CY8CKIT-017 CAN/LIN Expansion Board Kit Guide

Doc. # 001-57814 Rev. *D

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone (USA): 800.858.1810
Phone (Intnl): 408.943.2600
http://www.cypress.com

Copyrights

© Cypress Semiconductor Corporation, 2010-2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

PSoC® Creator™ is a trademark and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Flash Code Protection

Cypress products meet the specifications contained in their particular Cypress PSoC Data Sheets. Cypress believes that its family of PSoC products is one of the most secure families of its kind on the market today, regardless of how they are used. There may be methods, unknown to Cypress, that can breach the code protection features. Any of these methods, to our knowledge, would be dishonest and possibly illegal. Neither Cypress nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Cypress is willing to work with the customer who is concerned about the integrity of their code. Code protection is constantly evolving. We at Cypress are committed to continuously improving the code protection features of our products.

Contents

Safety Information

Regulatory Compliance

The CY8CKIT-017 is intended for use as a development platform for hardware or software in a laboratory environment. The board is an open system design, which does not include a shielded enclosure. This may cause interference to other electrical or electronic devices in close proximity.

In a domestic environment, this product may cause radio interference. In this case, the user may be required to take adequate prevention measures. Also, the board should not be used near any medical equipment or RF devices.

Attaching additional wiring to this product or modifying the product operation from the factory default may affect its performance and cause interference with other apparatus in the immediate vicinity. If such interference is detected, suitable mitigating measures should be taken.

The CY8CKIT-017 as shipped from the factory has been verified to meet with requirements of CE as a Class A product.

The CY8CKIT-017 contains electrostatic discharge (ESD) sensitive devices. Electrostatic charges readily accumulate on the human body and any equipment, and can discharge without detection. Permanent damage may occur on devices subjected to high-energy discharges. Proper ESD precautions are recommended to avoid performance degradation or loss of functionality. Store unused CY8CKIT-017 boards in the protective shipping package.

End-of-Life/Product Recycling

The end of life for this kit is five years from the date of manufacture, mentioned on the back of the box. Contact your nearest recycler for information on how to disposition the kit.

General Safety Instructions

ESD Protection

ESD can damage boards and associated components. Cypress recommends that you perform procedures only at an ESD workstation. If one is not available, use appropriate ESD protection by wearing an antistatic wrist strap attached to chassis ground (any unpainted metal surface) on your board when handling parts.

Handling Boards

CY8CKIT-017 boards are sensitive to ESD. Hold the board only by its edges. After removing the board from its box, place it on a grounded, static free surface. Use a conductive foam pad if available. Do not slide board over any surface.

1. Introduction

The CY8CKIT-017 CAN/LIN Expansion Board Kit (EBK) is an expansion board that is used with the CY8CKIT-001 PSoC® Development Kit (DVK) or the CY8CKIT-030 PSoC 3 Development Kit (DVK). It enables you to evaluate the Controller Area Network (CAN) and Local Interconnect Network (LIN) slave communication capability of PSoC 3 and PSoC 5 devices. You can design your own projects with an easy-to-use CAN and LIN slave components in Cypress's PSoC Creator™ software, or by altering code examples provided with this kit.

1.1 Kit Contents

This kit contains:

- CAN/LIN expansion board
- Quick start guide
- Kit CD

Inspect the contents of the kit; if anything is missing, visit http://www.cypress.com/?rID=40215 or contact Cypress Technical Support.

1.2 Kit Compatibility

This kit contains an expansion board and requires other Cypress kits to use it. It is designed to add CAN and LIN capabilities to the CY8CKIT-001 PSoC Development Kit (DVK). This DVK supports PSoC 1, PSoC 3, and PSoC 5 families. However, it may be necessary to obtain or purchase additional processor modules for the CY8CKIT-001 to develop applications for a particular PSoC device family.

This kit is also compatible with the CY8CKIT-030 PSoC 3 Development Kit. The EBK can be attached to port E of the CY8CKIT-030 DVK. A CY8CKIT-030 kit can generally be substituted for a CY8CKIT-001 kit when using the CY8CKIT-017 kit. Therefore, any information regarding the CY8CKIT-001 kit in this document generally applies to the CY8CKIT-030 kit as well.

The CY8CKIT-017 can also interface with the CY3280-22x45 Universal CapSense Controller (UCC) kit for CY8C2xx45 PSoC 1 devices. This EBK can add LIN capabilities to the UCC kit. However, it does not add CAN capabilities to this kit, because PSoC 1 devices do not have CAN hardware.

CAN and LIN are communication protocols and require more than one CAN or LIN node to set up communication between nodes. Therefore, it is recommended to have two CY8CKIT-001 DVKs and two CY8CKIT-017 EBKs. This enables you to set up CAN or LIN communication between two CAN or LIN nodes. An alternative recommendation is to have a CAN or LIN bus emulator or analyzer. This enables you to emulate a CAN or LIN node to communicate with a PSoC CAN or LIN controller. See sections Using a CAN Bus Analyzer Tool on page 22 and Using a LIN Bus Analyzer Tool on page 22 for more details on using a CAN analyzer or LIN analyzer tool with this kit.

