**Freescale Semiconductor** Document Number: WCT100XAWCAUG
User's Guide Rev. 3.5, 12/2015

# WCT1001A/WCT1003A Automotive A13 Wireless Charging Application User's Guide

## 1. Key Features

The Automotive A13_Rev3 (A13_Rev3_SCH-28216_A1, A13_Rev3_LAY-28216_A) wireless charging demo is used to transfer power wirelessly to a charged device. A charged device can be any electronic device equipped with a dedicated wireless charging receiver.

The main parameters of the wireless charging transmitter (WCT) are as follows:

- The input voltage ranges from 9 V DC to 16 V DC (automotive range).
- The input voltage can drop down to 6 V DC level during the start-stop function.
- The nominal delivered power to the receiver is 5 W (at the output of the receiver).
- Designed to meet the Qi/PMA specification.
- Working frequency: 110 kHz for Qi/PMA devices.

**Contents**

# 2. Hardware Setup

## 2.1 Pack content

1. WCT Automotive A13 board
2. Power supply connector
3. Power supply 12V
4. USB-UART converting board
5. Touch board

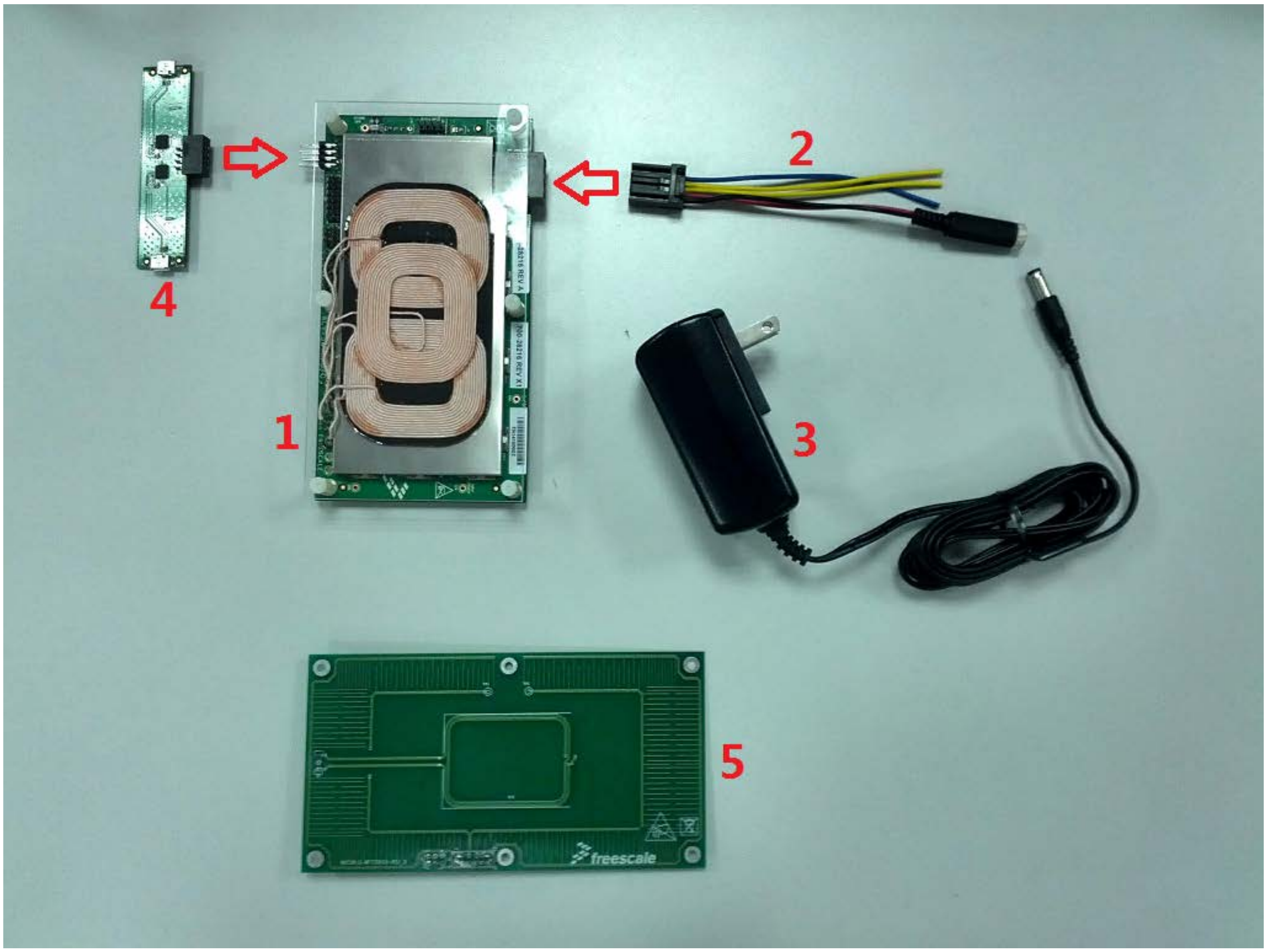


**Figure 1 Hardware pack contents**

## 2.2 Board description

The WCT is connected to the system by the main power connector. It comprises the automotive battery connection (red wire = +12V line, black wire = GND line), the CAN connection (yellow wires), and the IGNITION (blue wire).

The connectors on the bottom edge of the board provide a JTAG connection for programming and debugging, 2xSCI for the FreeMASTER tool connection for the debug option and the console connection, and the temperature & touch sense to sense the temperature and receiver placed on the optional Touch Board that covers the coils and is the interface surface of A13.

The circuitry on the board is covered by the metal shield to lower the EMI and provide a fixed position for the coils. The figure below shows the device.

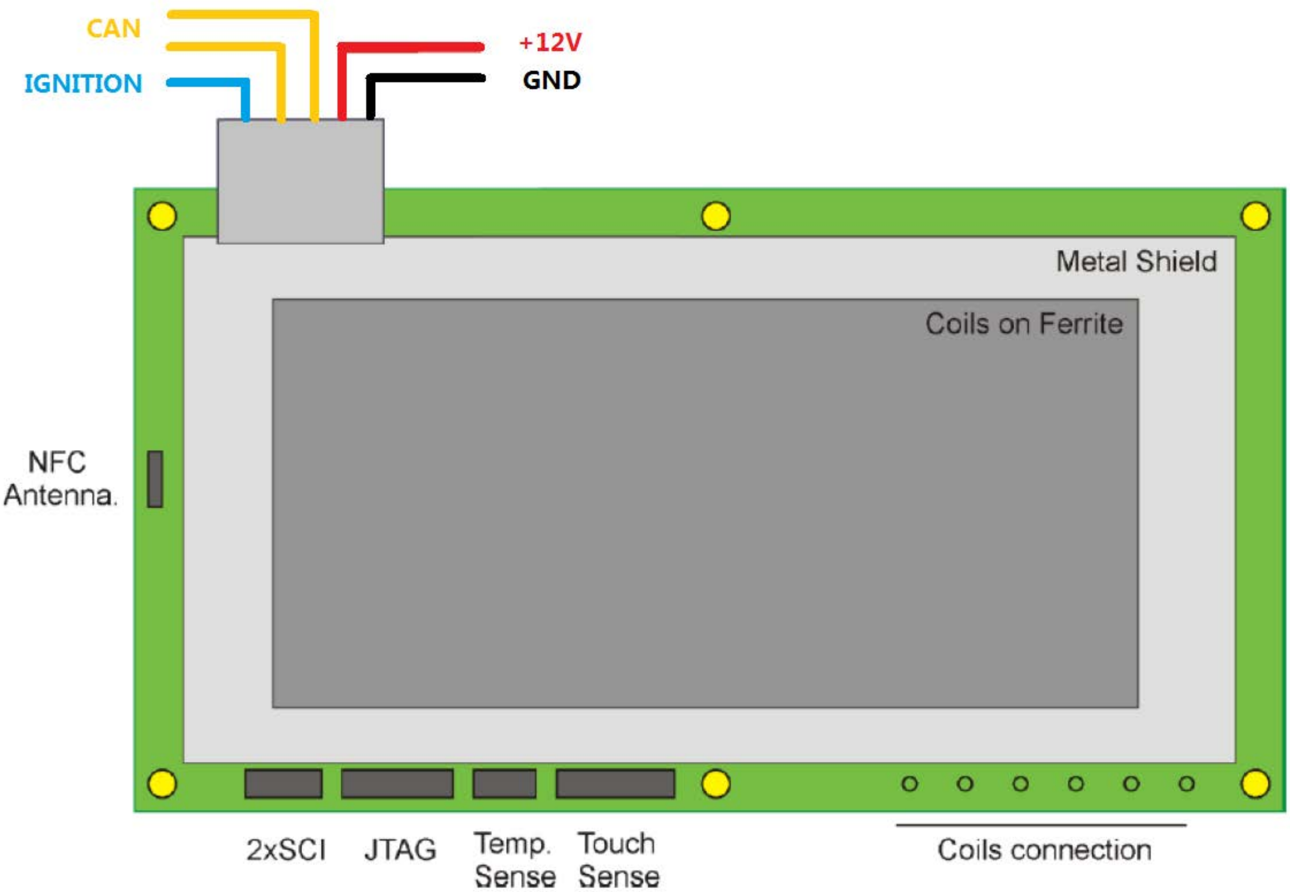


**Figure 2 Device**

## 2.3 Turning on a board

To turn on a board, perform the following steps:

1. Plug power supply 12V to the socket.
2. Plug the power supply connector into the board.
3. Connect power supply 12V and power supply connector.

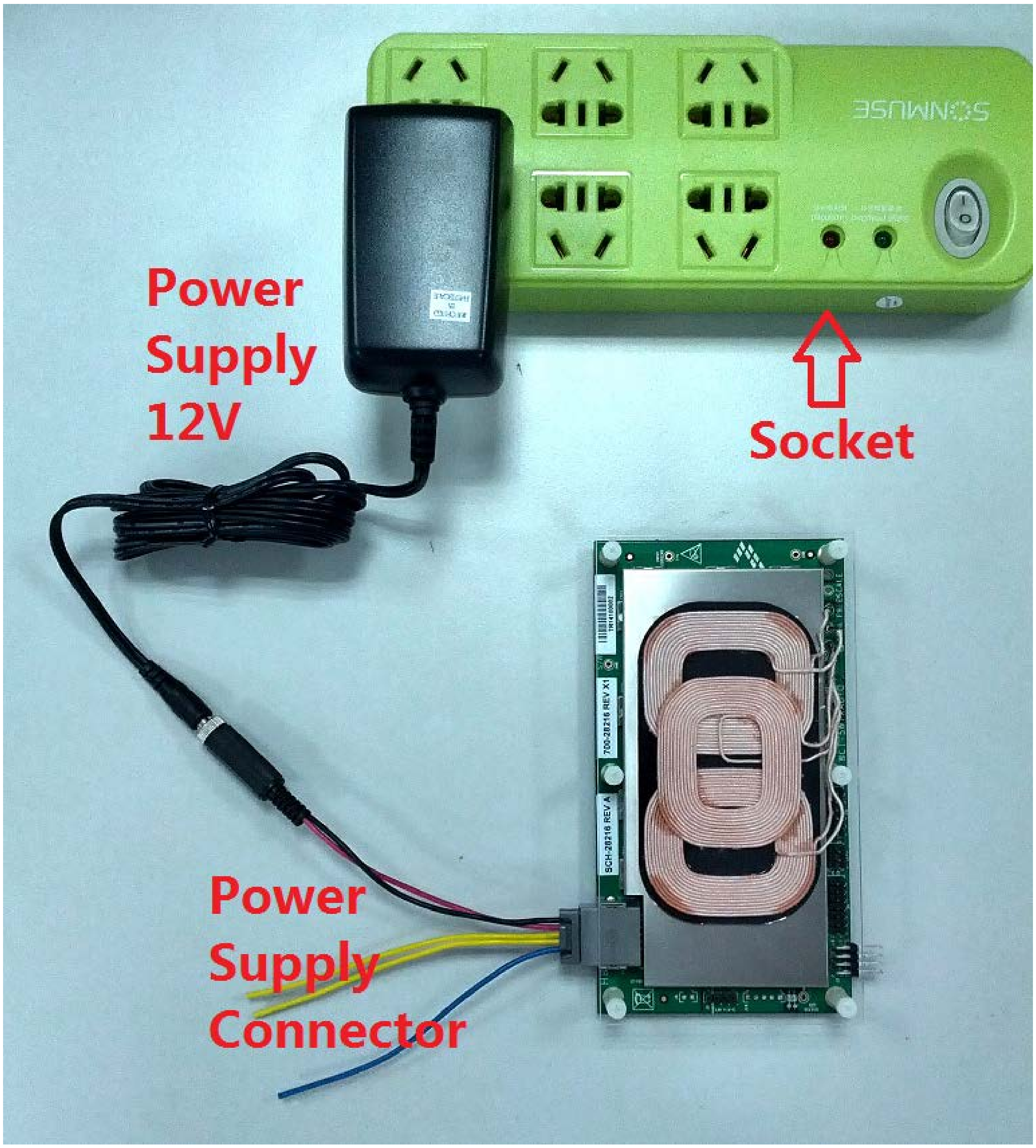


**Figure 3 Power supply components**

## 2.4 Hardware setup for FreeMASTER and Console communication

To set up the hardware for FreeMASTER and Console communication, perform the following steps:

1. Download the driver at www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx and install CP210xVCPInstaller_x86/x64.exe on your computer.
2. Plug the USB-UART converting board to SCI connector J2. The two MicroUSB connectors are for different purposes: FreeMASTER and Console.

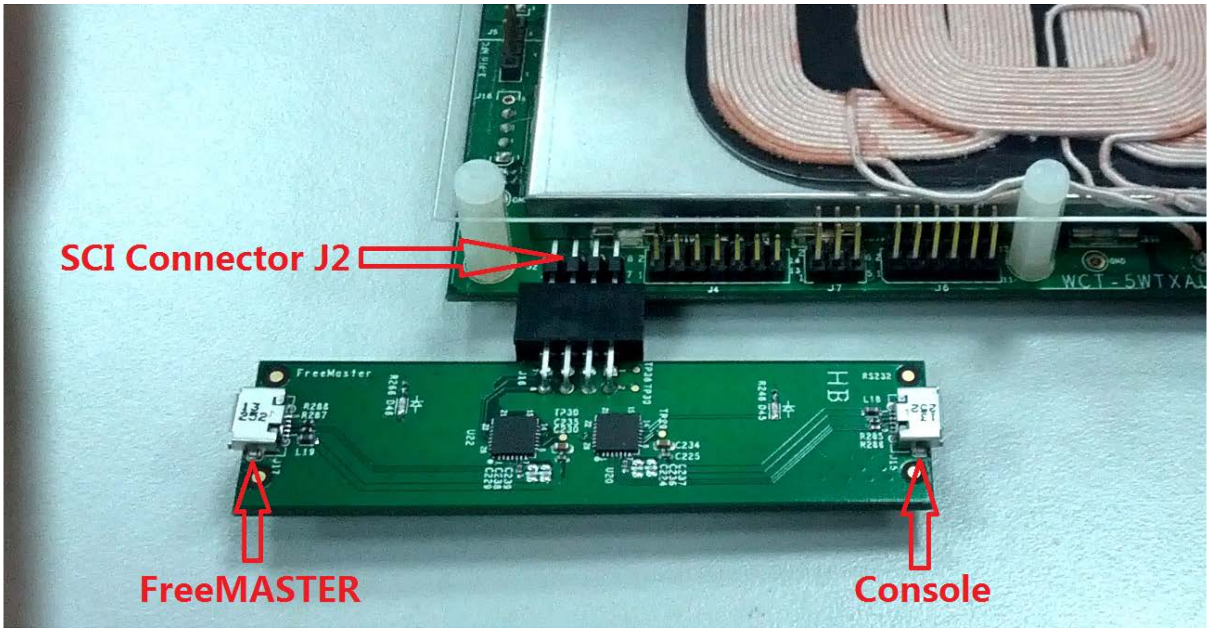


**Figure 4 SCI and MicroUSB connectors**

# 3. Application Operation

Connect the demo to the supply voltage +12 V DC. The WCT starts to send periodically the power ping to check the compatible device, wireless charging receiver (WCR), placed on the charging surface.

When the Qi/PMA-compliant device is placed on the top of the coils area, the WCT starts the charging process. If there is no correct answer from the WCR side, the power transfer will not start.

If the WCR answers properly, the power transfer starts. The actual level of the transferred power is controlled by the WCT in accordance with WCR requirements. The receiver side sends messages to the WCT through the power magnetic field. If the receiver-side device is fully charged, it sends the request for power transfer termination. The power transfer is terminated if the charged device is removed from the WCT magnetic field.

There are two types of Qi WCR devices: Qi_Ver-1.0 compliant and Qi_Ver-1.1 compliant. The main difference is that the later one provides the information to the WCT about the received power. This information serves for the Foreign Object Detection (FOD) on the top of the charging surface. The WCT calculates the difference between the power sent from the WCT and the power received by the WCR. If the difference is greater than the preset limit, the power transfer will be terminated in short time. The FOD power limit can be simply set by the application software.

# 4. Hardware Description

The following figure shows the block diagram of the automotive wireless charger A13.

Go to the Freescale website to obtain the latest Hardware Design files.

The whole design consists of several blocks, which are described in the following sections.

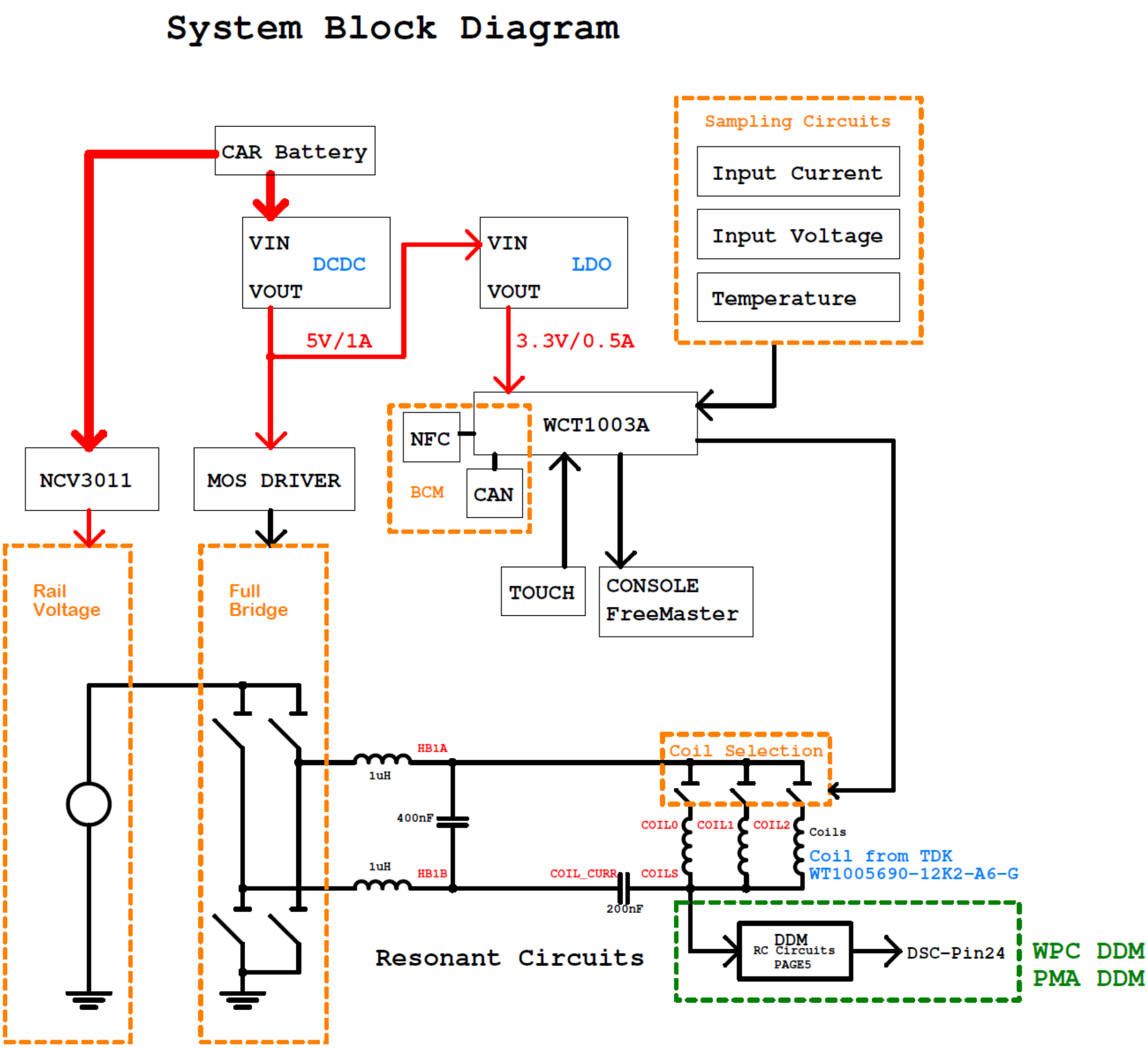


**Figure 5 Block diagram of the automotive wireless charger A13**

## 4.1 Input EMI filter

The input connector J1 provides the whole connection to the car wiring. It connects the battery voltage to the WCT and CAN communication interface. The Ignition signal is reserved if MC33907 is used.

