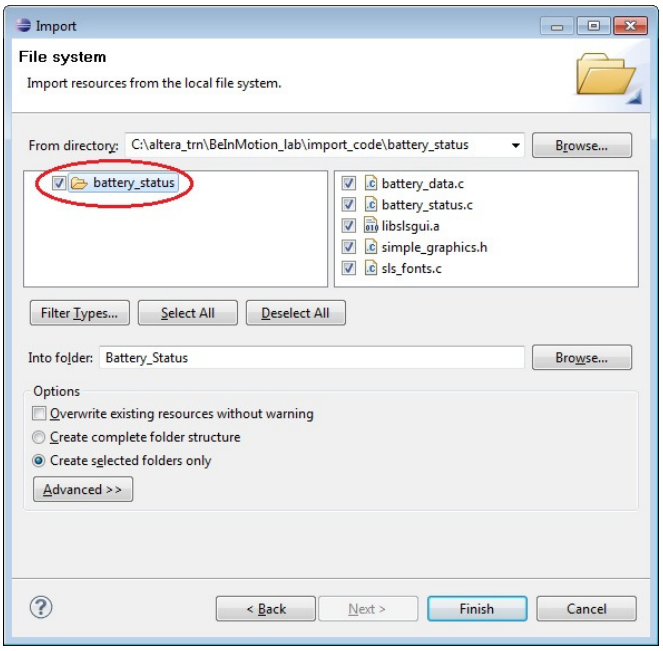
- Right click on **Battery_Status**, choose **Import**



- Expand **General** and choose **File System**
- Click **Next**

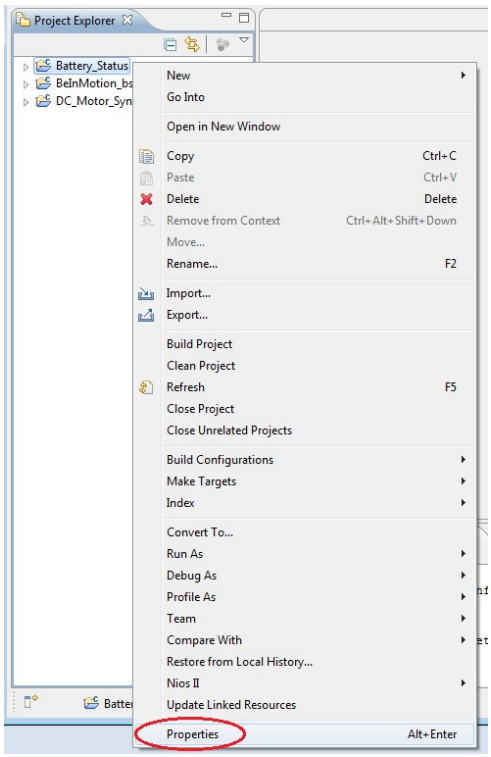


- Click **From directory: Browse...**
- Navigate to **C:\altera_trn\beinmotion_lab\import_code\battery_status**
- Click **OK**
- Click the check box next to the **battery_status** folder
- Accept the rest of the defaults and click **Finish**

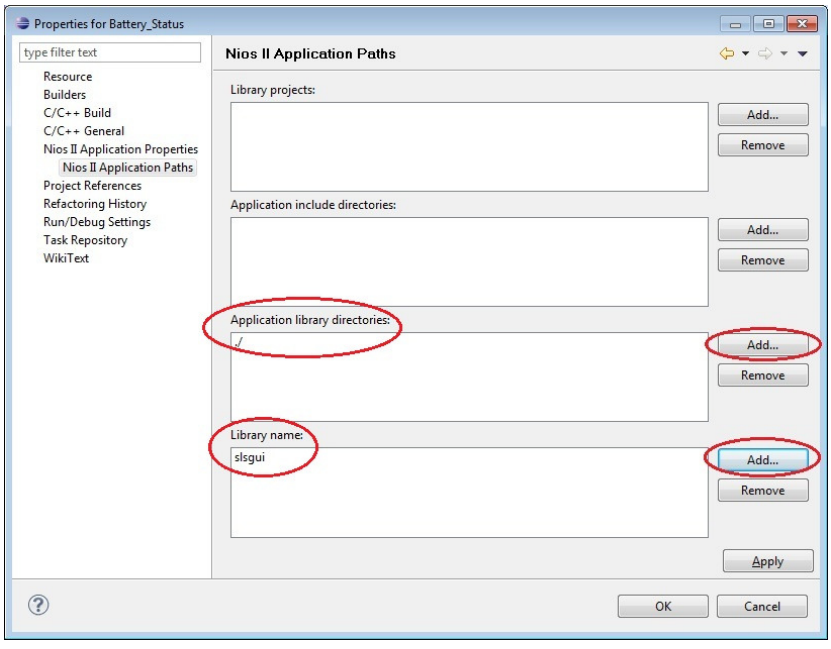


The design includes LCD support files which are in library format, set the library path in Properties