For detailed information about the differences between PSoC 1, PSoC 3, and PSoC 5 devices, go to http://www.cypress.com/psoc. For more information about Cypress' kits, go to the Cypress Store at http://www.cypress.com/shop.

1.3 PSoC Creator

Cypress' PSoC Creator software is a state-of-the-art, easy-to-use software Integrated Development Environment (IDE). It introduces a hardware and software co-design environment based on classical schematic entry and revolutionary embedded design methods.

With PSoC Creator, you can:

- Create and share user defined, custom peripherals using hierarchical schematic design.
- Automatically place and route select components and integrate simple glue logic that is normally present in discrete muxes.
- Trade-off hardware and software design considerations allowing you to focus on what matters and get to market faster.

PSoC Creator also enables you to tap into an entire tools ecosystem with integrated compiler tool chains, RTOS solutions, and production programmers to support both PSoC 3 and PSoC 5.

1.4 Getting Started

To get started, see the Kit Operation chapter on page 13 for a description of the kit operation and how to program the PSoC 3 device. Code examples are used to explain how to use the CAN/LIN expansion board with the CY8CKIT-001 DVK. The Hardware chapter on page 23 provides details of the expansion board hardware. The Code Examples chapter on page 31 guides you to create simple code examples. The Appendix chapter on page 61 provides the schematics and bill of materials (BOM) associated with the expansion board.

1.5 Additional Learning Resources

Visit http://www.cypress.com for additional learning resources in the form of datasheets, technical reference manuals, and application notes.

Application Note AN52701 describes the implementation of CAN bus communication between two PSoC devices. It explains how to send and receive CAN messages and handle error messages.

1.6 Technical Support

If you have any technical questions or issues related to this kit, call Cypress Customer Support

+1 (800) 541-4736 Ext. 8 (in the USA),

+1 (408) 943-2600 Ext. 8 (International),

or visit http://www.cypress.com/go/support

1.7 Document History

Revision	Release Date	Description of Change
**	01/27/2010	Initial version of the guide
*A	02/12/2010	CDT based Updates
*B	02/14/2011	Added hyperlinks in Introduction chapter on page 7. Many minor text edits. Updated to include information about the kit's CD and the CY8CKIT-030 kit.
*C	12/16/2011	Updated images for PSoC Creator version (2.0)
*D	09/11/2012	Added a code example for the LIN component and included updates related to LIN throughout the document. Added Safety Information chapter on page 5 and Regulatory Compliance Information on page 62.

1.8 Document Conventions

Convention	Usage
Courier New	Displays file locations, user entered text, and source code: C:\ ...cd\icc\
Italics	Displays file names and reference documentation: Read about the *sourcefile.hex* file in the *PSoC Designer User Guide*.
[**Bracketed, Bold**]	Displays keyboard commands in procedures: [**Enter**] or [**Ctrl**] [**C**]
File > Open	Represents menu paths: File > Open > New Project
Bold	Displays commands, menu paths, and icon names in procedures: Click the **File** icon and then click **Open**.
Times New Roman	Displays an equation: 2 + 2 = 4
Text in gray boxes	Describes Cautions or unique functionality of the product.

2. Installation

2.1 Prerequisite Software

The CY8CKIT-017 CAN/LIN EBK requires the PSoC Programmer and PSoC Creator software programs to be installed before the kit can be used. If you want to use this EBK to develop applications for PSoC 1 devices, then use the PSoC Designer software. You can install these programs from the CY8CKIT-001 kit CD, this kit's CD, or by downloading them from the following locations:

- PSoC Programmer: http://www.cypress.com/psocprogrammer
- PSoC Creator: http://www.cypress.com/creator
- PSoC Designer: http://www.cypress.com/psocdesigner

2.2 Software Installation

2.2.1 Installation from CD

Follow these steps to install the CY8CKIT-017 CAN/LIN EBK software from the kit's CD:

1. Insert the kit CD into your computer.
2. On the autorun screen that appears, choose the **Install the kit contents from CD** option to install the kit contents from the CD.

   Note If the Autorun screen does not appear, go to **My Computer**, open the drive containing the kit CD. Click **CAN-LIN EBK > Run CANLINEBKSetup.exe**.
3. Follow all on-screen prompts to proceed with the installation. When installing the kit software, the installer checks if the prerequisite software is installed in your system. These include PSoC Creator, PSoC Programmer, Windows Installer, .NET, Acrobat Reader, and Keil Complier. If these applications are not installed, the installer prompts you to download and/or install them.

2.2.2 Installation from Internet

Follow these steps to install the CY8CKIT-017 CAN/LIN EBK software from the internet (this is done to ensure that the latest software is installed):

1. Insert the kit CD into your computer.
2. Choose the **Install the latest kit contents from web** option on the auto run screen. This directs you to http://www.cypress.com/?rID=40215, where you can download the latest installer.
3. Download the installer executable file.
4. Run the installer executable file after it is downloaded.
5. Follow all on-screen prompts to proceed with the installation. When installing the kit software, the installer checks if the prerequisite software is installed in your system. These include PSoC Creator, PSoC Programmer, Windows Installer, .NET, Acrobat Reader, and Keil Complier. If these applications are not installed, the installer prompts you to download and/or install them.