The input filter consists of the Common Mode Filter FL1 and the filter capacitors C1, C3, C4, C14 and L1.

The main battery voltage switch is equipped with MOSFET Q1. This stage is controlled by the main controller WCT1001A/WCT1003A. The hardware overvoltage protection (more than 20 V DC) is also implemented by D1 and Q2 to this switch.

## 4.2 System voltage DCDC and LDO

The 12V Car Battery input is connected to a DCDC U25. Its output is 5 V and supplies LDO U26, MOSFET Driver, and CAN Transceiver. The 3.3V output of LDO is mainly for WCT1001A/WCT1003A and other 3.3 V components.

Mostly the DCDC works at the light-load conditions. High efficiency in light-load is significant important for this DCDC.

## 4.3 Rail voltage buck

The Qi/PMA specification for the A13 topology requires the DC voltage control of the output power. The buck converter provides the regulated DC voltage in range from 1 V DC to 10 V DC for the full-bridge power supply.

The output Rail voltage of the buck is controlled by the analog signal RAIL_CNTL generated by the WCT1001A/WCT1003A controller. The buck is also controlled by DCDC_EN for enabling or disabling, and is monitored by DCDC_PG for fault detection.

The buck converter uses the automotive grade DC-DC synchronous buck controller followed by the power stage Q5, Q6, L2. The input and output of the DC-DC converter are blocked by the series of the low ESR ceramic capacitors.

## 4.4 Full-bridge and resonant circuits

The full-bridge power stage consists of two MOSFET Drivers, U8 and U9, as well as four power MOSFETs, Q13, Q15, Q19 and Q20. The MOSFET Drivers are powered by the stable voltage level 5 V DC that decreases the power losses in the drivers and MOSFETs. The full-bridge power stage converts the variable DC voltage VRAIL to the square wave 50% duty-cycle high frequency voltage with the frequency equal to 110 kHz. The range of the used frequency (105 kHz to 115 kHz) is defined in the Qi specification for the A13 topology.

The resonant circuits consist of L10, L11, C113, C114, C115, C116, C111 and C112, all of which are fixed values defined in the Qi specification for the A13 topology. The snubber RC pairs connected in parallel to power MOSFETs are used to lower the high frequency EMI products. The coil discharge circuit Q23, R96, R108 is switched ON while the coils are not energized. This circuit maintains energy-free coils while the power transfer is not active.

The Current Sense Transformer T1 is used only when Powermat/PMA is employed.

## 4.5 Demodulation

There is one-way communication between the transceiver side and receiver side. The receiver measures the received power and sends back to transmitter the information about the required power level. This message is amplitude modulated (AM) on the coil current and sensed by A13.

There is only one demodulation circuit on A13, which is Digital De-Modulation (DDM) for both Qi compliant Receiver and PMA Receiver.

The RC circuits (C210, R116, R118, R224), known as DDM, sample the signals from the coil, compress the signal amplitude, and feed to ADC B-channel of WCT1001A/WCT1003A. The information about the current amplitude and modulated data are processed by the embedded software routine.

## 4.6 FOD

Required by Qi V1.1, FOD is introduced into A13. The input power to the full-bridge and output power from the coil should be calculated.

Current Sensor U21 plays the role of getting the full-bridge input current. The output power of the coil can be estimated by a specific curve-fitting method.

For details of FOD, see the *WCT1001A/WCT1003A Run-Time Debug User's Guide* (WCT100XARTDUG).

## 4.7 Coil selection

The Qi specification defines the A13 as the more-than-one coil topology with one coil energized at a time.

The coil selection topology connects only one coil to resonant circuits at a time. The coil is equipped with the dual N-MOSFETs, Q9, Q12, or Q16, controlled by the WCT1001A/WCT1003A controller through the control interface based on the low power bipolar transistors.

## 4.8 Analog sensing

Some ports of the ADC A-channel of WCT1001A/WCT1003A are used for sensing analog signals, such as temperature, full-bridge input current, input voltage, and Rail voltage.

## 4.9 Touch sensing

An accessory touch board is included in the hardware package of A13.

After the Plastic Board is replaced by Touch Board, and TOUCH is enabled in the software, any object placed on the top of the Touch Board, can be sensed by the WCT1001A/WCT1003A GPIO port. The GPIO Touch function is based on the capacitance change on the Touch Electrodes.

For a better power consumption, you can use a dedicated Touch Controller to free the WCT1001A/WCT1003A when waiting for a touch event.

## 4.10 Control unit

The control unit WCT1001A/WCT1003A is the heart of the whole application. This controller runs to code based on the dedicated wireless charging software library. It controls the whole wireless power transfer and runs other customer's tasks.

The following figure shows the Functional Block Diagram of WCT1001A/WCT1003A. The whole control consists of several blocks, which are described in the following sections.

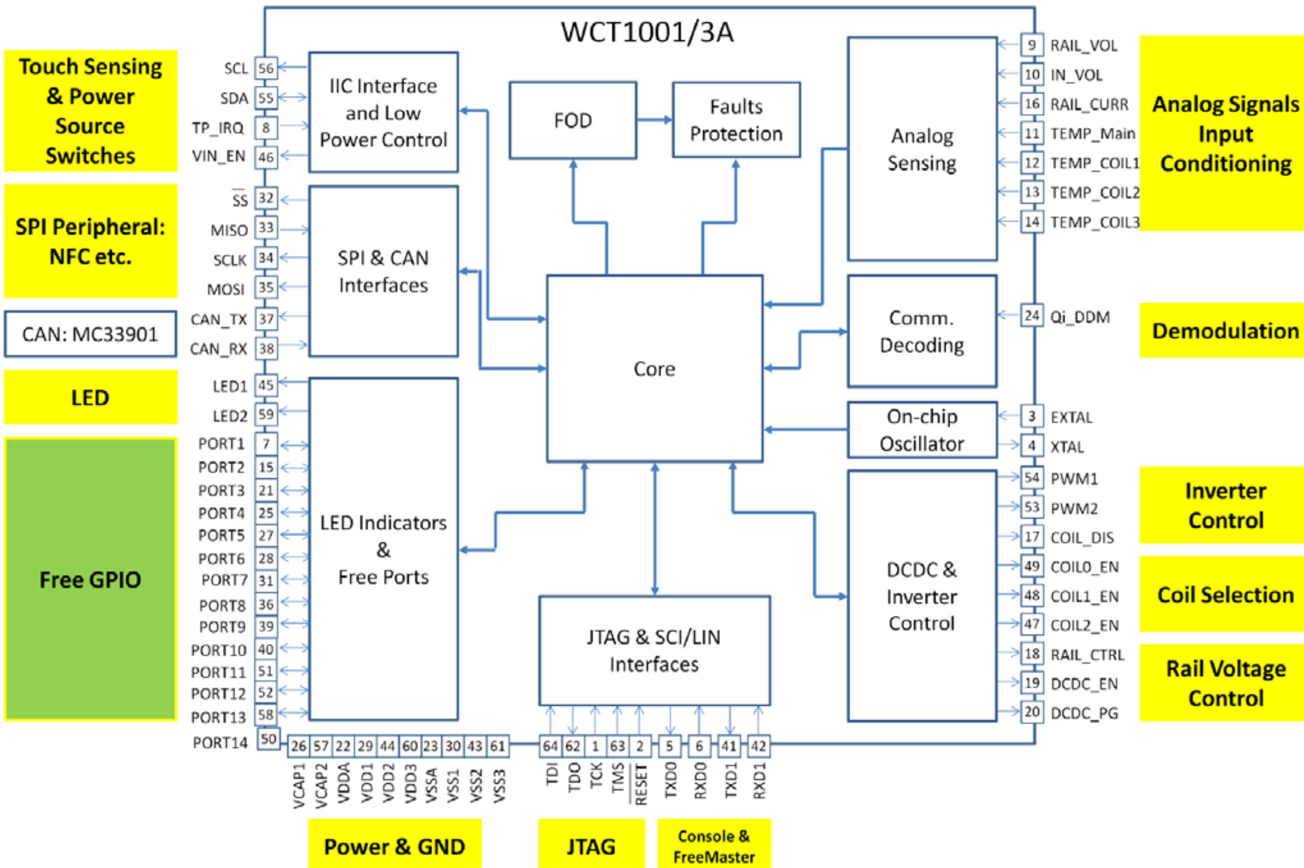


**Figure 6 WCT1001A/WCT1003A system functional block diagram**

- Touch Sensing: see Section 4.9 “Touch sensing”.
- Power Source Switch: see Section 4.1 “Input EMI filter”.
- SPI Peripheral: used to connect with a SPI peripheral, such as NFC.
- CAN Transceiver: to connect with the CAN transceiver.
- LED: outputs for the signal LEDs.
- Analog Signals Input Conditioning: see Section 4.8 “Analog sensing”.
- Demodulation: see Section 4.5 “Demodulation”.
- Inverter Control: see Section 4.4 “Full-bridge and resonant circuits”.

- Coil Selection: see Section 4.7 “Coil selection”.
- Rail Voltage Control: see Section 4.3 “Rail voltage buck”.
- JTAG: recommended to keep them as JTAG, not to use as GPIO.
- Console and FreeMASTER: serial communication interface for the Console and FreeMASTER.
- Power and GND: VCAP1 and VCAP2 are used for internal core circuits, requiring external capacitors.
- Free GPIOs: can be used freely by customers except for ADC input.

# 5. Application Monitoring and Control Through FreeMASTER

FreeMASTER is a user-friendly real-time debug monitor and data visualization tool for application development and information management. Supporting nonintrusive variable monitoring on a running system, FreeMASTER allows the data from multiple variables to be viewed in an evolving oscilloscope-like display or in a common text format. The application can also be monitored and operated from the web-page-like control panel.

## 5.1 Software setup

To set up the software, perform the following steps:

1. Install FreeMASTER V1.4.4 or later from the Freescale website: [freescale.com/freemaster](freescale.com/freemaster)
2. Plug the USB-UART converting board to SCI connector J2, and connect the FreeMASTER MicroUSB port to your computer.
3. Open the Device Manager, and check the number of the COM port.

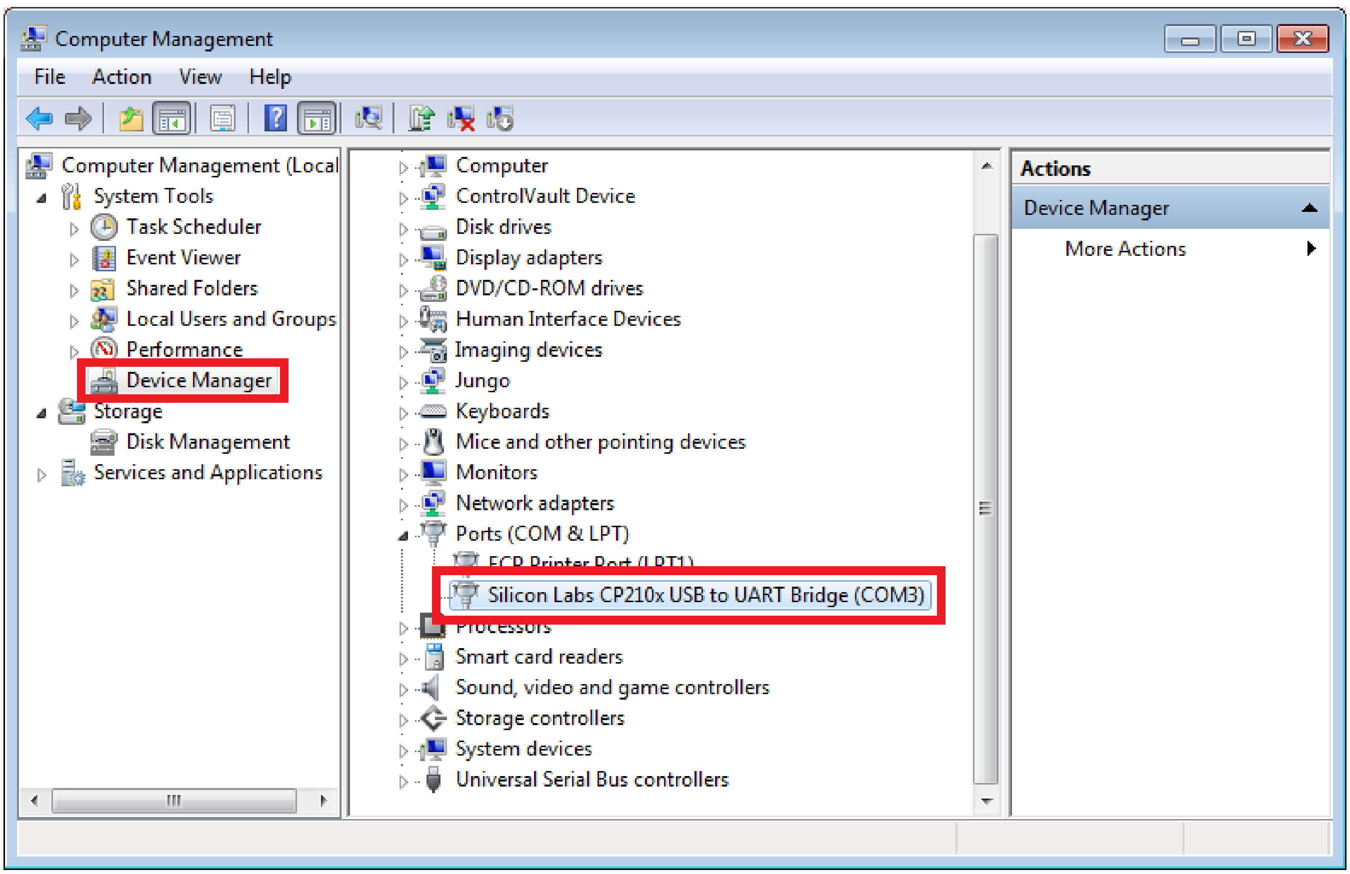


**Figure 7 Device manager**

4. Unpack the embedded source code to your local disk.
5. Start the FreeMASTER application by opening:

- MWCT1003A
  <unpacked_files_location>/A13/example/WCTxxxx/WCTAutoA13_WCT1003A/WCTAutoA13_WCT1003A.pmp
- MWCT1001A
  <unpacked_files_location>/A13/example/WCTxxxx/WCTAutoA13_WCT1001A/WCTAutoA13_WCT1001A.pmp

6. Choose **Project** –> **Options**.

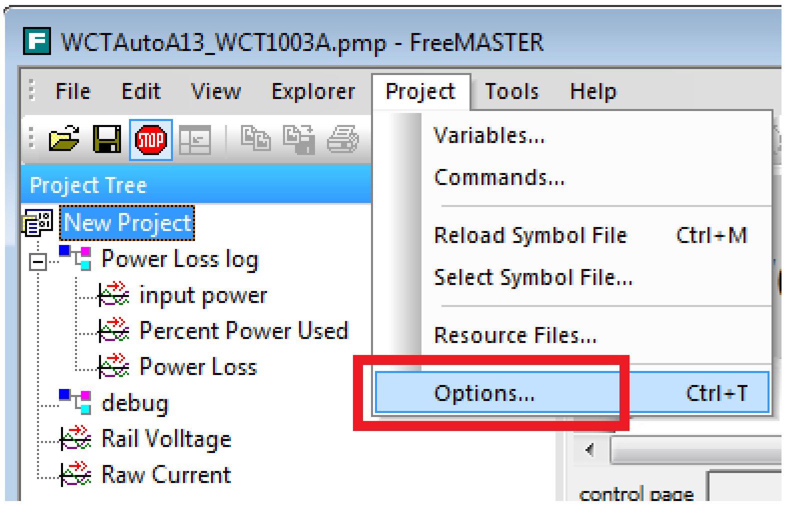


**Figure 8 Choosing Options**

7. Ensure that the correct virtual **Port** (according to Step 3) and **Speed** is selected.

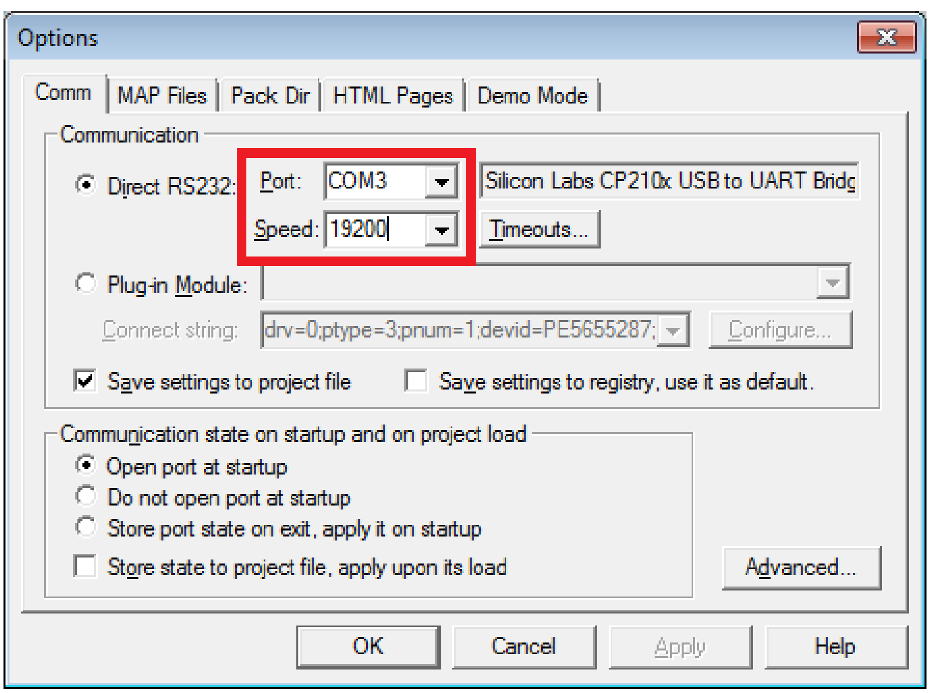


**Figure 9 Setting Port and Speed**

8. Ensure that the MAP file is correct. The default directories are as follows:

- MWCT1003A

  <unpacked_files_location>/A13/example/WCTxxxx/WCTAutoA13_WCT1003A/cw10/LDM_Debug/WCTAutoA13_WCT1003A.elf

- MWCT1001A

  <unpacked_files_location>/A13/example/WCTxxxx/WCTAutoA13_WCT1001A/cw10/SDM_Debug/WCTAutoA13_WCT1001A.elf

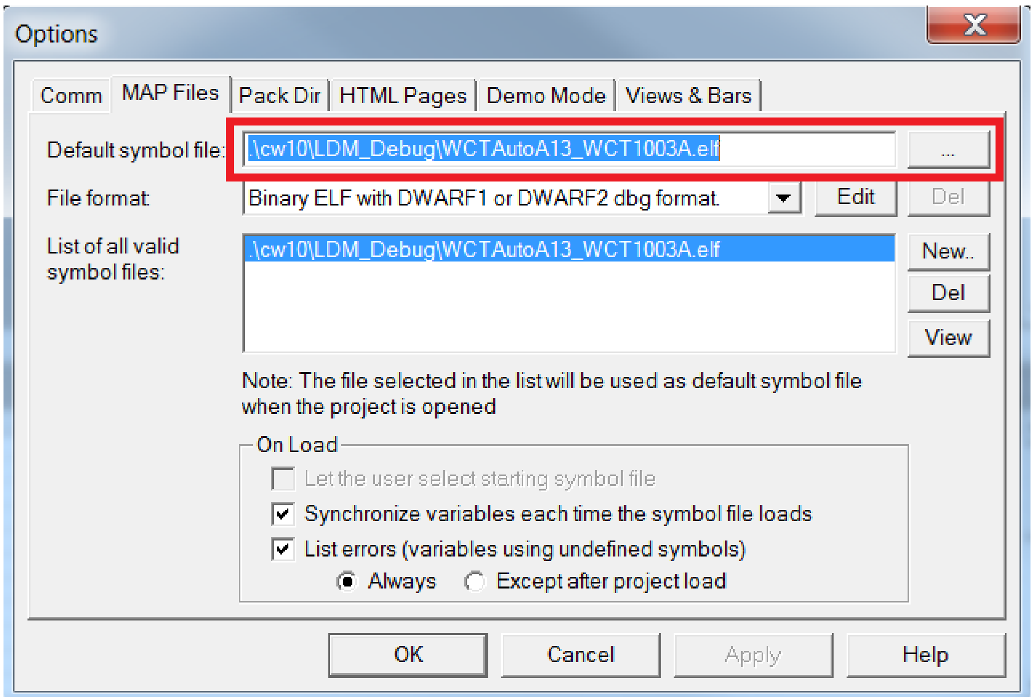


**Figure 10 Setting the MAP file**

If you are not sure whether the .elf file matches the board, you can try another method to connect FreeMASTER with the board. WCT GUI supports operations without the .elf file. It supports monitoring the real-time application variables described in Section 5.2 and reading and writing the application variables described in Section 5.3.