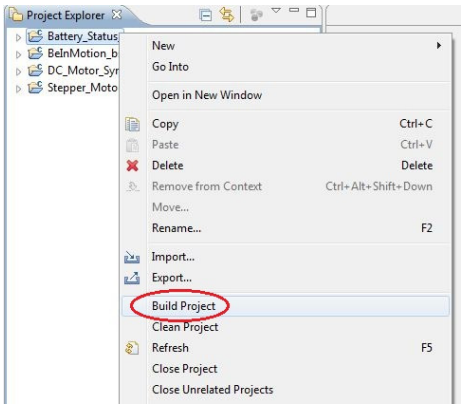
- Right-click on the Application folder, **Battery_Status**, and choose **Properties**



- Expand **Nios II Application Properties** and choose **Nios II Application Paths**
- Click **Add...** next to **Application library directories:**
- Select the **battery_status** application directory located at **C:\altera_trn\beinmotion_lab\software\battery_status**
- Click **Yes** to convert the path to a relative path
- Click **Add...** next to **Library name:**
- Enter **slsgui**
- Click **OK**



- The application is ready to be compiled. Right-click on the Application folder, **Battery_Status**, and choose **Build Project**.



B.2 RUN THE BATTERY STATUS APPLICATION ON THE BEINMOTION KIT

- Once the build is complete, Right click on the Application folder **Battery Status**. Click **Run As -> Nios II Hardware**

- If the Run Configurations window appears, click **Run**

The reset options for the battery gas gauge are displayed upon startup. Resetting the battery gas gauge with the minimum or maximum coulomb count values will provide the most accurate status within 1%. The battery gas gauge will continue monitoring the battery status even when the kit is powered off, or a new software application is loaded.

- **SW5** writes a coulomb count value to the battery gas gauge according to the current voltage level. This method is 90-95% accurate. Press and hold **SW5** until the "Reset Complete!" message appears.
  - **NOTE:** Disconnect the mini-USB and BeMicroSDK USB cables before pressing SW5 to ensure an accurate voltage reading. SW5 needs to be pressed only the first time the application is run, or if the battery has been disconnected. The charge status will report "FULL" until this is done.
- **SW7** writes the minimum coulomb count value to the battery gas gauge. When the battery is fully discharged and the kit will not power on, connect the mini-USB charge cable and power on the kit. Press and hold **SW7** until the "Reset Complete!" message appears.
- **SW8** writes the maximum coulomb count value to the battery gas gauge. When the battery has completed charging (green LED is off), press and hold **SW8** until the "Reset Complete!" message appears.

APPENDIX C: TAKING THE NEXT STEP

Altera has a number of resources available to assist you further in product development

Some of the resources available are:

Get more information about Industrial Motor Control

http://www.altera.com/end-markets/industrial/motor-control/ind-motor-control.html

Get more information about the Nios II Processor

Nios II Processor Reference Handbook
http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf

Get more information about the Nios II Software Development Tools

Nios II Software Developer's Handbook
http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf

Get more information about Embedded System Design

Embedded Design Handbook
http://www.altera.com/literature/hb/nios2/edh_ed_handbook.pdf

Get more information about the Qsys System Integration Tool

Qsys Documentation , Tutorials and Training
http://www.altera.com/support/software/system/qsys/sof-qsys-index.html

Embedded Peripherals Handbook
http://www.altera.com/literature/hb/nios2/edh_ed_handbook.pdf

Get free online training or take an in-person training course

http://www.altera.com/education/edu-index.html

APPENDIX D: TROUBLESHOOTING

1. **LCD and LEDs flicker when the kit is powered on.**

   The battery does not have sufficient charge to power on the kit due to the amount of current required by the stepper motor. Fully charge the battery or connect either USB cable and power on the kit.

2. **Files are not added to the Quartus project when the Add All button is clicked.**

   Make sure the Quartus working directory matches the location of the source files. For example, if BeInMotion.zip is extracted to C:\altera_trn, the working directory will be C:\altera_trn\BeInMotion_lab

3. **Quartus error "Error: Port "stpr_motor_export" does not exist in macrofunction "b2v_inst"**

   When creating the stepper motor component in Qsys, the signal type for the conduit port must be set to Export. Edit the Stepper Motor Controller component, click on the pull-down selection for Signal Type and choose Export. Regenerate the Qsys system before recompiling the Quartus project.

4. **No USB Blaster found in the Quartus Programmer Hardware Setup, accompanied by error "Attempted to access JTAG server --internal error code 82 occurred"**

   The JTAG server service is not running due to a firewall or anti-virus program running. See http://www.altera.com/support/kdb/solutions/rd04042011_1000.html If the Altera JTAG Server is not listed in Services, disable the anti-virus program and install the Quartus II Stand-Alone Programmer https://www.altera.com/download/software/prog-software to install the JTAG server service.