Note All kit contents are found on the kit CD. Also, after the installation is complete, the kit contents are found at the following location:

```
C:\Program Files\Cypress\CY8CKIT-017 CAN-LIN Expansion Board Kit\
```

2.3 Software Uninstallation

Follow these steps to uninstall the CY8CKIT-017 CAN/LIN EBK software:

1. Open the Cypress Update Manager program. This is a program that is installed along with other Cypress software.
2. Click the **Uninstall** button associated with the CY8CKIT-017 kit software.
3. Follow the on-screen prompts to uninstall the software.

The software can also be uninstalled by using the **Add/Remove Programs** tool included with Windows.

2.4 Hardware Installation

Follow these steps to install the hardware:

1. The EBK board must be physically attached to port A of the CY8CKIT-001 DVK.

   Note This document explains how to use this EBK with port A of the CY8CKIT-001 DVK. You can also attach this board to port B or port C of CY8CKIT-001 DVK and port E of CY8CKIT-030 DVK. You need to modify pin assignments of the example projects to use with other ports. See Port Options with CY8CKIT-001 DVK on page 28 for pin assignment details and limitations of using with CY8CKIT-001 DVK ports. See Table 5-1 and Table 5-2 for pin assignment details with port E of CY8CKIT-030 DVK.
2. A MiniProg3 device programmer must be connected to your computer to program this kit's code examples into the PSoC devices. Follow the instructions in the MiniProg3 kit documentation to connect it to your computer.
3. For the CAN example testing:
   a. Connect the two CAN/LIN EBK boards together with a male-to-male, 9-pin, RS-232 cable with "straight-through" connections, as shown in Figure 3-8 on page 17, or
   b. Connect a CAN analyzer to P2 of the EBK.
4. For the LIN example test, connect VCC, LIN bus, and GND of the LIN analyzer to connector P5 of the EBK.

3. Kit Operation

3.1 Introduction

The CY8CKIT-017 CAN/LIN EBK includes two CAN code examples and one LIN code example. The CAN code examples demonstrate two-way CAN communication between two PSoC 3 CAN controller nodes on a CAN bus. These examples must be downloaded into two separate PSoC 3 devices.

- **Code Example 1: CAN_Example_1**

  This project demonstrates sending and receiving of CAN messages. The project sends an 8-bit value on the CAN bus and also receives an 8-bit value from the CAN bus.

- **Code Example 2: CAN_Example_2**

  This project is identical to the 'CAN_Example_1' project, except the CAN message IDs are reversed. Therefore, this project implements a CAN node that can communicate with a PSoC 3 that is programmed with the 'CAN_Example_1' project. See Code Examples on page 31 for more information.

  Note Most of the information related to CAN code examples describes kit operation when two CYC8KIT-001 DVKs and two CY8CKIT-017 EBKs are available. If only one CY8CKIT-001 DVK and one CY8CKIT-017 EBK are available, see Using a CAN Bus Analyzer Tool on page 22 for information on alternative ways of using this kit.

- **Code Example 3: LIN_Example**

  This project demonstrates the LIN slave functionality of PSoC 3. It receives an unconditional frame having eight bytes of data from the LIN bus with frame ID 0x10. The eight bytes are: Byte 1 is the scalar signal of 7-bit length and Byte 2 to 8 are the byte array signal of 7-byte length. This is written to another unconditional frame. The LIN master can read back the data by sending the frame ID 0x11.

  Because the LIN master component is not yet available with PSoC 3, a LIN analyzer is required to test this project. See Using a LIN Bus Analyzer Tool on page 22 for more information.