**Note:** This feature is not enabled in firmware versions earlier than GA3.2.

9. Connect FreeMASTER.

Power on A13, and then start the communication by clicking the STOP button on the FreeMASTER GUI.

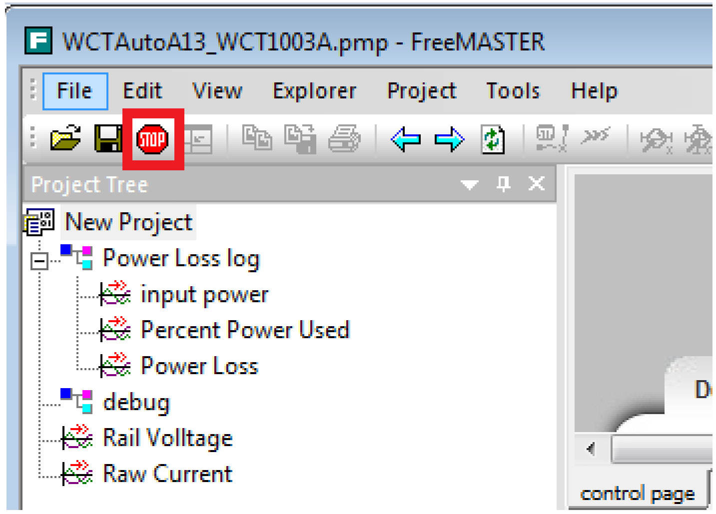


**Figure 11 Stop button**

## 5.2 Real-time application variables monitoring

FreeMASTER enables to monitor and update all the application global variables. In this application, several key variables are displayed in the scope windows. The user can observe the following variables in real time during charging or idle mode:

- Input power
- Power used
- Power loss
- Rail voltage
- Raw current

Particular charts are accessible by clicking on the name of the scope window.

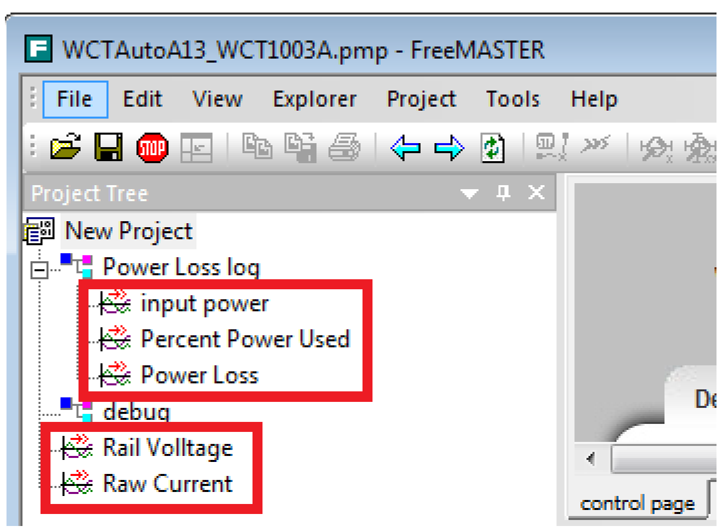


**Figure 12 Real-time application variables**

## 5.3 Application parameters modification

The application parameters can be easily viewed and changed on the control panel. The control panel contains the web page elements (buttons, check boxes, text fields) that enable a user-friendly way to visualize and change the application control parameters.

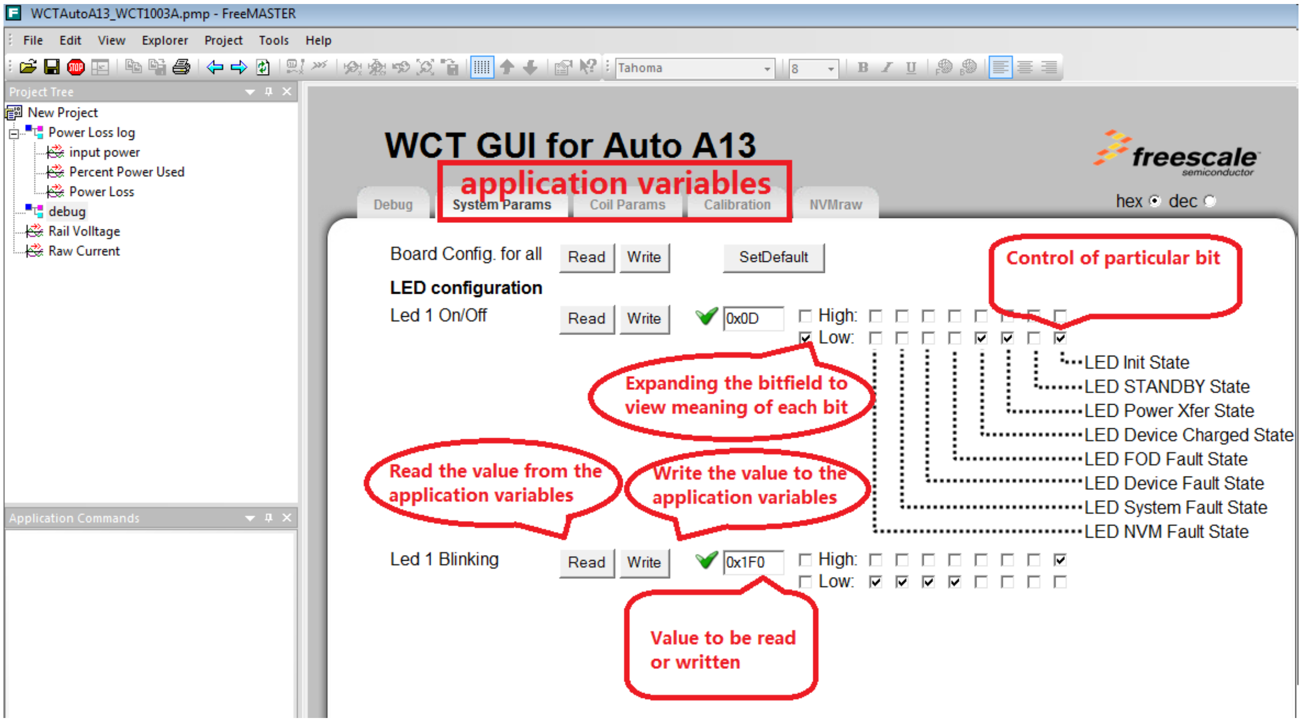


**Figure 13 Application variables**

The application variables are divided into three tabs:

- ***System Params*** – group of general system parameters
- ***Coil Params*** – enables access to the variables related to the coils control
- ***Calibration*** – group of parameters for calibration of the input current, input voltage, and foreign objects detector

The meaning of each parameter is described next to the text field.

## 5.4 Debug mode

The most left **Debug** tab contains the control elements to trigger multiple events manually:

- Key Fob Avoidance Trigger
- Turn on/off the battery switch
- Enable debug mode of the WCT library
- Change coil frequency
- Change rail voltage
- Perform rail voltage calibration, input current calibration, or calibration of Foreign Object Detector

Before changing the parameters, click **Enter** next to **Debug Mode** to put the system to debug mode.

If Touch Sensing is enabled, to avoid putting system to sleep, scroll down to the lower part of the window, write **255** to **byTouchTimeout**, touch the Touch Board by finger, and press **Enter** on the keyboard.

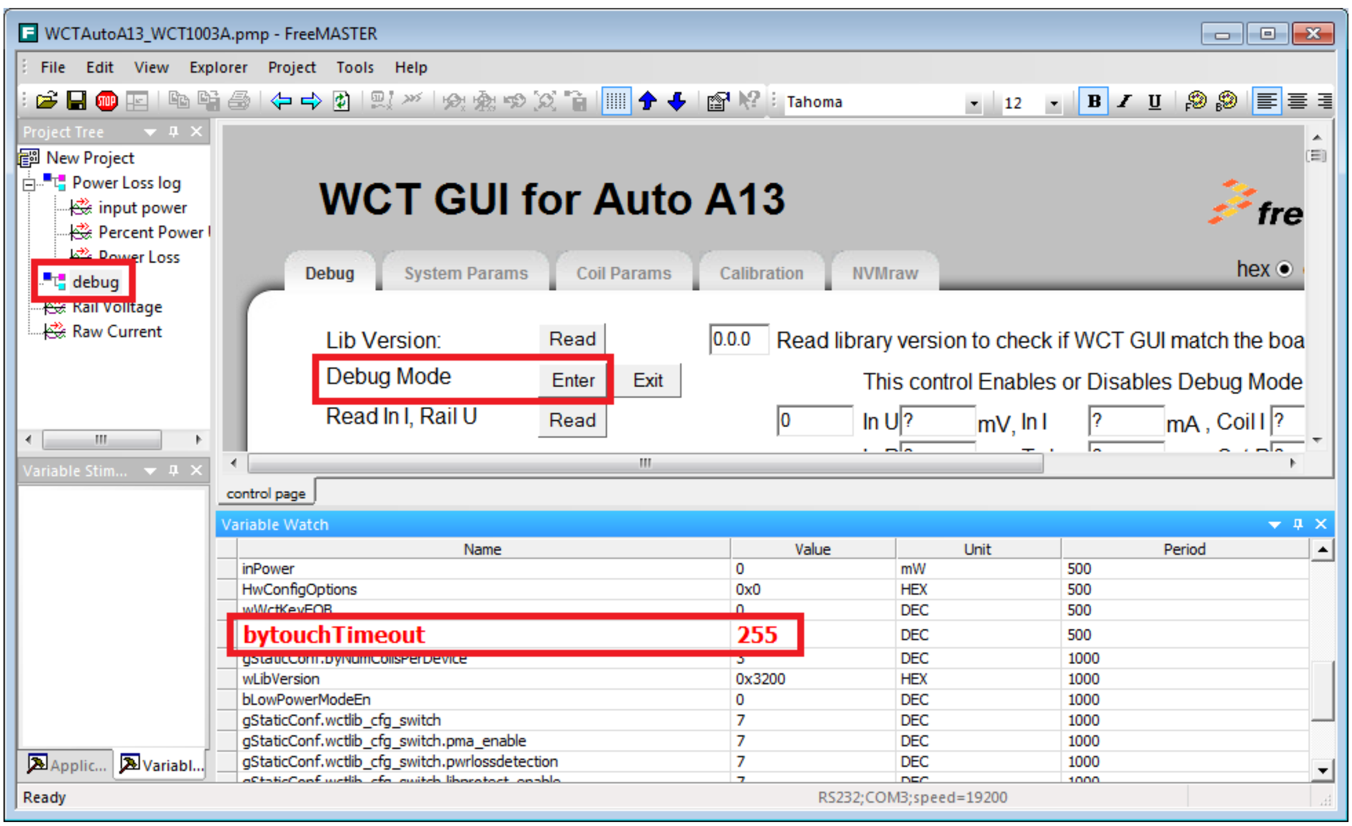


**Figure 14 Setting debug mode**

The following figure shows the **Debug** tab of the control page. The control elements have the same meaning as on the previously described tabs.

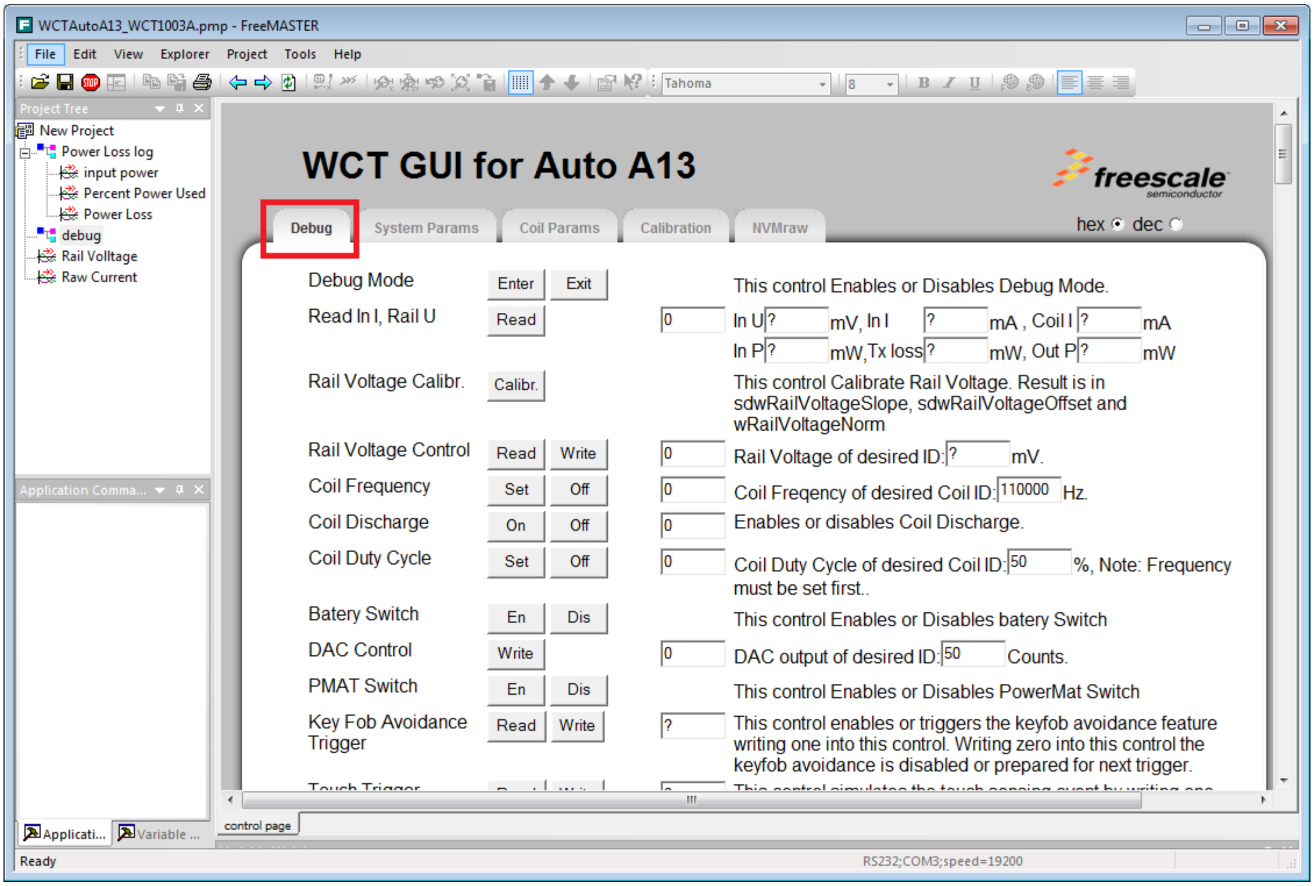


**Figure 15 Debug tab**

# 6. Application Monitoring Through Console

The application sends some information and error states through SCI to the console. The information is sent when the board is turned on, when the device is charging, or in case of some error state.

## 6.1 Software setup

1. Plug the USB-UART converting board to SCI connector J2, and connect the console MicroUSB port to your computer.
2. Open the **Device Manager**, and check the number of the COM port.

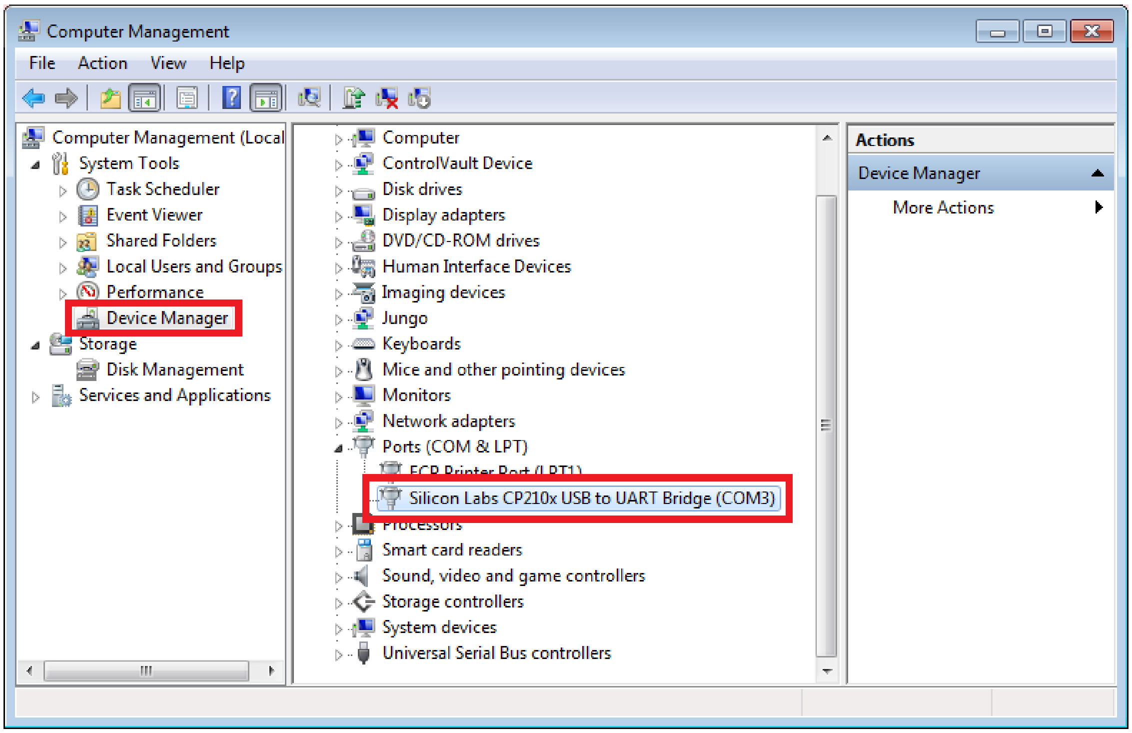


**Figure 16 Device Manager**

3. Run the communication program supporting console, such as HyperTerminal or RealTerm.
4. The following table shows the communication setup.

**Table 1 Port configurations**

| Port number | Serial port from Device Manager |
|---|---|
| Baud | 19200 |
| Data Bits | 8 |
| Stop Bits | 1 |
| Parity | None |
| Hardware Flow Control | None |
| Display As | ASCII |

5. Open the port or start communication, which depends on the used Terminal.

# 7. Program New Software and Calibration

Freescale provides users a software package, which includes a WCT1001A/WCT1003A project and a Bin file (.S). Users can flash alternative to the board. After flashing new software, board calibration must be carried out.

## 7.1 Install CodeWarrior 10.6 or later

1. Download installation files.

   For proper installation of CodeWarrior 10.6 or later, you need to install both **CodeWarrior for Microcontrollers 10.x** and **CW MCU v10.x Wireless Charging MWCT1xxx Service Pack**, and make sure that both files are with the same revision.

   Access the below webpage and sign in:

   http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-MCU10&fpsp=1&tab=Design_Tools_Tab

   Click the download button and select 10.6 version in the prompted page.

   Sign in and accept the license agreement

   Download CodeWarrior 10.6 and CW MCU v10.6 Wireless Charging MWCT1xxx Service Pack

   In addition, there are two kinds of CodeWarrior editions:

   - **Evaluation Software**

     Evaluation Editions are available for free download and work for a limited time, that is, the tools should be installed with all the features available for 30 days.