5. **Application folders do not appear when creating a new Nios II application.**

   The Nios II Software Build Tools must be run "As Administrator" on PCs with Windows 7

6. **The Run Configurations window does not show any target connections.**

   Click Refresh Connections to refresh the JTAG connection and detect the Nios II processor

7. **The Run Configurations window reports "System timestamp mismatch"**

   The hardware image running on the BeInMotion does not match the BSP. Load the newest hardware image (BeInMotion.SOF) using the Quartus programmer, then re-run the application.

8. **Problem "The system console model is invalid. Cannot launch"**

   Click OK and re-run Run As->Nios II Hardware. This usually occurs when the Nios II Console is still running from a previous launch. To prevent, click the red box to "Terminate and Remove Launch" on the Nios II Console prior to launching Run As-> Nios II Hardware.

APPENDIX E: RESTORING THE FACTORY IMAGE

The BeMicro SDK - BeInMotion kit has been flashed with an FPGA hardware image and a software image.

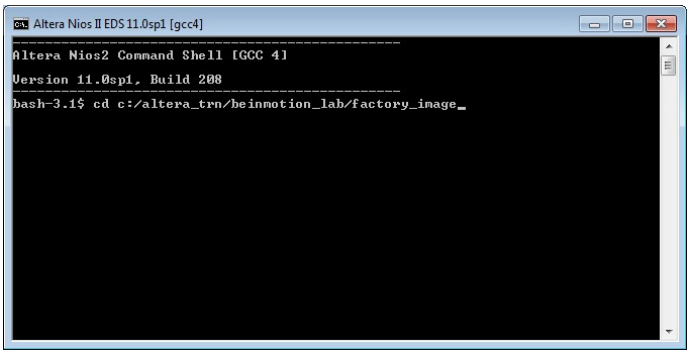
During development new images can be flashed to the board. Follow these steps to restore the factory image.

Open a Nios II Command Shell

- Start->All Programs->Altera->Nios II EDS 11.0->**Nios II 11.0 Command Shell**

Change directory to the Factory_Image directory

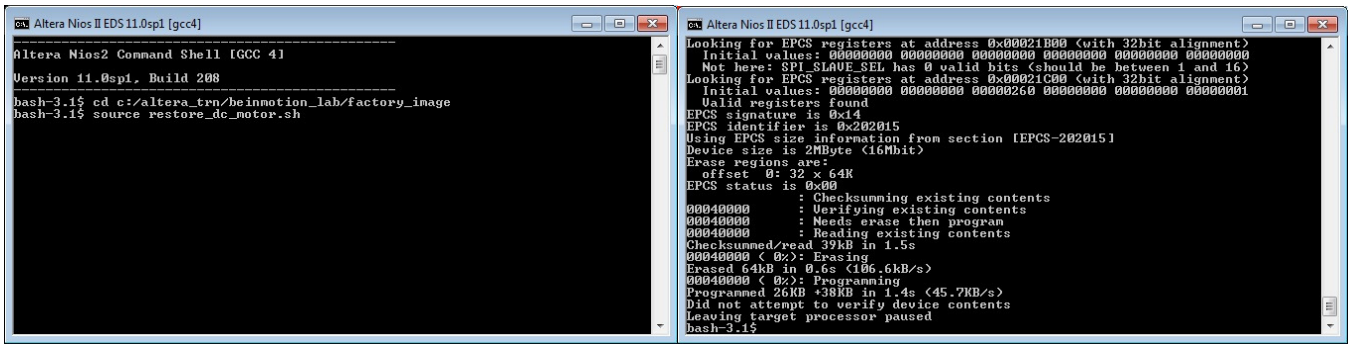
- **cd c:/altera_trn/beinmotion_lab/factory_image**



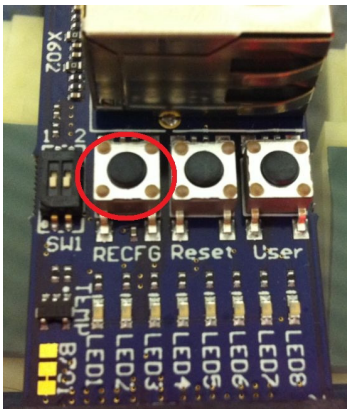
Connect the BeMicro SDK-BeInMotion kit to a USB port on the PC

Run the Script and wait for it to complete

To restore DC_Motor_Synchronization, type	**source restore_dc_motor.sh**
To restore Stepper_Motor, type	**source restore_stepper_motor.sh**
To restore Battery_Status, type	**source restore_battery_status.sh**



Force the hardware to reconfigure with the new image



- Press the **Reconfigure Button** on the BeMicro SDK