3.2 Programming PSoC 3 Device

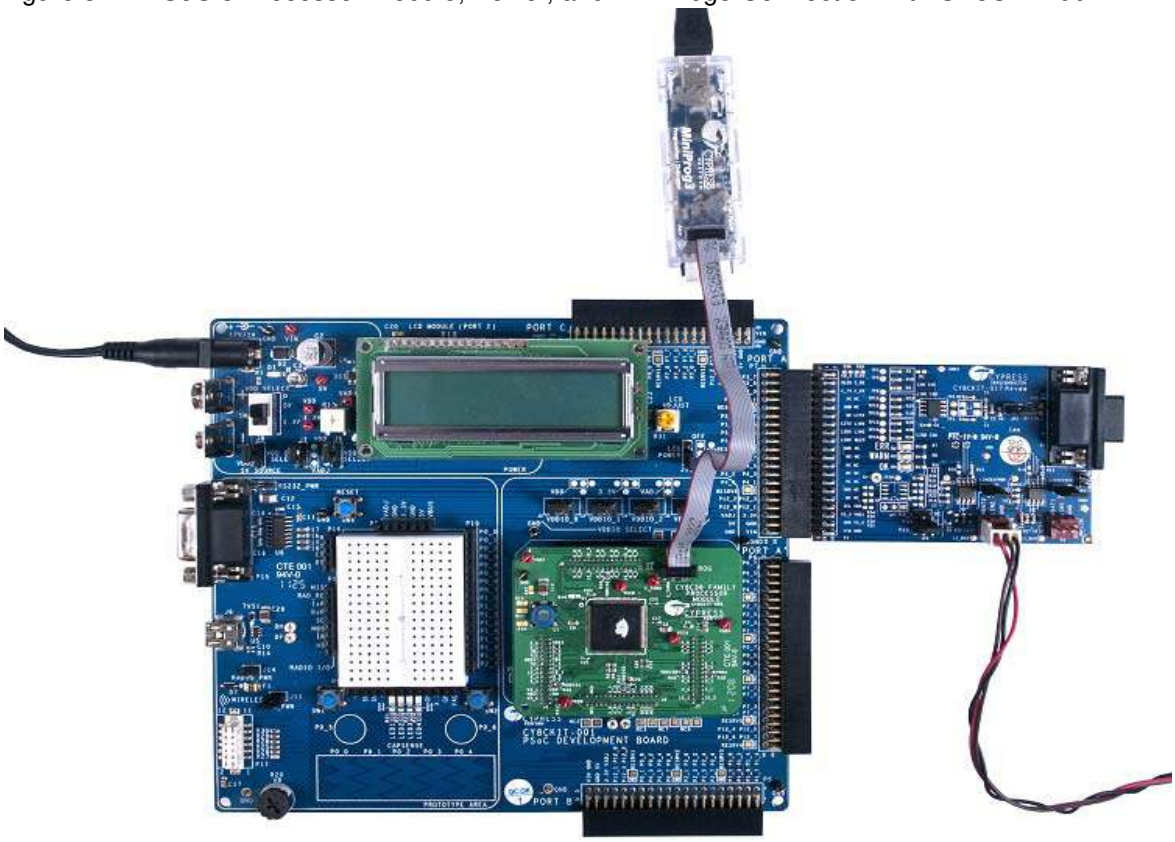
The code examples are provided on the Start Page of PSoC Creator after the CY8CKIT-017 kit contents are installed. This section provides details on programming the PSoC 3 device.

To program the ‘CAN_Example_1’ project to the PSoC 3 silicon, follow these steps:

1. Place the PSoC 3 processor module on the CY8CKIT-001 DVK.
2. Power the DVK using either battery connections or a wall power unit.
3. Connect the MiniProg3 JTAG cable to the JTAG connector, both on MiniProg3 and the PSoC 3 processor module. Connect the MiniProg3 to a host PC USB high speed port using a USB cable.

The connections for steps 1 to 3 are shown in Figure 3-1.

Figure 3-1. PSoC 3 Processor Module, Power, and MiniProg3 Connection with CY8CKIT-001 DVK



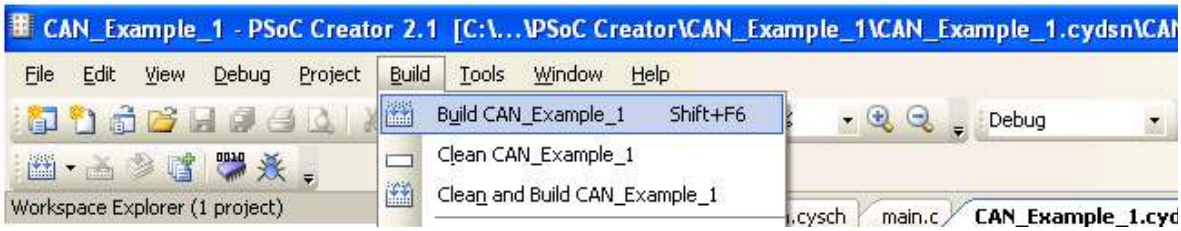
Note See the *PSoC Development Kit Board Guide* for details on connecting and programming PSoC devices.

4. Click on the code example, *CAN_Example_1* located in **Examples and Kits** on the Start Page of PSoC Creator.

Figure 3-2. PSoC Creator Start Page

5. Create a folder in the desired location and click **OK**. The project opens in PSoC Creator and is saved in that folder.
6. Build the project by selecting the **Build** option.

Figure 3-3. Build Project



7. Click the **Program** icon.

Figure 3-4. Program Option

8. The project is programmed successfully.
9. Reset the device by pressing the SW4 switch on the DVK; see Figure 3-5.

Figure 3-5. Reset

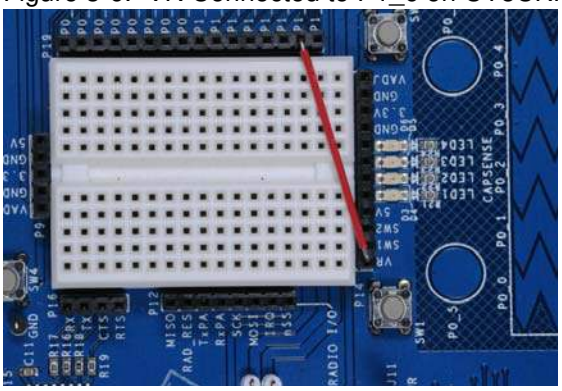
10. Follow steps 1 through 9 to program other code examples, CAN_Example_2 and LIN_Example, on PSoC 3.