   - **Special Edition Software**

     Special Editions are fully functional free download versions of the CodeWarrior Development Studio with code size restrictions on the build chain.

2. Double-click **Setup.exe** after downloading.

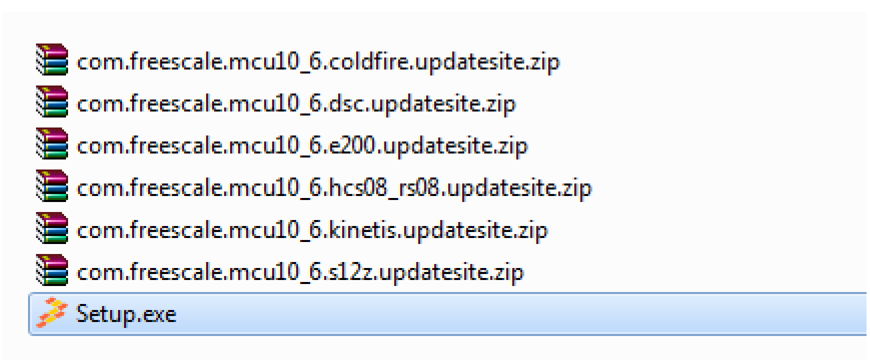


**Figure 17 Setup file**

3. Make sure that **DSC** is selected.

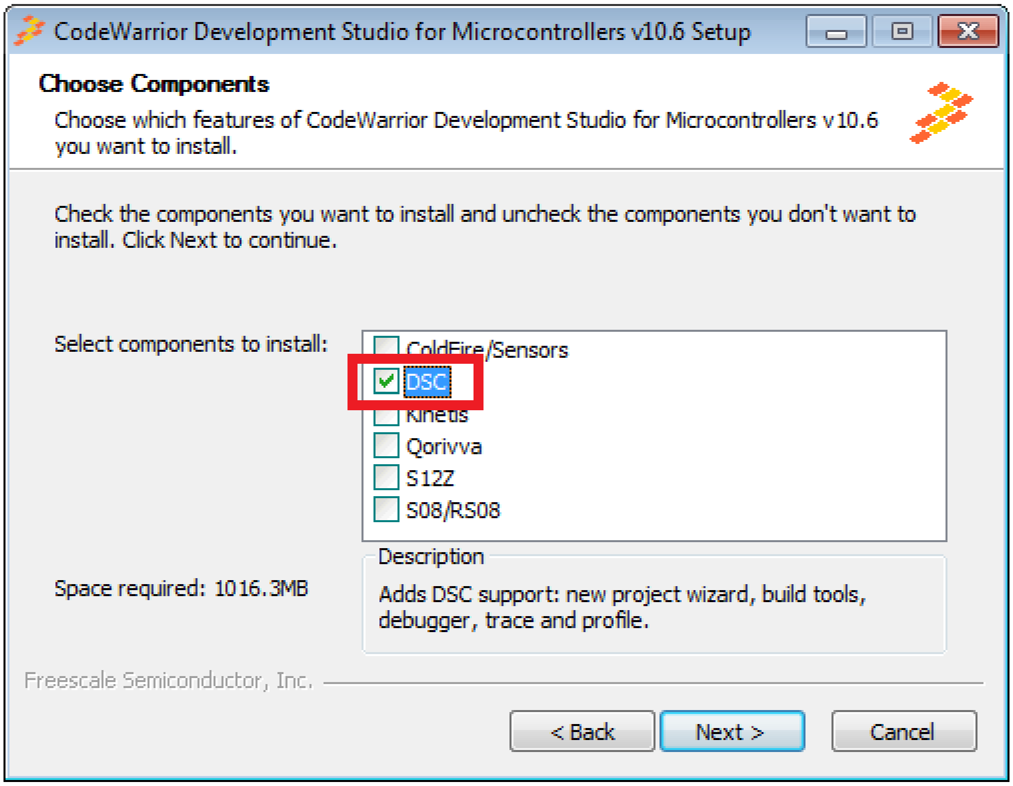


**Figure 18 DSC installed**

4. Click **Allow access** when the **Windows Security Alert** dialog box is displayed.

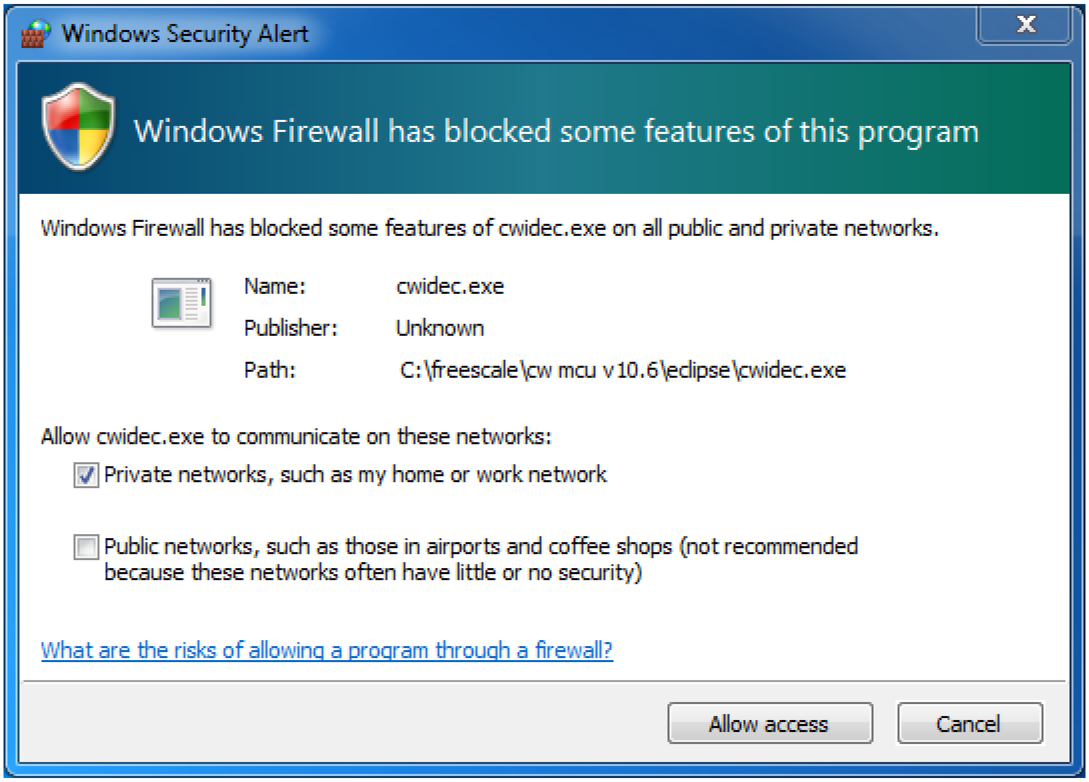


**Figure 19 Windows Security Alert dialog box**

5. Select **Always trust software from "Freescale Semiconductor"**, and then click **Install**.

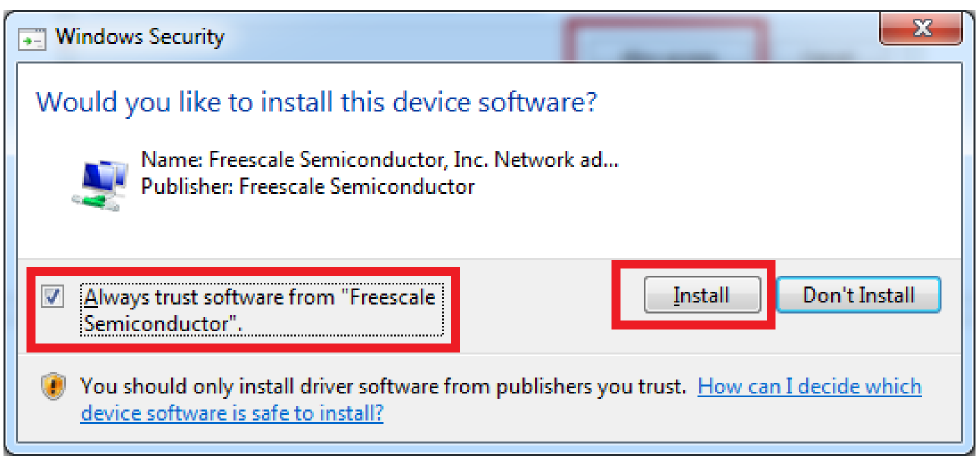


**Figure 20 Windows Security dialog box (1)**

6. Select **Always trust software from "Jungo LTD"**, and then click **Install**.

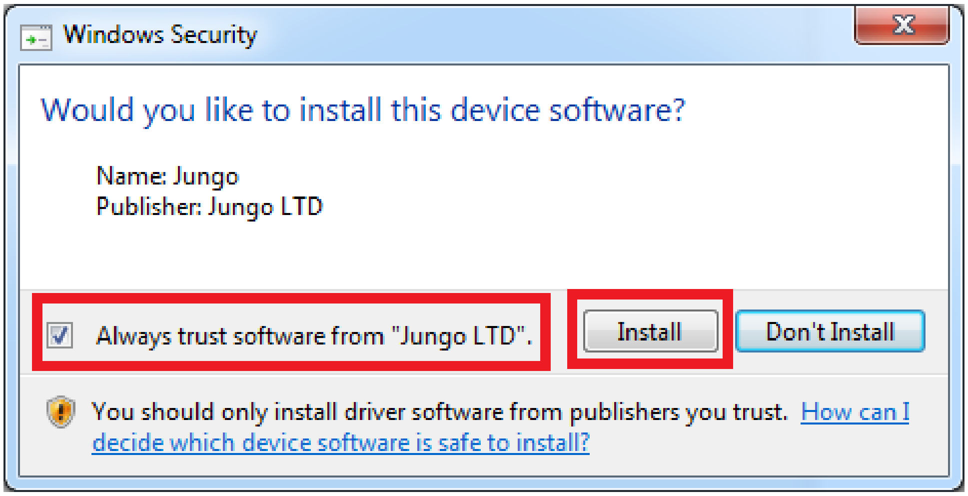


**Figure 21 Windows Security dialog box (2)**

7. Select **Always trust software from "PE Microcomputer Systems, Inc"**, and then click **Install**.

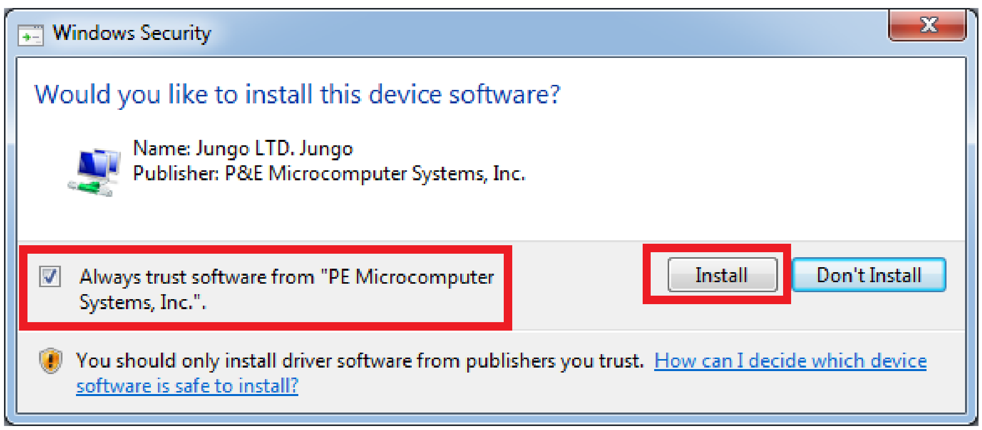


**Figure 22 Windows Security dialog box (3)**

8. Launch CodeWarrior, create a folder workspace, and select it as the default workspace.

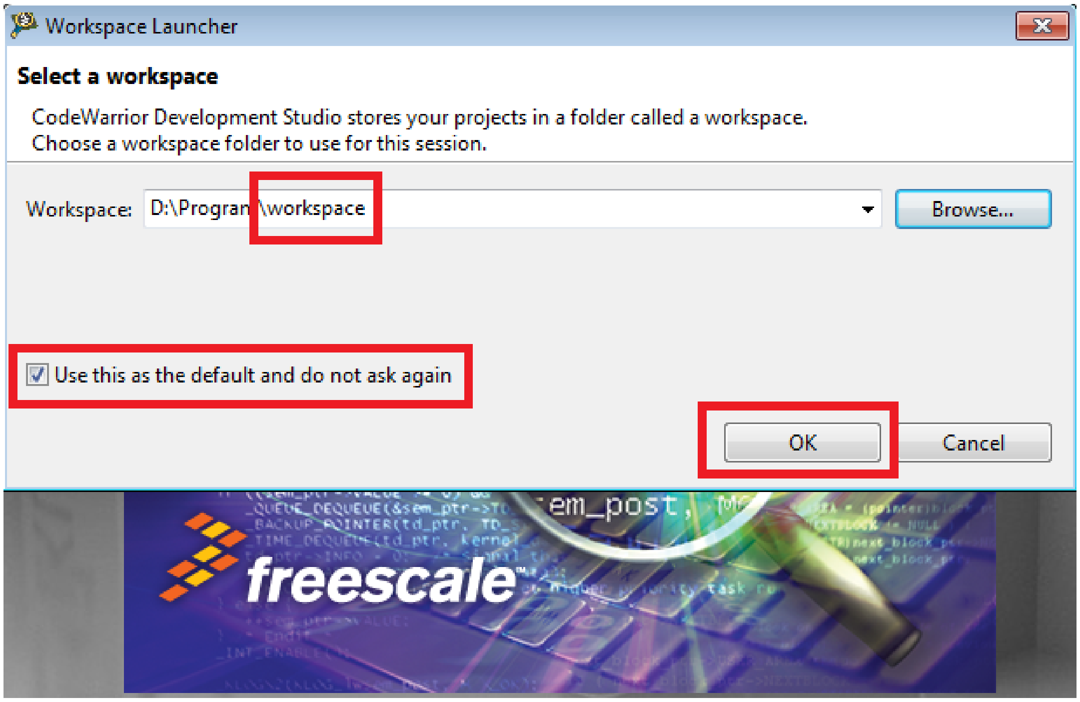


**Figure 23 Workspace Launcher dialog box**

9. Choose **Help –> Install New Software**.

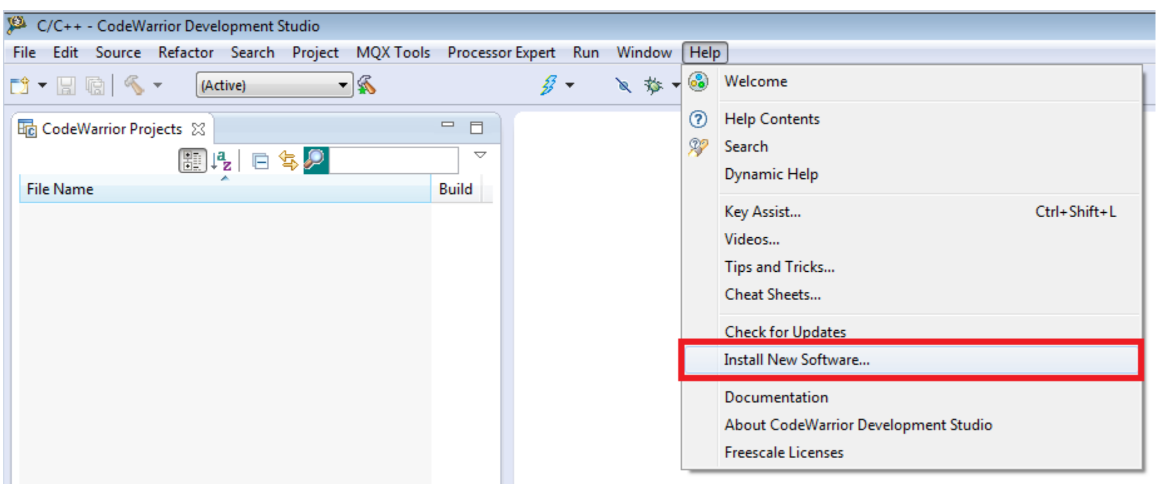


**Figure 24 Install New Software**

10. Click **Add** and **Archive**, and then select **com.freescale.mcu10_6.WCT_WCT100x_WCT1100x_WCT1200x.win.sp.v1.0.1.zip** (CW MCU v10.6 Wireless Charging MWCT1xxx Service Pack).

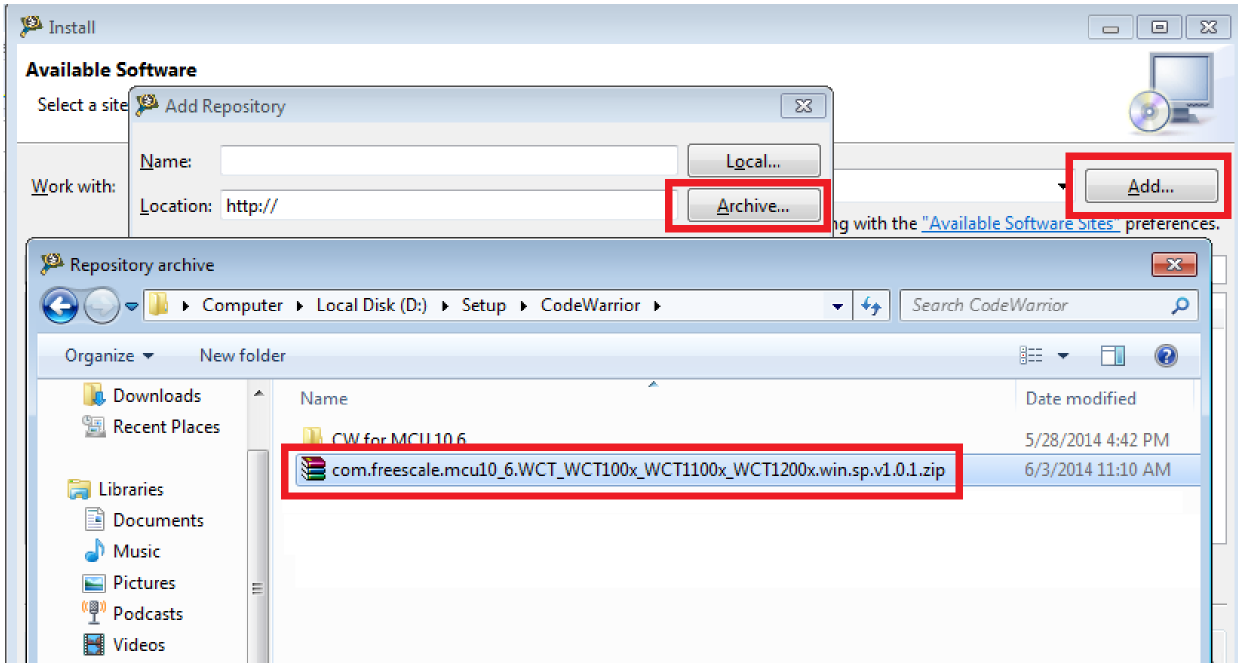


**Figure 25 Selecting the service pack**

11. Select **MCU v10.6 DSC Service Packs**, and then click **Next**.

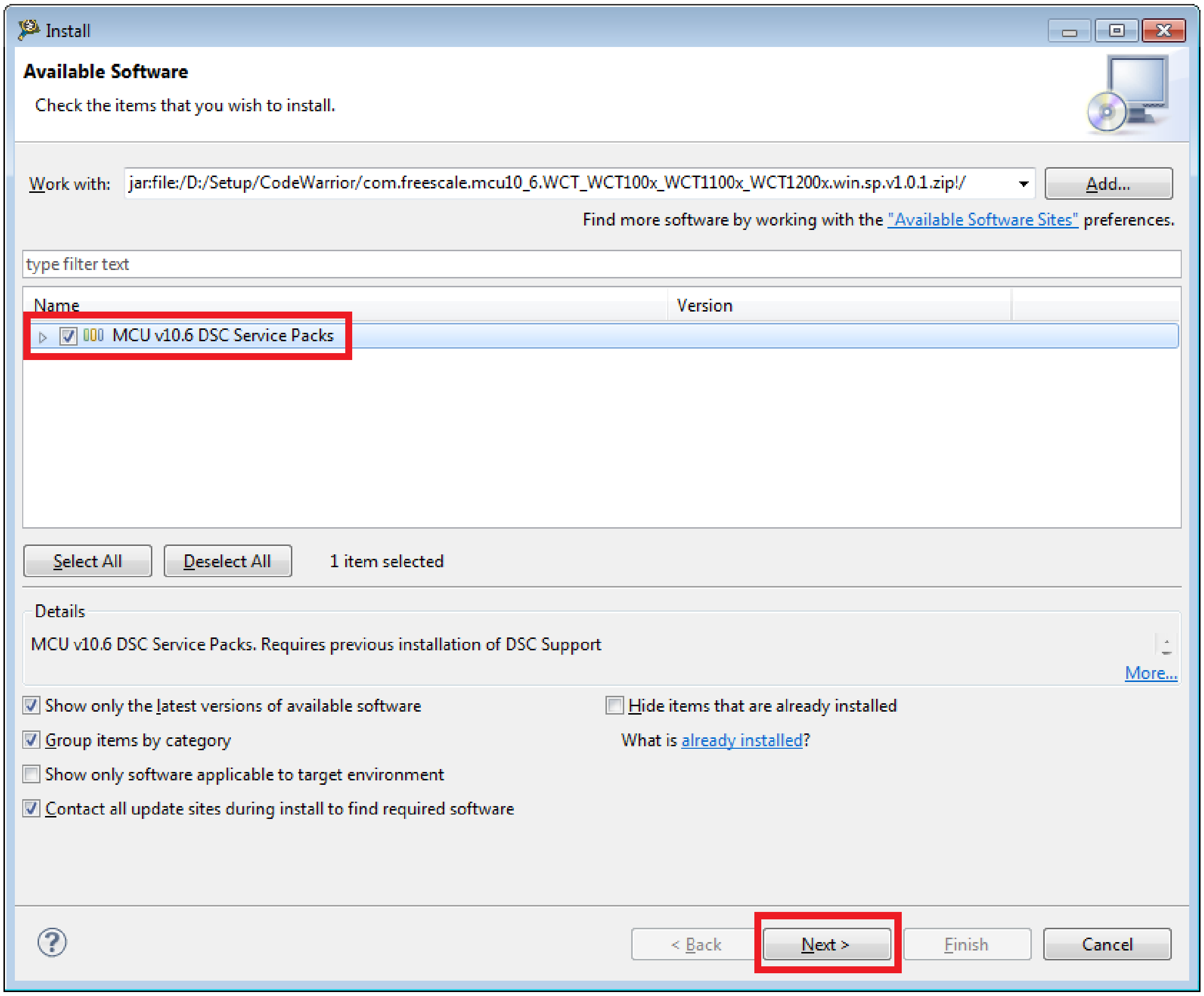


**Figure 26 Selecting MCU v10.6 DSC Service Packs**

12. Click **Next**.

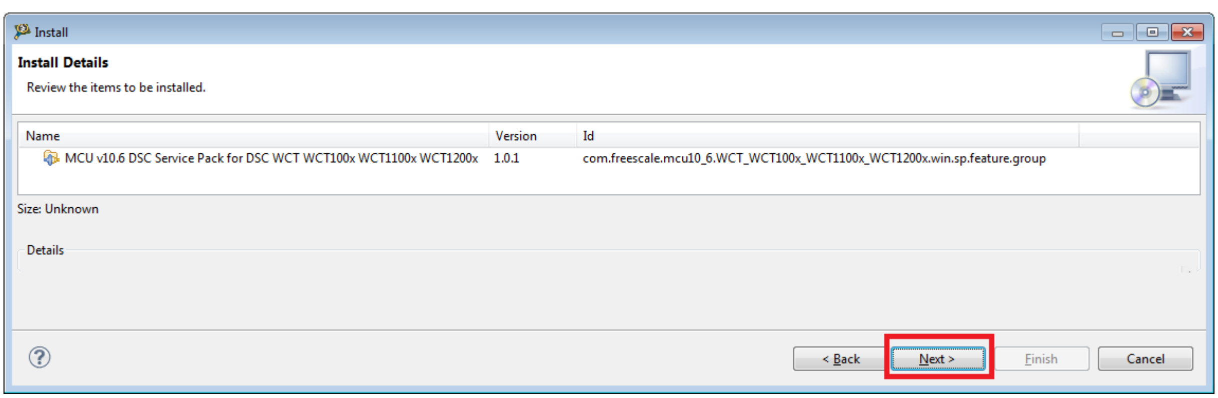


**Figure 27 Installing the pack**

13. Review the license terms. If you agree with the license terms, select **I accept the terms of the license agreement**, and then click **Finish**.

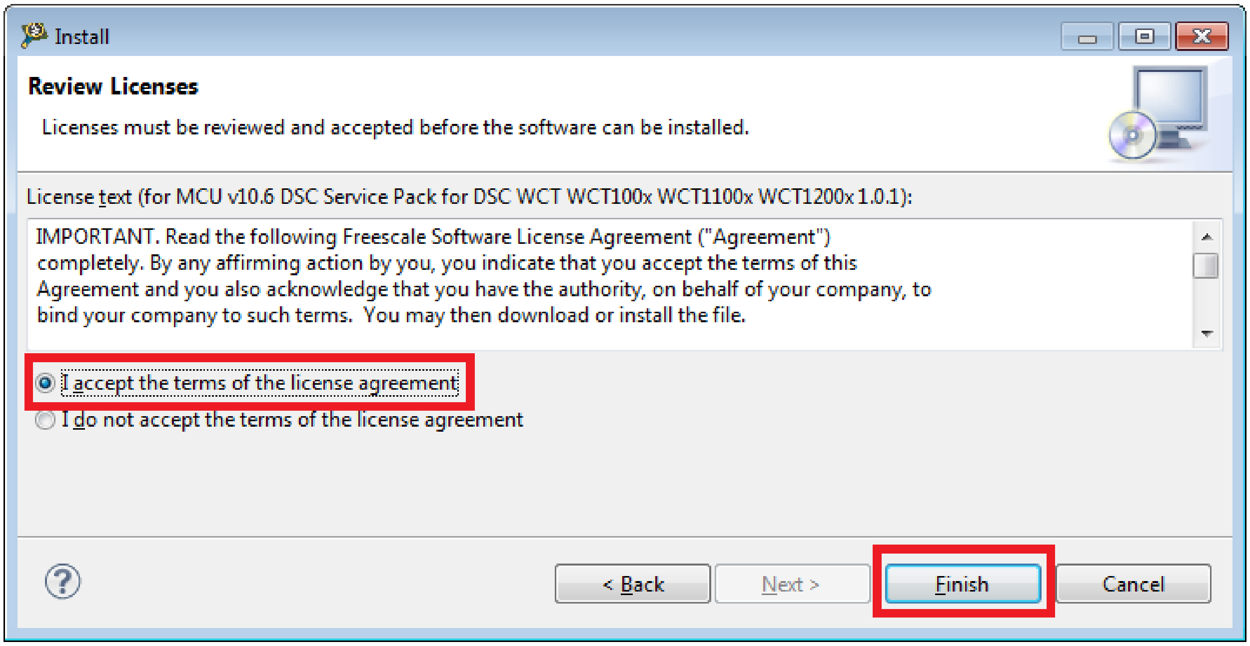


**Figure 28 Installation finished**

## 7.2 Board and programmer connection

1. Connect **FSL USB TAP ONCE** or **PnE U-MultiLink** to your PC and install the driver.

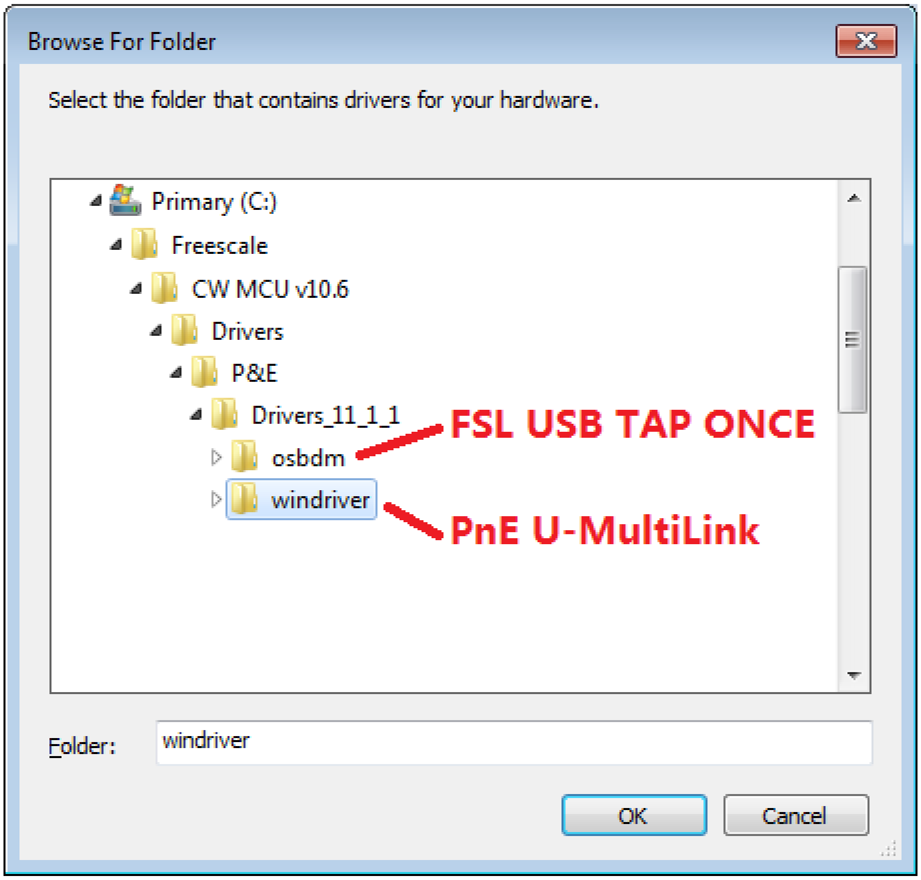


**Figure 29 Browse for Folder**

2. Connect the 14-pin debug cable to J4 of the board (notice pin-1 position of cable).


**Figure 28 Connecting the debug cable to the board**

## 7.3 Program using the project file

1. Drag the **.project** file to the **CodeWarrior Projects**.

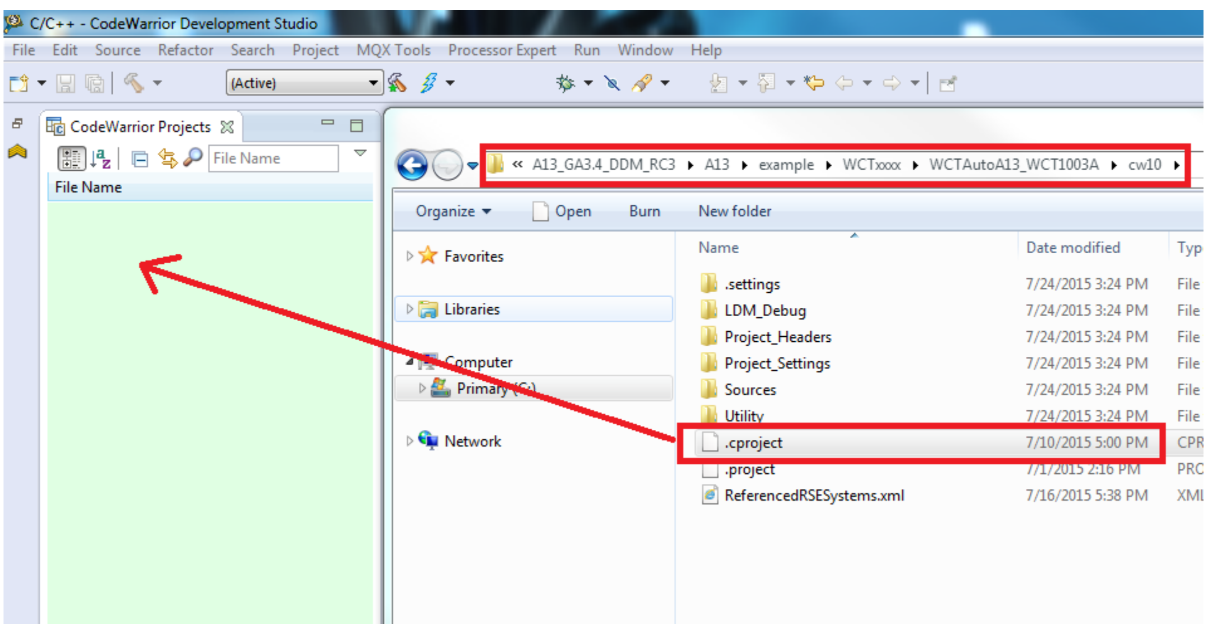


**Figure 29 Dragging the project file**

2. Choose **WCTAutoA13_WCT1003A** –> **LDM_Debug**. Select **SDM_Debug** if the chip is **MWCT1001A**.

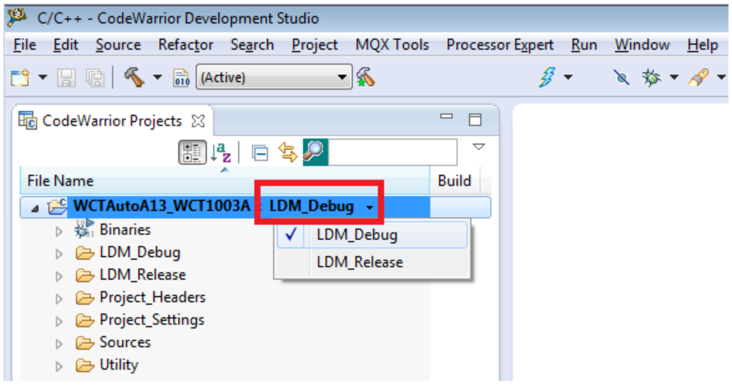


**Figure 30 LDM_Debug**

3. Right-click **WCTAutoA13_WCT1003A**, and then choose **Clean Project** and **Build Project**.

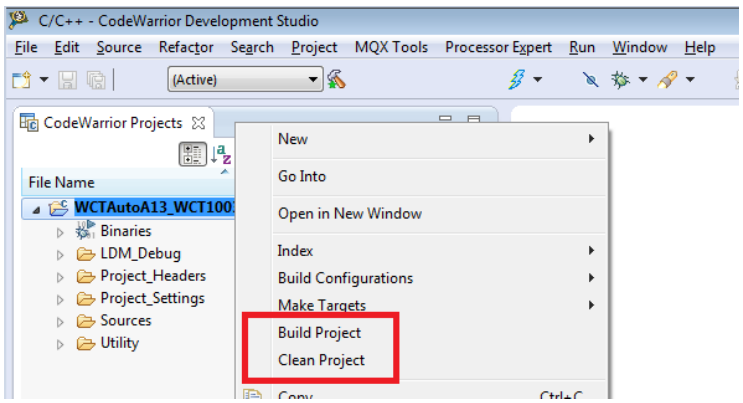


**Figure 31 Clean Project and Build Project**

4. Power on A13 and choose **Run –> Run Configurations**.

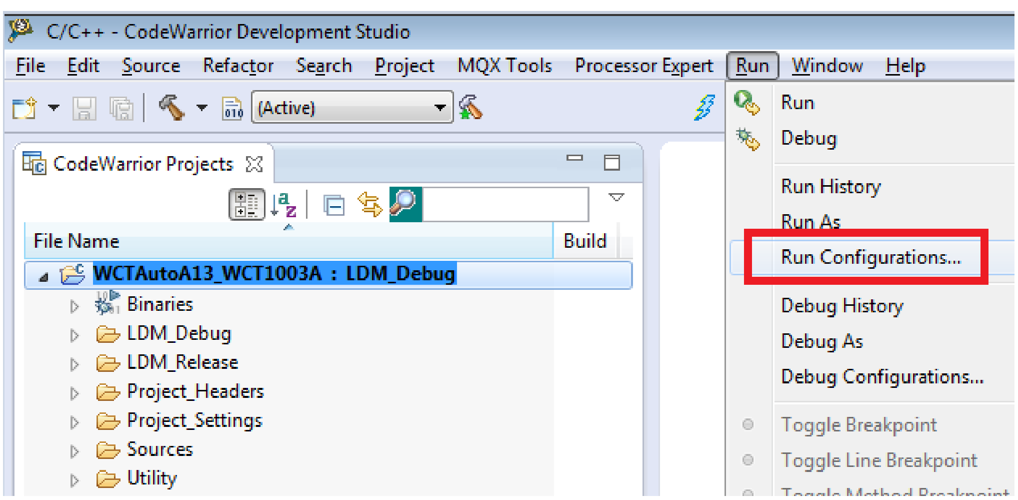


**Figure 324 Run Configurations**

5. Select **WCTAutoA13_WCT1003A_LDM_Debug_FSL USB TAP**, and click **Run**.

   Select **WCTAutoA13_WCT1001A_SDM_Debug_FSL USB TAP** if the chip is **MWCT1001A**.

   Select **PnE U-MultiLink** if the programmer is **MultiLink**.