3.3 Hardware Connections

3.3.1 CAN Communication Hardware Setup

1. Connect the CAN/LIN expansion board to port A of CY8CKIT-001 DVK, as shown in Figure 3-1.
2. Connect a second CY8CKIT-017 expansion board to port A of a second CY8CKIT-001 DVK, as shown in Figure 3-1.
3. On both CY8CKIT-001 DVK boards, connect the analog input from the potentiometer (VR slot in CY8CKIT-001 DVK) to the P1_6 on the DVK, as shown in Figure 3-6.

Figure 3-6. VR Connected to P1_6 on CY8CKIT-001 DVK

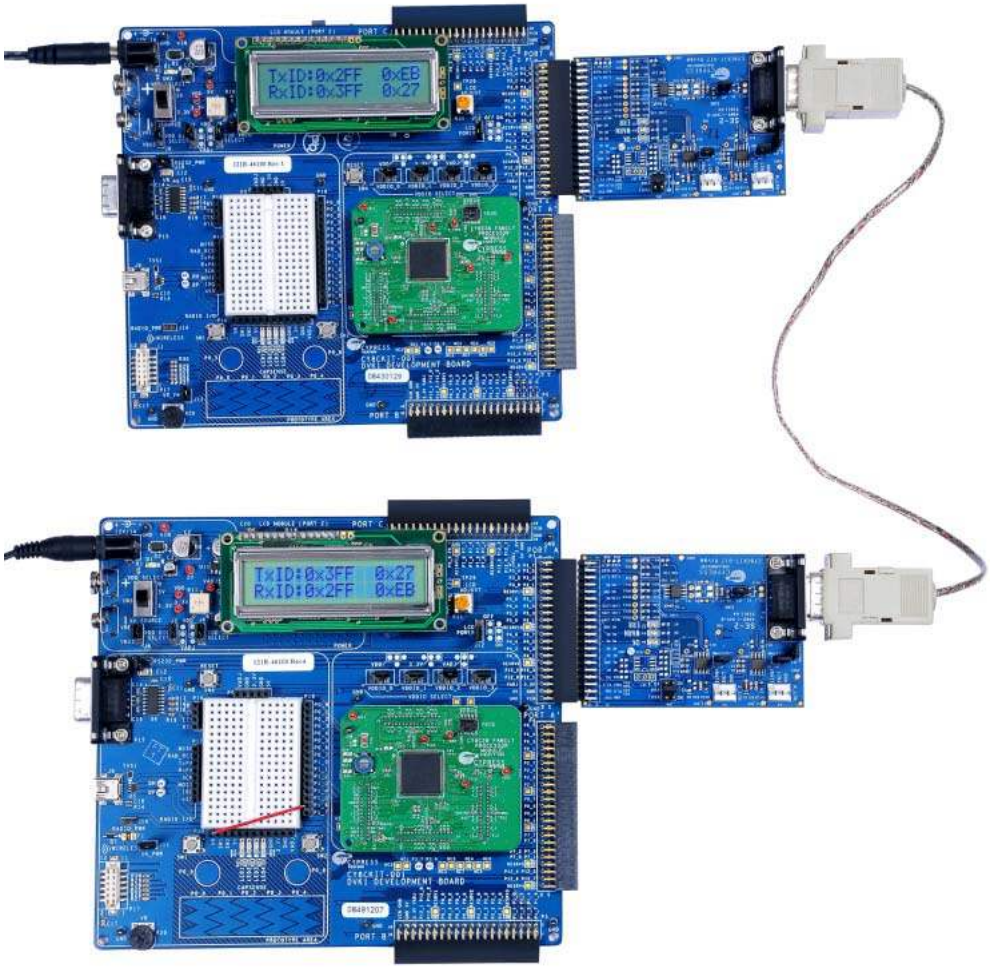


4. On both DVK boards, power the VR by setting the jumper J11 to ON position.

Figure 3-7. Jumper J11 to ON position on CY8CKIT-001 DVK

5. The remaining jumper settings on both DVKs should be in the default state. See the *PSoC Development Kit Board Guide* for the default setting of jumpers.
6. Connect the two CAN/LIN EBK boards together with a male-to-male, 9-pin, RS-232 cable with "straight-through" connections, as shown in Figure 3-8. Connect the cable to the CAN DB9 connector (P2) on both EBK boards.