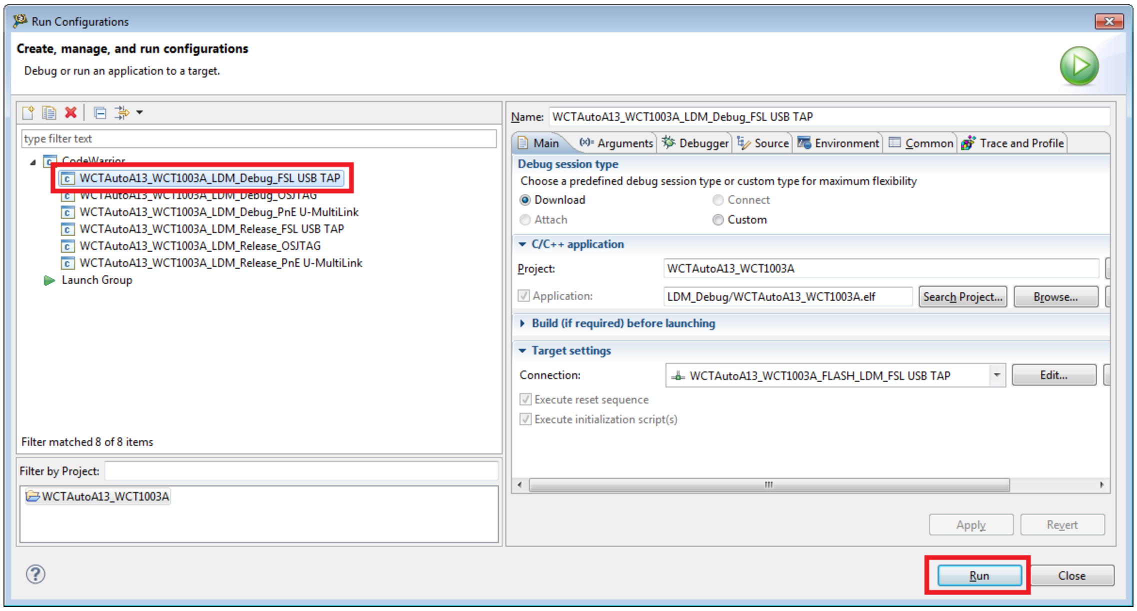

**Figure 33 Selecting WCTAutoA13_WCT1003A_LDM_Debug_FSL USB TAP**

6. Check the status at the bottom right corner, and wait until the programming is finished.

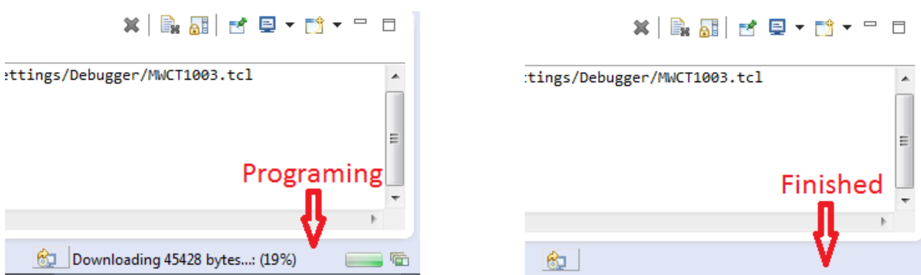


**Figure 34 Programming finished**

## 7.4 Program using the Bin file (.S)

1. Choose **Flash Programmer –> Flash File to Target**.

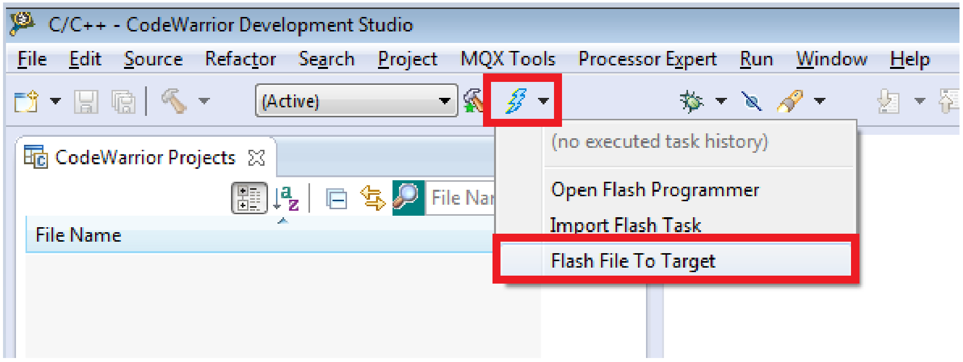


**Figure 35 Choosing Flash File to Target**

2. Click **New** to create a new connection.

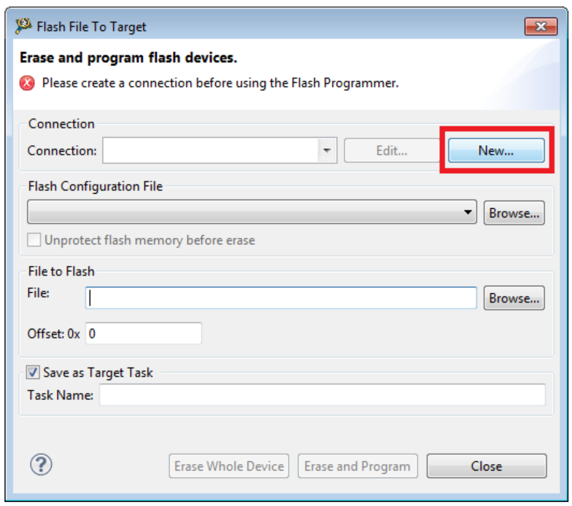


**Figure 36 Creating a new connection**

3. In the **Name** text box, enter a connection name (any name is OK), and click **New** to create a target.

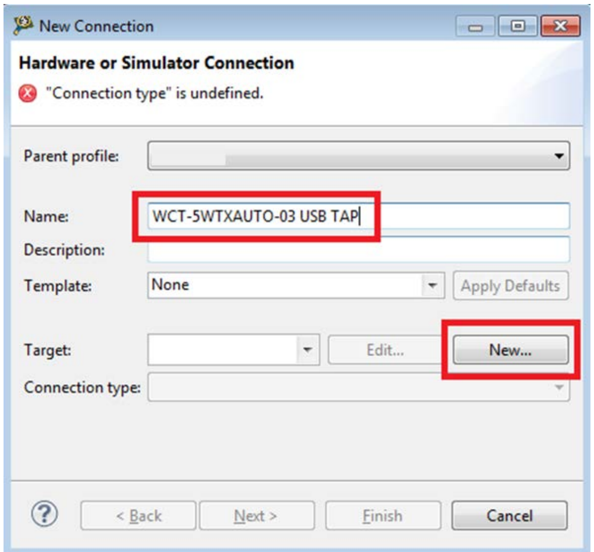


**Figure 39 Entering a connection name**

4. In the **Name** text box, enter a target name (any name is OK but cannot be same with the connection name), and choose **dsc.MWCT10xx -> MWCT1003** from the **Target Type** drop-down list.

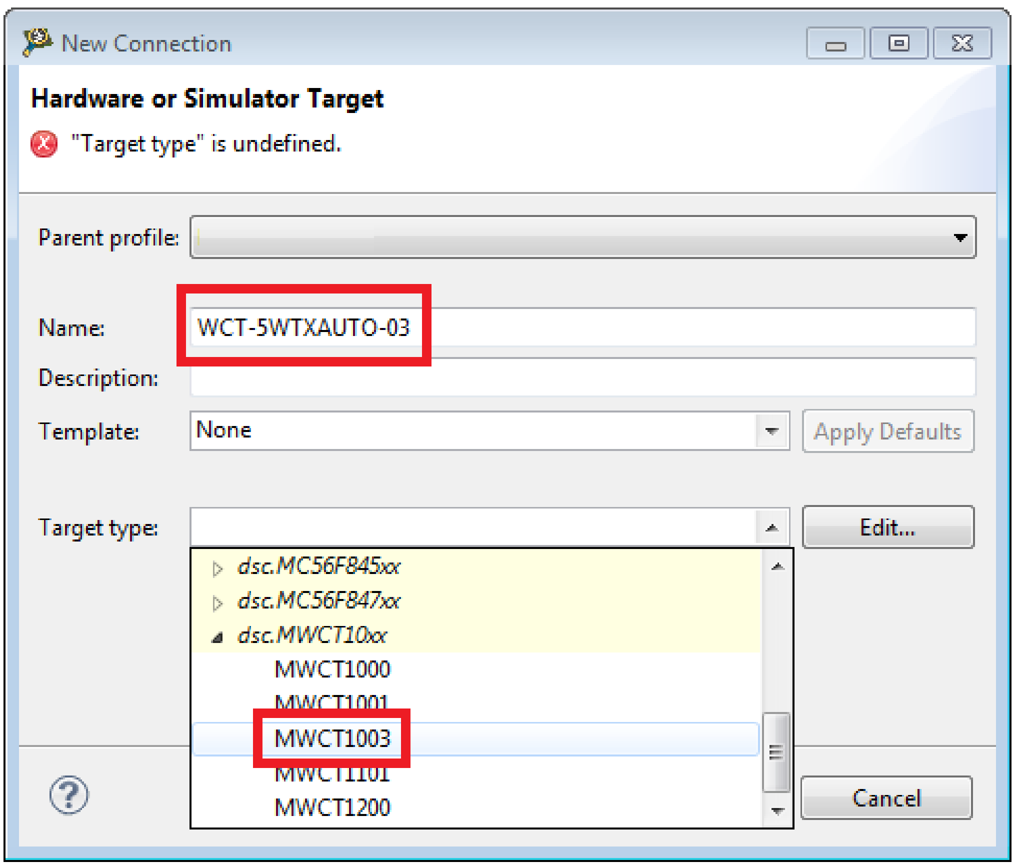


**Figure 37 Choosing MWCT1003**

5. Select **Execute reset** and **Initialize target**, set the initialization target file path to the CW installation folder, and then select **MWCT1003.tcl**.

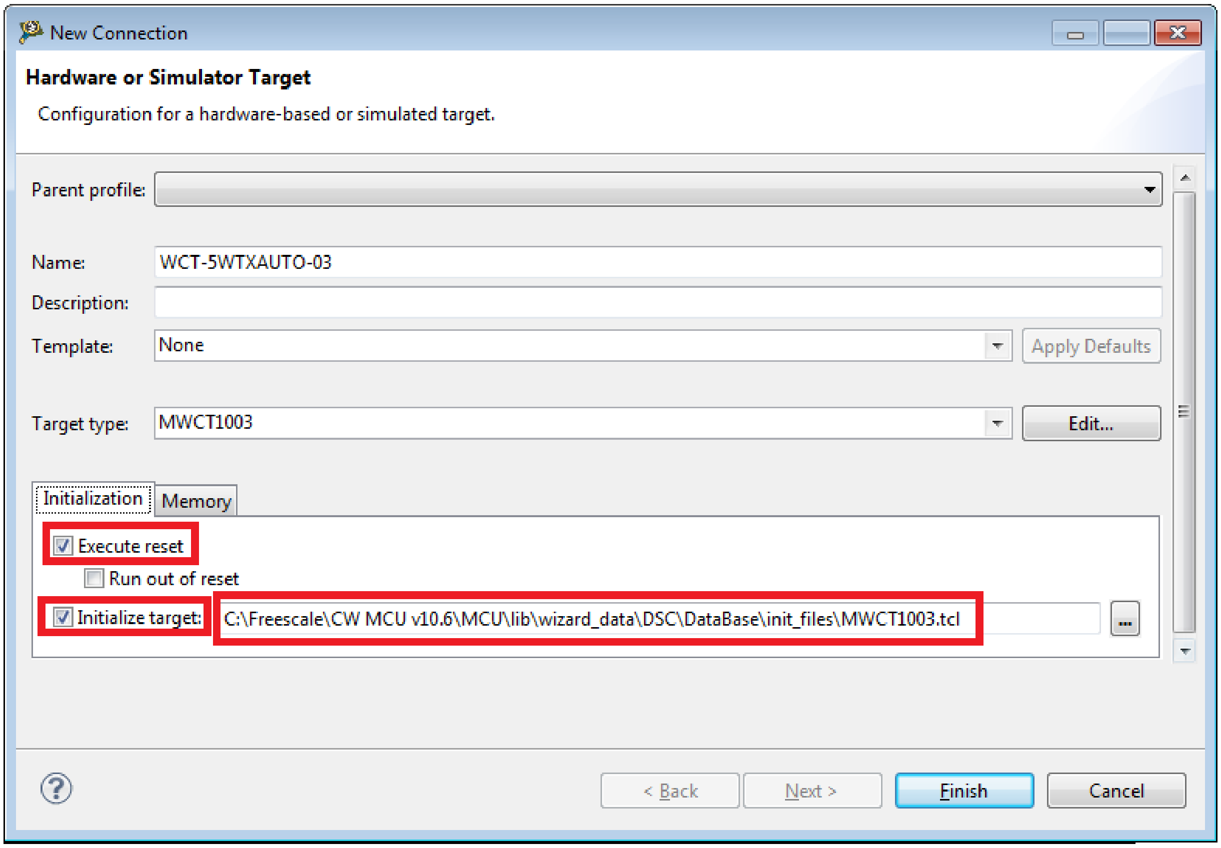


**Figure 38 Executing reset and initializing target**

6. Click the **Memory** tab. Select **Memory configuration**, set the memory configuration file path to the CW installation folder, and then select **MWCT1003.mem**. Click **Finish**.

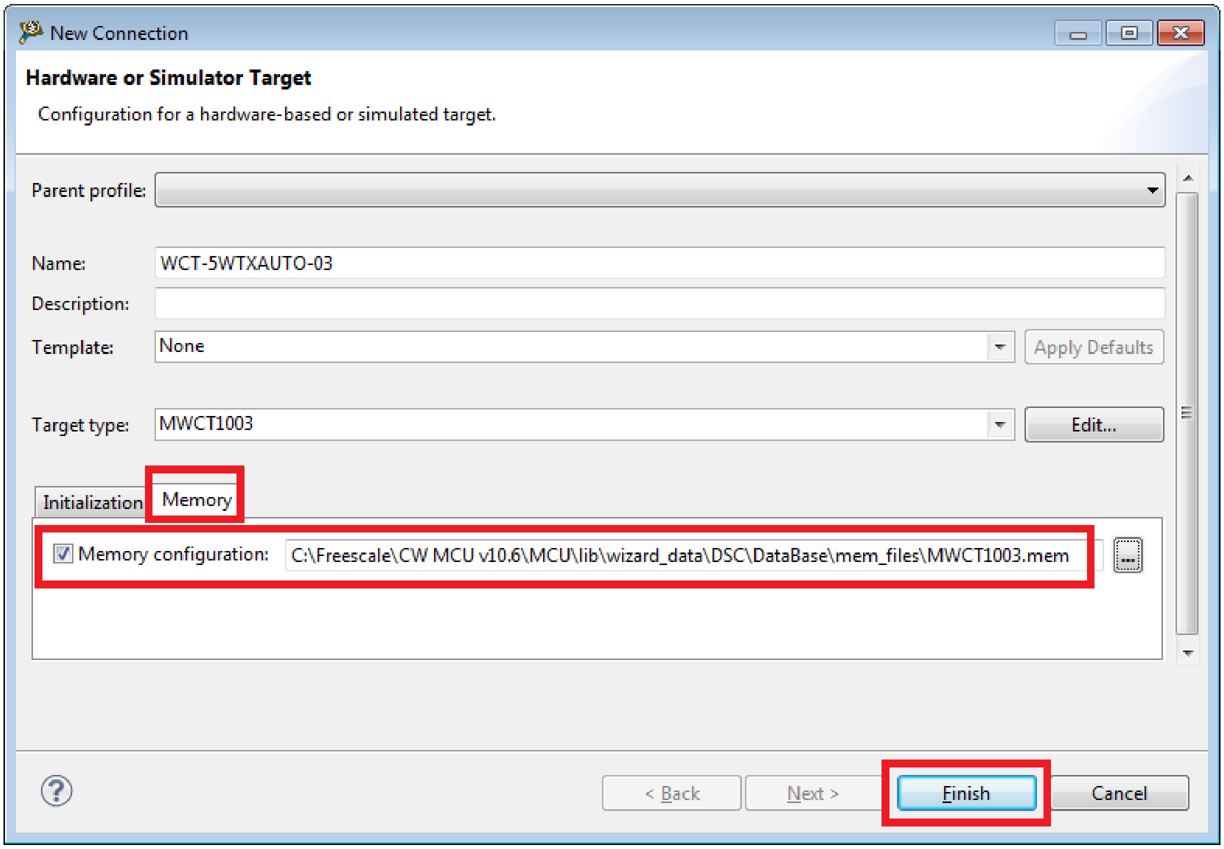


**Figure 39 Memory configuration**

7. Select **USB TAP** for the **Connection type**, and then click **Finish**.

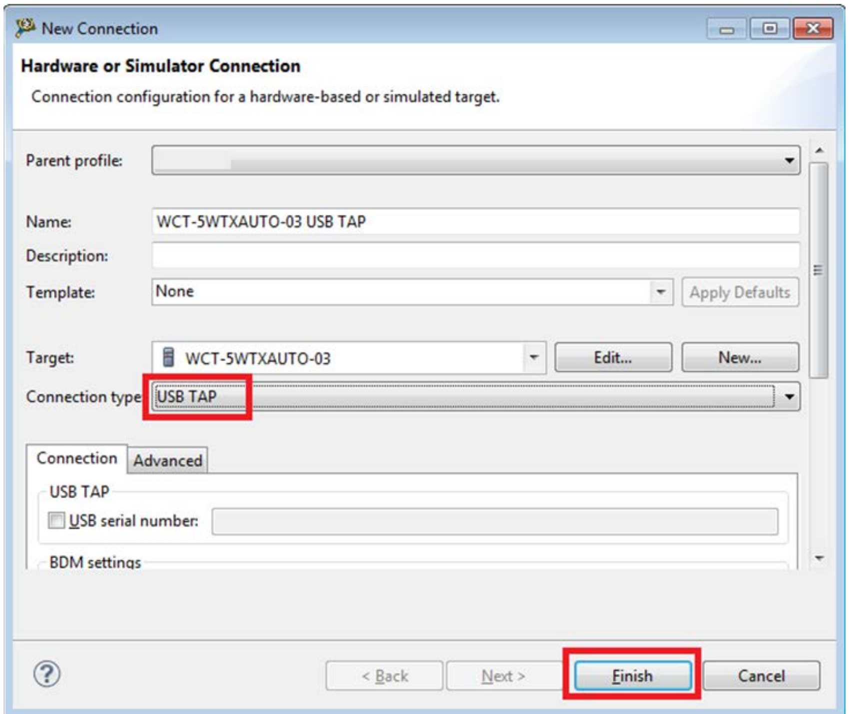


Figure 40 Setting the connection type

8. Set the .S file to be **File to Flash**. Select **Save the Target Task** for future programming. Power on A13 and click **Erase and Program**.

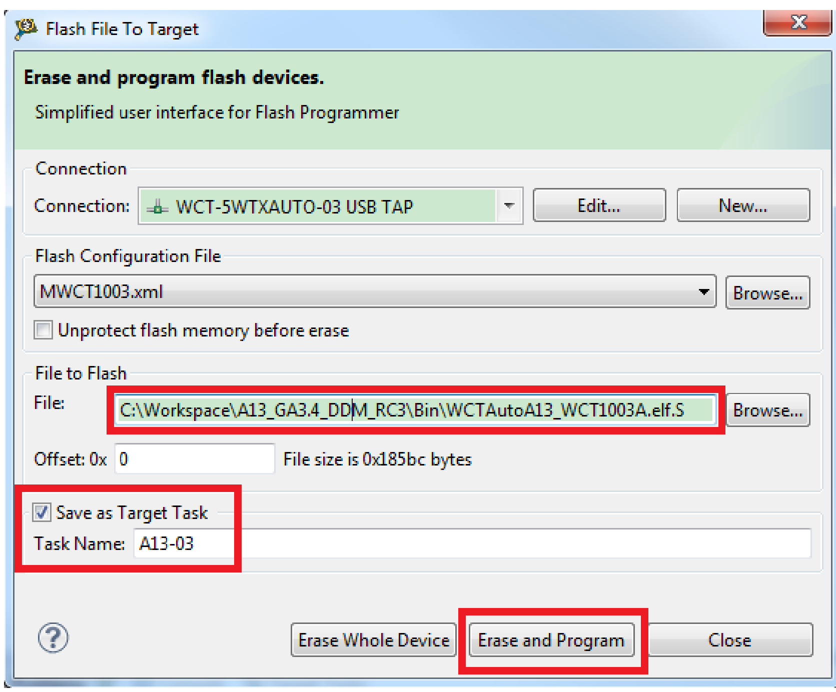


Figure 41 Erase and Program

9. Select the task path to save the task.

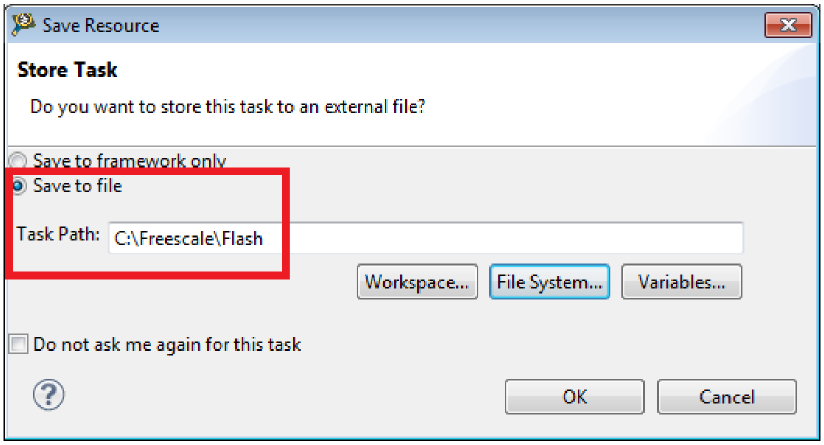


**Figure 42 Selecting the task path**

10. When program is finished, the **Console** window displays the following log.

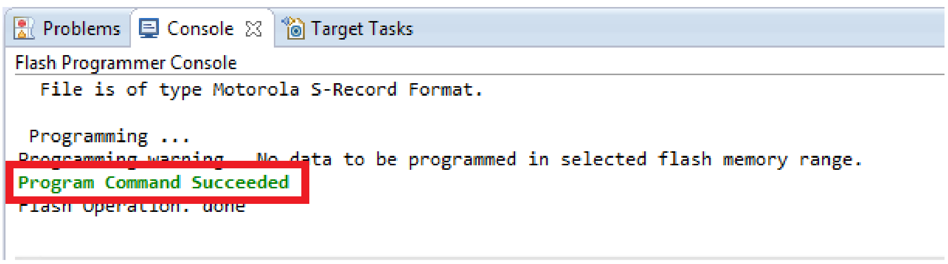


**Figure 43 Programming finished log**

11. For future programming, just select **A13-03** and wait until the programming is finished.

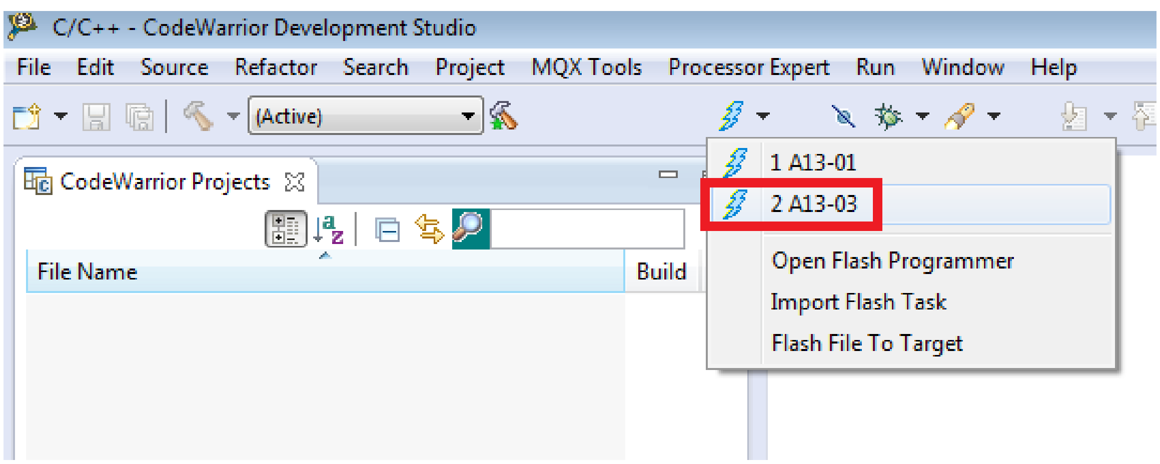


**Figure 44 Future programming**

## 7.5 Board calibration

Freescale provides the FreeMASTER GUI tool for calibration. For board calibration, see the *WCT1001A/WCT1003A Run-Time Debug User's Guide* (WCT100XARTDUG).

# 8. Software Description

## 8.1 Software overview

### 8.1.1 Directory Structure

The following figure shows the directory structure of the whole WCT100xA_A13_V3.x distribution.

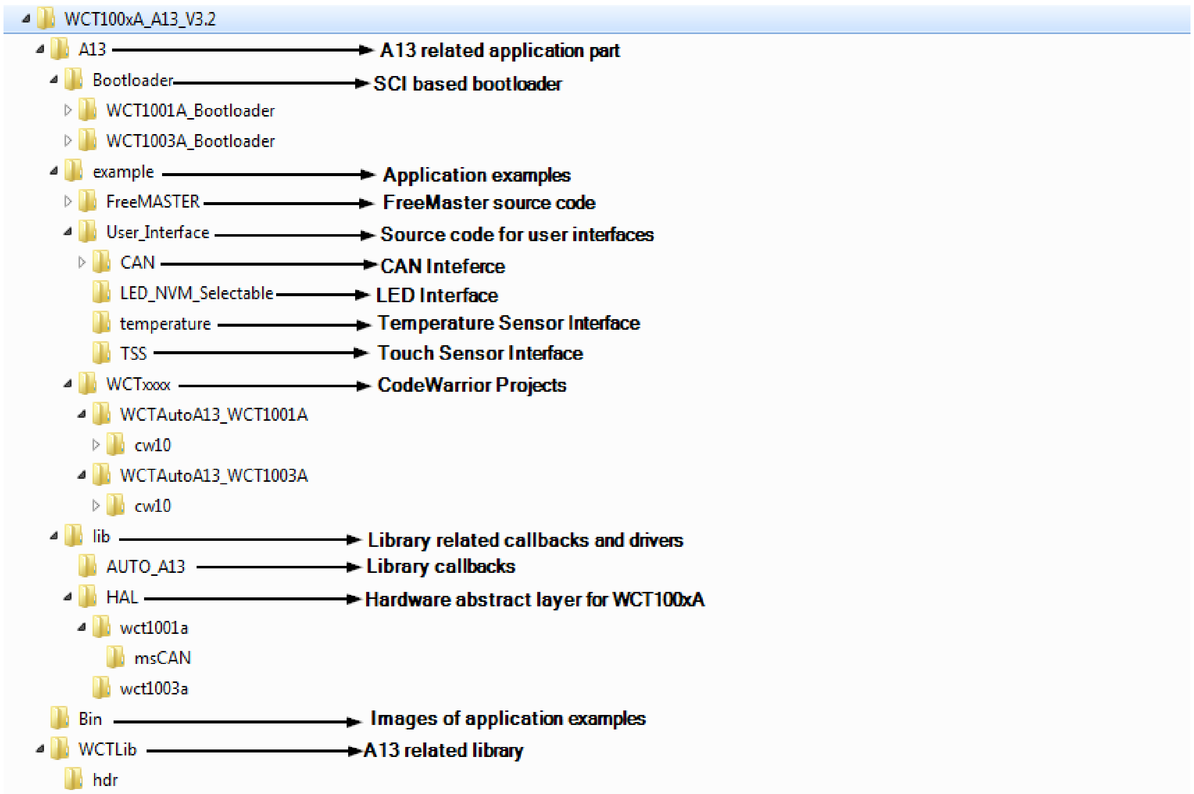


**Figure 45 Directory structure of the whole WCT100xA_A13_V3.x distribution**

### 8.1.2 CodeWarrior Projects

There are four CodeWarrior projects in the package. The following figure shows all the four projects in CodeWarrior GUI when all of them are imported.

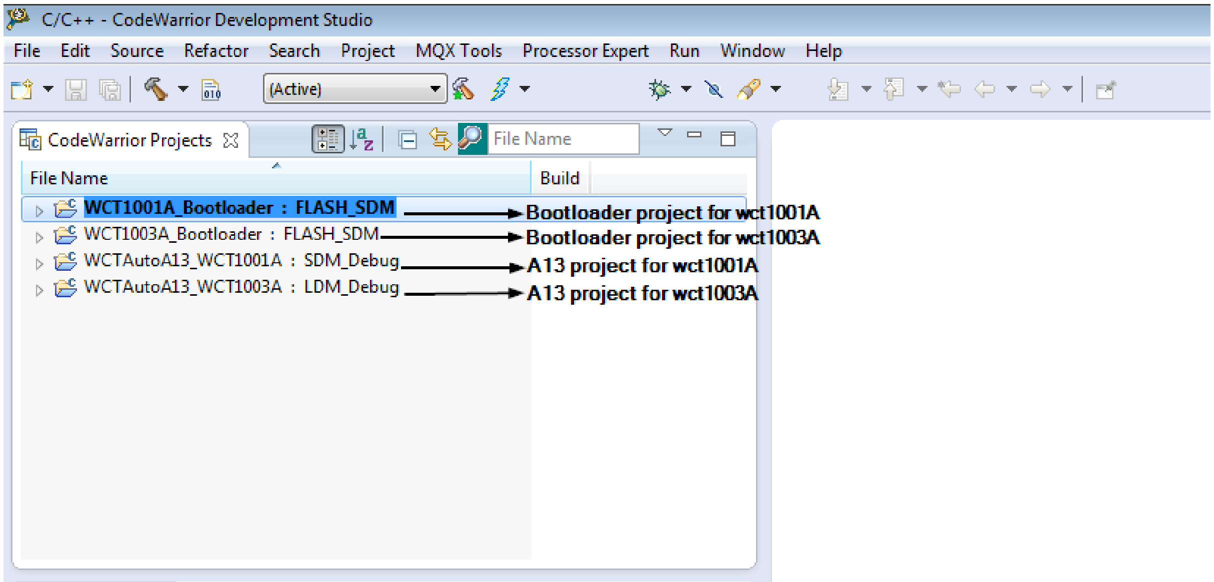


**Figure 49 CW projects**

Combined with different program models and different user cases, multiple build configurations are predefined in respective projects.

There are two program models provided for WCT parts.

- Small Program Model: The compiler generates a more efficient switch table, when the code is in the range 0x0-0xFFFF. This model is more efficient, but the code size is limited to 64K words.
- Larger Program Model: Extends DSP56800E addressing range by providing 24-bit address capability to instructions. That allows user accesses beyond the 64K-word boundary of 16-bit addressing.

For WCT1001A, there are two build configurations:

- SDM_Debug: Small Program Model, including code for debugging.
- SDM_Release: Small Program Model, excluding debugging code to save memory size.

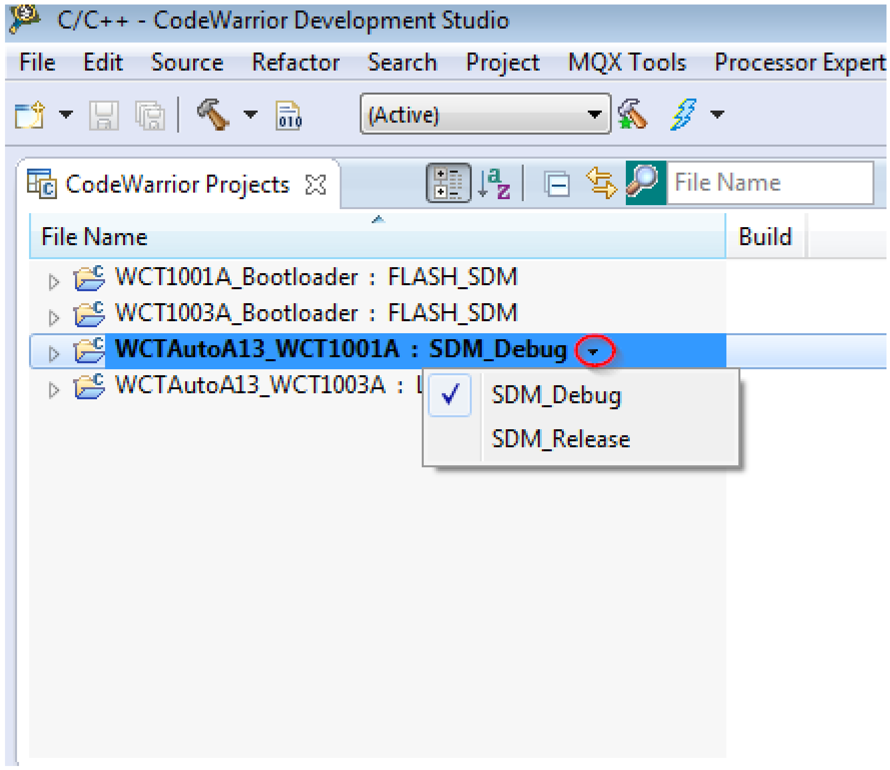


**Figure 46 WCT1001A build configuration**

For WCT1003A, there are three build configurations:

- LDM_Debug: Large Program Model, including code for debugging.
- LDM_Release: Large Program Model, excluding debugging code to save memory size.
- NFC_LDM_Debug: NFC dedicated build configuration. Large Program Model, including code for debugging.

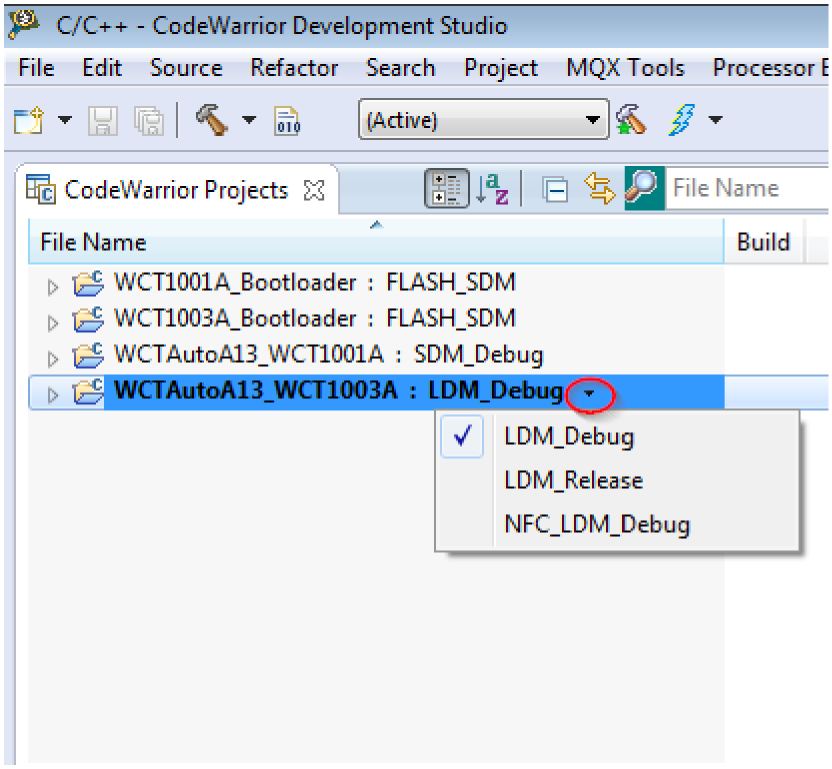


**Figure 47 WCT1003A build configuration**

## 8.2 Functional description

### 8.2.1 Touch function

Two types of touch sensing are supported.

- Using a touch sensor
  - An external touch sensor is used for sensing touch. When touch is detected, the MCU is woken up by the sensor and starts to send out ping signals.
  - This type is only supported by the revision 2 board.
- GPIO touch
  - A GPIO and a timer of MCU are used to sense the capacitance change. The MCU will do this measurement in a specified interval. When the capacitance change exceeds a specified threshold, the MCU starts to send out ping signals.
  - This type is only supported by the revision 3 board.

**Table 2 Touch functions**

| Configurations | Default value | Location | Description |
|---|---|---|---|
| TOUCH_USED | FALSE | application_cfg.h | Enables or disables the function. Set TRUE to enable it. |

| | | | |
|---|---|---|---|
| GPIO_TOUCH | TRUE | application_cfg.h | Selects the GPIO touch method. Set TRUE to select it. |
| MPR_TOUCH | FALSE | application_cfg.h | Selects the touch senor method. Set TRUE to select it. |
| TOUCH_TIMEOUT_MS | 3000 | application_cfg.h | Timeout for touch sensing. The unit is millisecond. During this time, the system is active to communicate with the receiver. If there is not any communication set up during this time, the system will enter standby mode, and touch sensing will start to monitor the touch pad. |
| TOUCH_TIMEOUT_PERIOD_MS | 100 | application_cfg.h | Counting unit time for TOUCH_TIMEOUT_MS. |
| TOUCH_TIMEOUT_CNT | TOUCH_TIMEOUT_MS/TOUCH_TIMEOUT_PERIOD_MS | application_cfg.h | Counting counter for TOUCH_TIMEOUT_MS. |
| GPIO_TOUCH_SENSE_RATE_MS | 500 | application_cfg.h | GPIO touch sensing interval. The unit is millisecond. The system measures the capacitance of the touch pad by this interval. Decreasing this value can increase the touch response rate, but the standby power will also be increased. |

### 8.2.2 LEDs indications

LED configurations are defined in application_cfg.h.

**Table 3 LED configurations**

| Configurations | Default value | Location | Description |
|---|---|---|---|
| USE_ONE_LED_CHANNEL_PER_DEVICE | TRUE | application_cfg.h | Configures the LED channel. When it is set to TRUE, the LED channel is equal to the device number. When it is set to FALSE, the LED channel is equal to the coil number. |
| LED_PORTS | GPIOE_DR,GPIOF_DR | application_cfg.h | Configures the GPIO data register for all the LED ports. |
| LED_BIT_MASKS | 0x01,0x08 | application_cfg.h | Configures the GPIO bit for all the LED ports. |
| LED_PIN_ASSIGNMENT_ACTIVE_HIGH | 0 | application_cfg.h | Defines the polarity for each pin assignment. '1' indicates that high is active, '0' indicates that low is active. |
| NUM_LED_BINS | 1 | application_cfg.h | Defines the number of LED bins. |
| LED_PIN_ASSIGNMENTS | 0 | application_cfg.h | Defines which LED bins use the alternate pin assignment. Set it to '0' for STANDARD, and set it to '1' for ALTERNATE. |

| | | | |
|---|---|---|---|
| LED_CONTROL_VIA_NVM_INIT | 1 | application_cfg.h | Defines which LED bins use NVM to define functionality. Set it to '0' for ROM Table Defined, and set it to '1' for NVM Defined. |
| NVM_LED_MATRIX | - | application_cfg.h | Defines LED functionality by ROM Table. |

Each LED function defined in NVM can be configured by FreeMASTER.

Each LED has two display patterns, ON/OFF and blinking. On the **System Params** tab, configure the state to enable the corresponding display pattern for each LED.

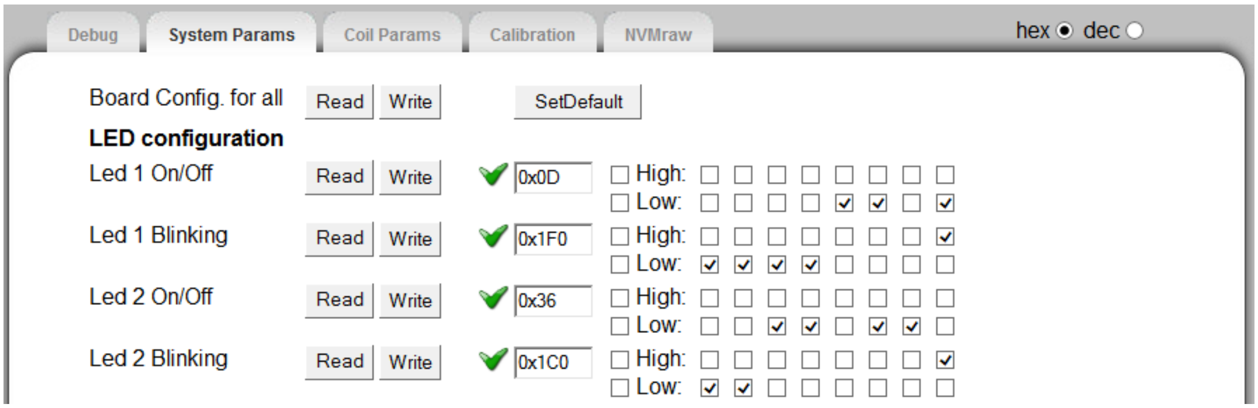


**Figure 48 LED configuration**

Select **High** or **Low**. Several states are displayed. Select the corresponding bit to enable the LED to work in that state.

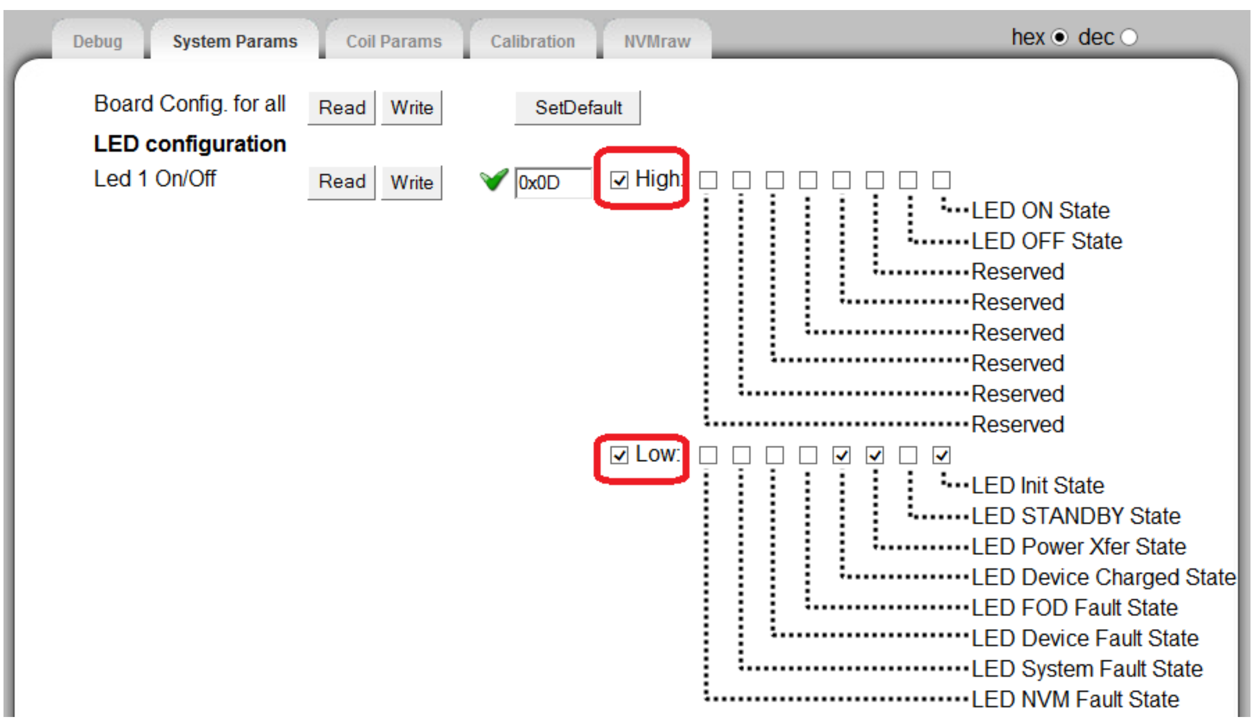


**Figure 49 LED bit configuration**

### 8.2.3 FreeMASTER function

FreeMASTER is supported. The following configuration is used to enable or disable it.

| Configurations | Default value | Location | Description |
|---|---|---|---|
| FREEMASTER_USED | TRUE | application_cfg.h | Enables or disables the function. Set TRUE to enable it. |

### 8.2.4 Low power mode implementation

Low power mode is supported. There are three methods to implement low power mode:

- GPIO touch
  - In the GPIO touch measurement interval, the MCU will enter LPSTOP mode and Vrail will be turned off.
- Touch Senor
  - In this method, the MCU will keep in LPSTOP mode and Vrail will be off until the MCU is woken up by the touch senor.
- Analog ping
  - In analog ping interval and digital ping interval, the MCU will enter LPSTOP mode and Vrail will be turned off.