Figure 3-8. Connected CAN/LIN EBK Boards

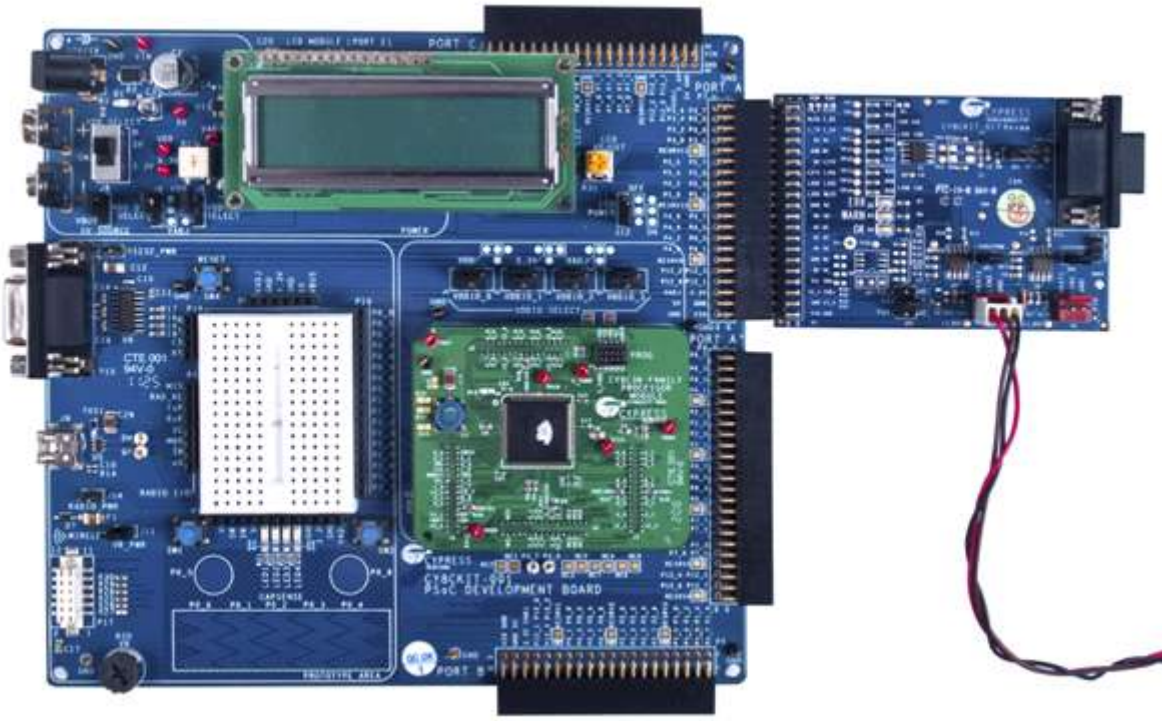


7. Power up one DVK board with the 12-V power supply. Then, power up the other DVK with a 12-V power supply. The second DVK must be powered up within approximately 5 seconds of powering the first DVK.

3.3.2 LIN Communication Hardware Setup

1. Connect the CAN/LIN expansion board to port A of CY8CKIT-001 DVK, as shown in Figure 3-9.

Figure 3-9. EBK to Port A of CY8CKIT-001 DVK

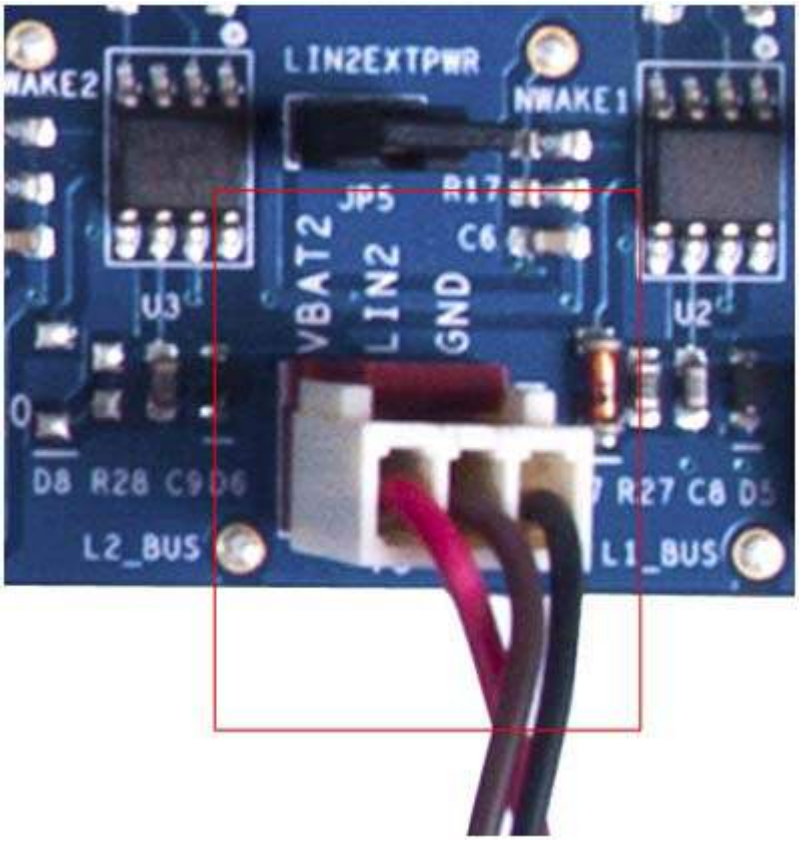


2. Set VDD SELECT switch (SW3) on the DVK to the 5 V position. The remaining jumper settings on the DVK must be set to or left at the default state.

Figure 3-10. VDD Select

3. Connect Vbat, LIN bus, and GND of the LIN analyzer/LIN master to S_LIN (P5) connector of EBK. See LIN Bus Connectors on page 26 for details of LIN bus connectors.

Figure 3-11. Connect to S_LIN



4. Connect a 12-V power supply adapter, which is supplied along with the CY8CKIT-001 DVK to the power jack of the DVK.

Note If you are using the CY8CKIT-030 PSoC 3 DVK, connect the CY8CKIT-017 EBK to port E.