**Table 4 Low power mode configurations**

| Configurations | Default value | Location | Description |
|---|---|---|---|
| LOW_POWER_MODE_SUPPORTED | FALSE | application_cfg.h | Enables or disables low power mode. Set TRUE to enable it. |
| LOW_POWER_MODE_BY_TOUCH | FALSE | application_cfg.h | Selects TOUCH as low power mode implemented method. Set TRUE to enable it. |
| LOW_POWER_MODE_BY_ANALOG_PING | TRUE | application_cfg.h | Selects Analog Ping as low power mode implemented method. Set TRUE to enable it. |
| ENABLE_NFC_LOW_POWER_MODE | TRUE | application_cfg.h | Enables or disables the low power mode of the NFC chip. Set TRUE to enable it. If the chip is not placed on the board, set it to FALSE. If NFC Stack is used, set it to FALSE. |
| ENABLE_DSC_LOW_POWER_MODE | TRUE | application_cfg.h | Enables or disables the low power mode of the WCT chip. Set TRUE to enable it. |
| VRAIL_RESTORE_TIME_MS | 5 | application_cfg.h | Time for Vrail to restore from OFF to ON. The unit is millisecond. |
| SYSTEM_RESTORE_TIME_MS | 10 | application_cfg.h | Time for the system to restore from STOP to RUN. The unit is millisecond. |

| | | | |
|---|---|---|---|
| ENTERN_LP_MODE_THRESHOLD_MS | 15 | application_cfg.h | Time threshold for the system to enter low power mode. The unit is millisecond. If the idle time is larger than this value, the system will enter low power mode. |

The following table lists the power consumption data of WCT1003A in various low power methods.

**Table 5 Power consumption data**

| Low power method | GPIO touch | Touch sensor | Analog ping |
|---|---|---|---|
| Conditions | 500ms refresh rate;<br>One timer is running in stop mode;<br>NFC in hibernate mode;<br>Rev D Touch pad | All peripherals of MCU are off;<br>NFC in hibernate mode;<br>Rev D Touch pad | 400ms ping interval;<br>NFC in hibernate mode;<br>Plastic Cover |
| Measured Mean Standby Current | 3.6mA@12V | 2.4mA@12V | 17.8mA@12V |

The following table lists the comparison among different low power methods.

**Table 6 Comparison among different low power methods**

| Low power methods | GPIO touch | Touch sensor | Analog ping |
|---|---|---|---|
| Standby Power | High | Low | Highest |
| Cost | Low | High | No extra cost |
| Response when Rx placed | Low | Fast | Fastest |
| Noise | No | No | Some noise is caused by Buck circuit that is switched on/off frequently |

### 8.2.5 Temperature sensor support

Temperature Sensor is supported. The following configuration is used to enable or disable it. The temperature sensor can be used to implement temperature protection for the whole system. When the temperature is higher than a specified threshold, the wireless charging function is disabled. When the temperature falls to a normal range, wireless charging can be restored.

**Table 7 Temperature sensor configurations**

| Configurations | Default value | location | Description |
|---|---|---|---|
| TEMPERATURE_USED | FALSE | application_cfg.h. | Enables or disables the touch function. Set TRUE to enable it. |
| OVERTEMP_WARNING | ADC count at 60℃ | temp_sense.h. | Temperature for warning. |
| OVERTEMP_SHTDN | ADC count at 80℃ | temp_sense.h. | Temperature for shutting charging down. |

| | | | |
|---|---|---|---|
| OVERTEMP_HYSTERSIS | ADC count at 70℃ | temp_sense.h. | The temperature for hysteresis. |

The following figure shows the temperature protection state transitions.

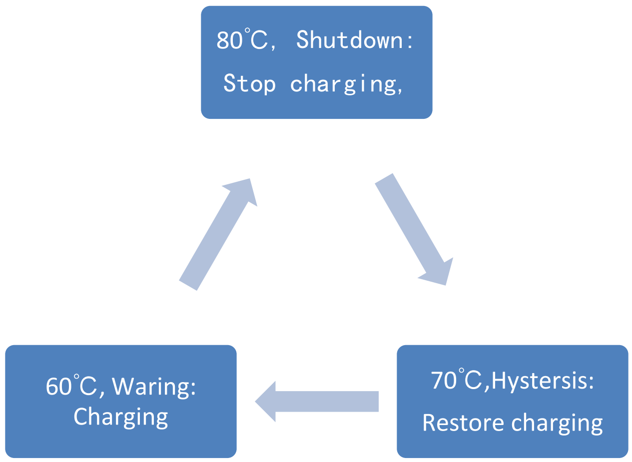


**Figure 50 Temperature protection state transitions**

## 8.3 Protection mechanisms

The following table lists the protections that can be implemented.

**Table 8 Protection mechanisms**

| Protection | Default limits | Description |
|---|---|---|
| Input Power | 12000 mW | Library implemented. If the input power exceeds the limit, charging will be turned off for 5 minutes. The limit value can be changed in NVM parameters. |
| PowerSupplyVoltage | User define | Application implemented. Users can get the input current by calling API: WCT_AnalogIoGetPowerSupplyVoltage(), and then do proper protection. |
| Input Current | User define | Application implemented. Users can get the input current by calling API: WCT_AnalogIoGetInputCurrent(),<br>then do proper protection. |
| Rail Voltage | User define | Application implemented. Users can get rail voltage by calling API: WCT_AnalogIoGetRailVoltage(), and then do proper protection. |
| Temperature | Warning: 60℃<br>Hysteresis: 70℃<br>Shutdown: 80℃ | Application implemented. See Section 8.2.5 |
| FOD | Power Limited: 600 mW | Library implemented. If NPLoss exceeds the limit, charging will be turned off for 5 minutes. The limit value can be changed in NVM parameters. |

# 9. System Bring Up

## 9.1 Ping sequences

When low power mode is disabled and no receiver is placed on the charging surface, the ping sequence is as follows:

Digital ping will appear at about every 5 seconds and the analog ping will appear at about every 400 ms. There will be 12 to 13 analog ping between two digital pings.

The following figures show the PWM waveforms of the ping sequence and ping patterns.

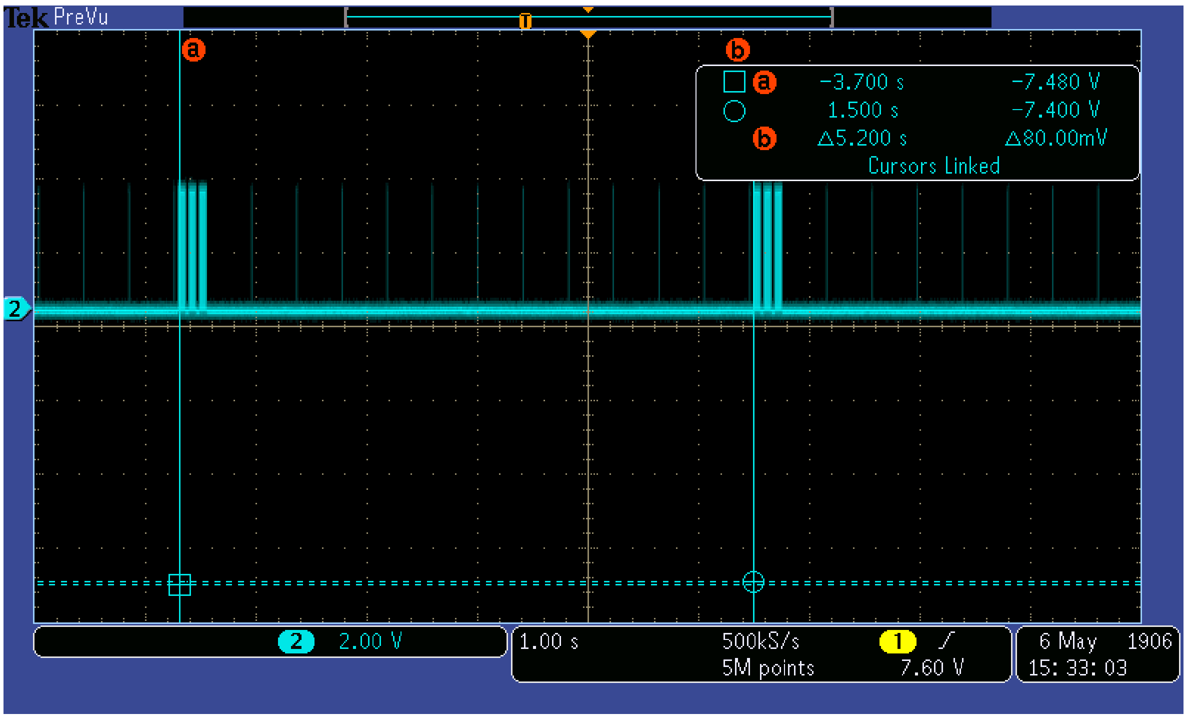


**Figure 51 Digital ping interval**

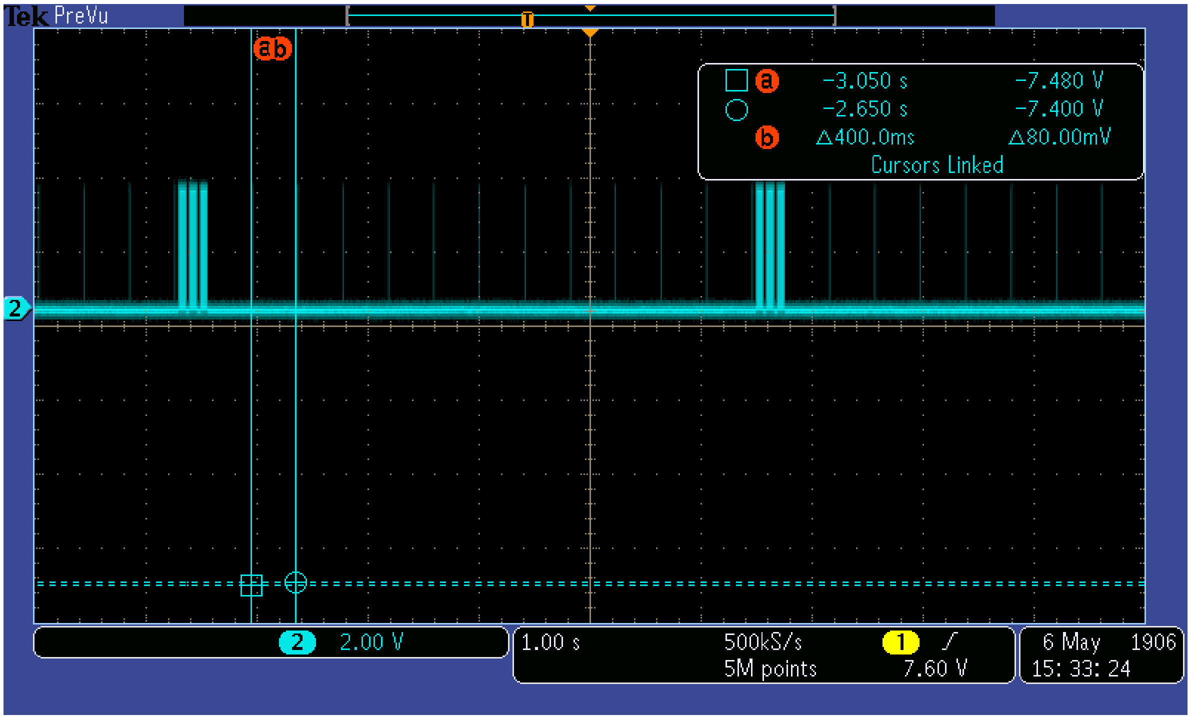


**Figure 52 Analog ping interval**

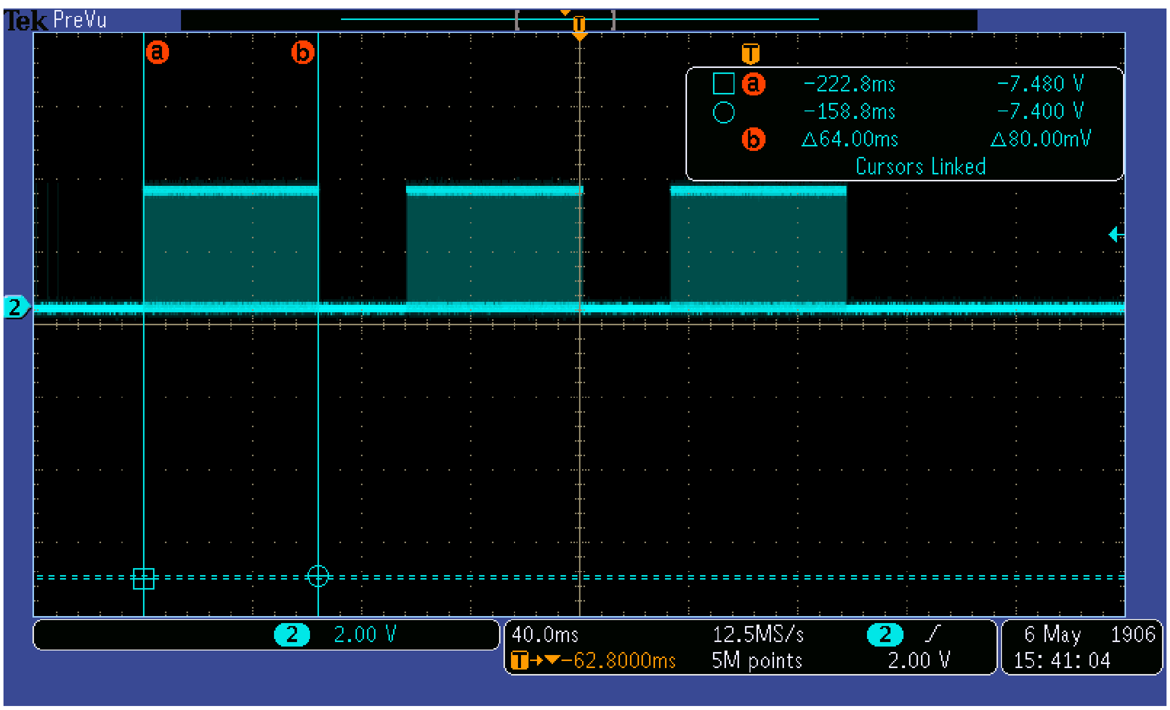


**Figure 53 Digital ping pattern**

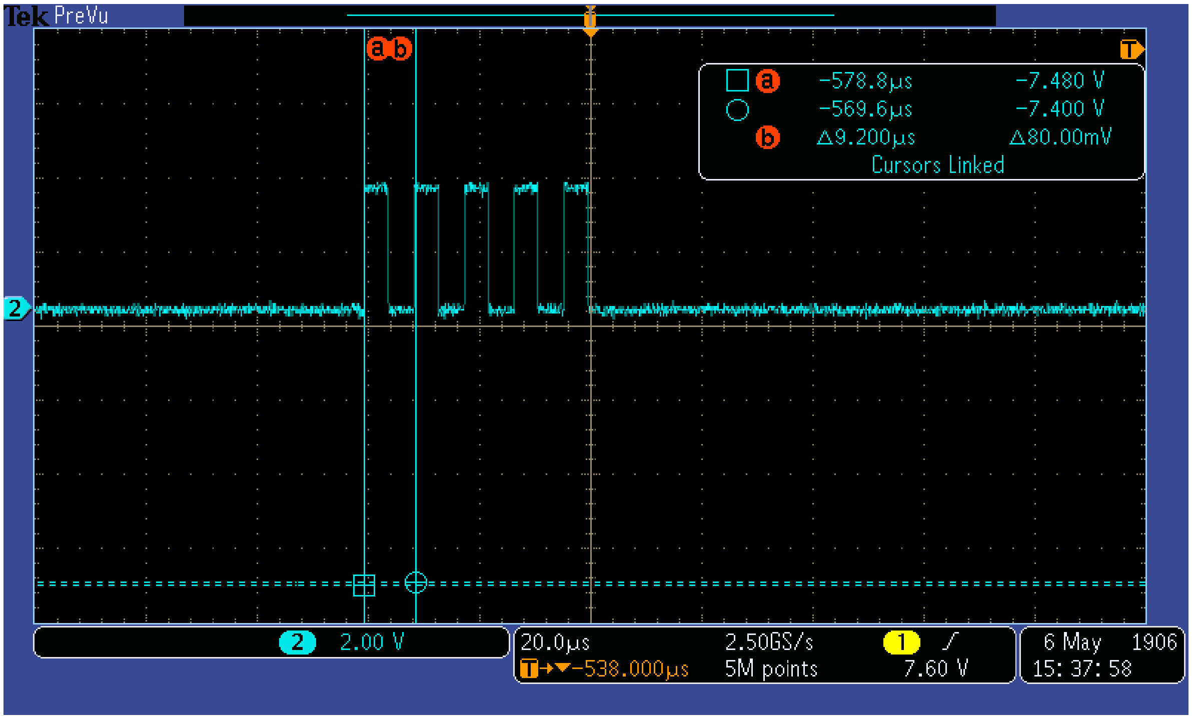


**Figure 54 Analog ping pattern**

## 9.2 Debug messages

The system is able to print messages from a specified SCI port to inform users about what happened in the system. That may be helpful for users to understand the system working procedure and debug issues.

Displays the PMA communication symbols received

**Table 9 PMA communication symbols**

| Value | Symbol | Signal name |
|---|---|---|
| 0 | Reserved | Reserved |
| 1 | S1 | PMA Decrease |
| 2 | S2 | PMA Increase |
| 3 | S3 | PMA NoChange |
| 4 | S4 | PMA EOC |
| 5 | S5 | PMA Msgbit |
| 6 | S6 | TBD |

**Message:** `EPT: <string>`

Displays the end of power transfer reason

**Table 10 End of power transfer reason**

| String | Meaning |
|---|---|
| CHGD | Received EPT packet with reason code: Charge Complete |
| FAULT | Received EPT packet with reason code: Internal Fault |
| OVS | Received EPT packet with reason code: Over Voltage |
| OTS | Received EPT packet with reason code: Over Temperature |
| OVC | Received EPT packet with reason code: Over Current |
| BAT | Received EPT packet with reason code: Battery Failure |
| RECONF | Received EPT packet with reason code: Reconfigure |
| NO RESP | Received EPT packet with reason code: No Response |
| UNKWN | Received EPT packet with reason code: Unknown |

**Message:** `~ <coil>: CEP Timeout`

Prints information when control error packet times out, either because of communication demodulator failed or RX removed

coil: coil ID

**Message:** `~ <coil> : R <rail voltage> D <duty cycle>E <reported error>`

Prints information during PID control

- *coil*: coil ID
- *rail voltage*: active rail voltage value
- *duty cycle*: active duty cycle value
- *reported error*: control error packet value

**Message:** `PWR <coil> <received power>`

Prints the RX received power

- *coil*: coil ID
- *received power*: received power packet value

**Message:** `Max Input Power (<power threshold> mW) Exceeded: <real power> mW`

Prints information when input power protection occurs

- *power threshold*: preset threshold for input power

- *real power*: calculated input power

**Message:** `CHG <coil> <percent>`

Prints RX charge status with information received from the charge status packet

- *coil*: coil ID
- *percent*: percent of power charged at the RX side

**Message:** `PROP <coil> <packet type>`

Prints information proprietary packet type received.

- *coil*: coil ID
- *packet type*: proprietary packet type received

**Message:** `PKT: <type>, DT: <data>`

Prints information received data packets from the Rx.

- *type*: packet type
- *data*: packet data

**How to Reach Us:**

**Home Page:**
www.freescale.com

**Web Support:**
www.freescale.com/support



Document Number: WCT100XAWCAUG
Rev. 3.5
12/2015