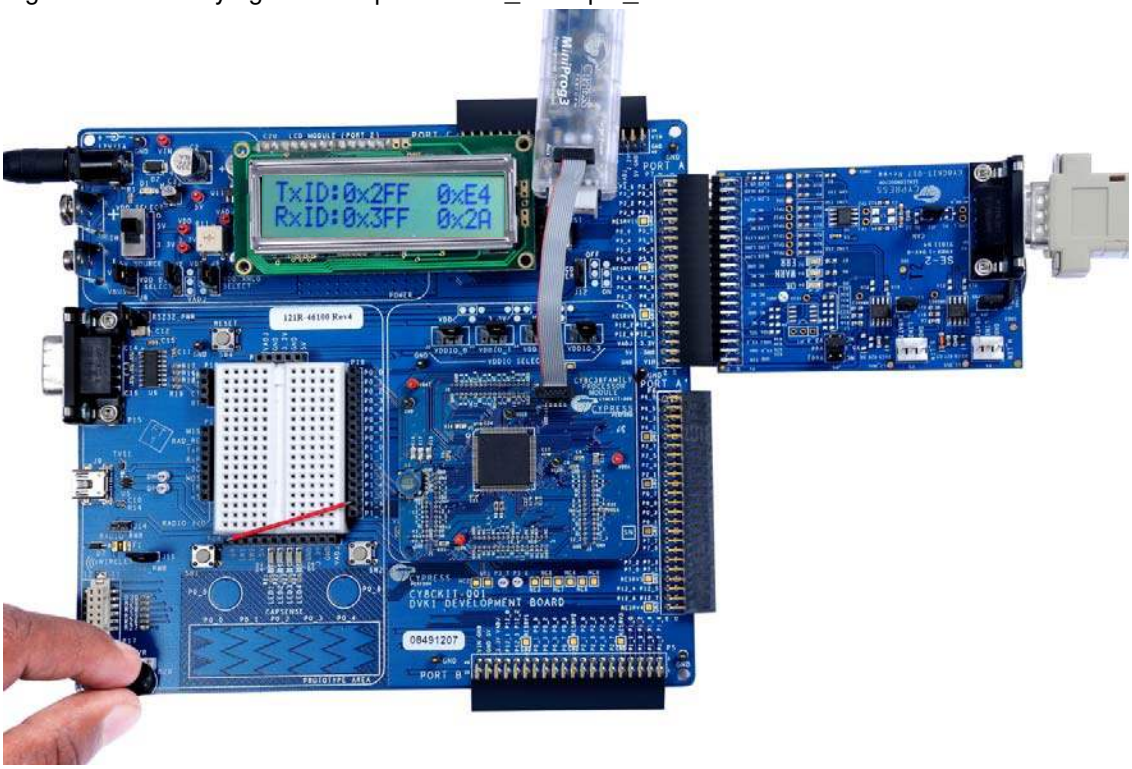
3.4 Verify Functionality

3.4.1 CAN Communication

Vary the VR (potentiometer on either DVK) and note the change in status displayed on the LCD of either DVK.

Note The CAN communication may not work correctly if the PSoC devices are not using an accurate, external clock source. See CAN Bus Clock Accuracy on page 24 for details on oscillator requirements.

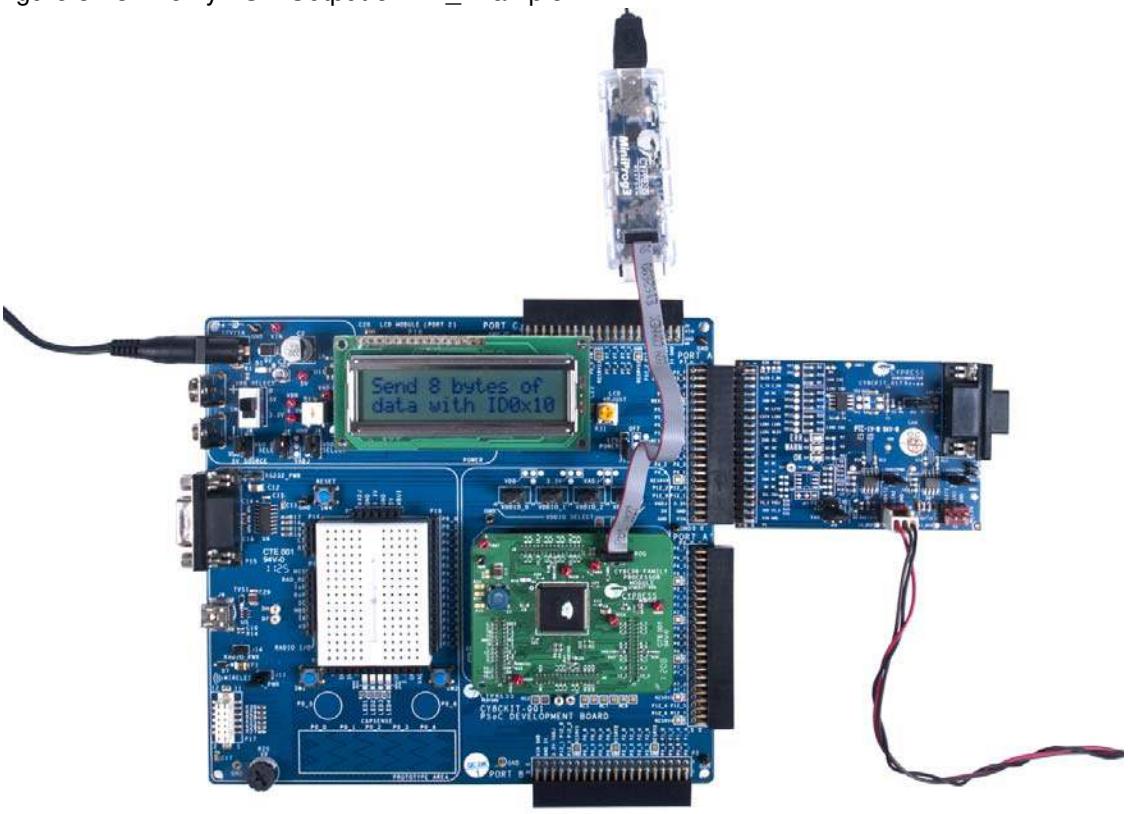
Figure 3-12. Verifying LCD Output of CAN_Example_1



3.4.2 LIN Communication

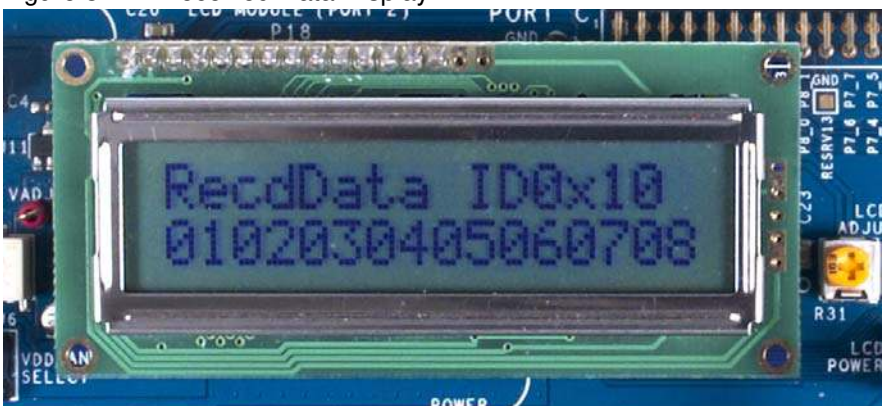
1. When the LIN_Example project starts, the LIN Slave component is initialized. If the startup is successful, the ‘LINS Initialize Successful’ message is displayed on the LCD, followed by ‘Send the 8 bytes of data with ID 0x10’.

Figure 3-13. Verify LCD Output of LIN_Example



2. Configure the LIN analyzer/master with checksum set to **Enhanced** and baud rate set to **19200**.
3. Send an unconditional frame having eight bytes of data (Byte 1 is the scalar signal of 7-bit length and Byte 2 to 8 is byte array signal of 7-byte length) with a frame ID of 0x10 from the LIN analyzer/master. The frame ID and values of data received are displayed after 'Recd-Data' on the LCD. This data will be displayed for 12 seconds.

Figure 3-14. Received Data Display



4. After 12 seconds, the LCD display changes to 'Send ID 0x11 to read back data'. Send a frame with an ID of 0x11 from the LIN analyzer/master.
5. When the frame with ID 0x11 is received, the data is sent to the LIN master. The frame ID and values of sent data is displayed on the LCD for 12 seconds.

Figure 3-15. Sent Data Display

6. After 12 seconds, the LCD display again switches to "Send the 8 bytes of data with ID 0x10". Repeat steps 3 to 5 to check LIN slave communication with a different set of data.

LIN Slave Communication Diagram

Protected Identifier	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Checksum (Enhanced)
0x50	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x8B
0x11	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0xCA
0x50	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	0x1A	0x1B	0x2F
0x11	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	0x1A	0x1B	0x6E

Legend:

Data published by LIN Master
Data published by LIN Slave

3.5 Using a CAN Bus Analyzer Tool

This kit functions most effectively when two CY8CKIT-001 DVKs and two CY8CKIT-017 EBKs are available. However, it is also possible to replace one CY8CKIT-001 DVK and one CY8CKIT-017 EBK with a CAN bus analyzer or emulator tool. It is even possible to use any other CAN node to communicate with this kit.

If you use a CAN bus analyzer or emulator tool to communicate with this kit, then the tool must be set up to send and receive CAN messages (at proper intervals) with proper length and message ID and at a proper baud rate. See the Code Examples chapter on page 31 for more details on the CAN controller configuration of this kit's code examples.

If you use any other CAN node to communicate with this kit, then you may need to modify the firmware to allow communication. You can modify the code examples, firmware, or settings of the other CAN node.

3.6 Using a LIN Bus Analyzer Tool

The LIN_Example project demonstrates functionality of the LIN slave device, so the LIN master device must be also used. The LIN bus analyzer or emulator tool can be used as the LIN master device. It is also possible to use any other LIN master node to communicate with this example project. If you use a LIN bus analyzer or emulator tool, then the tool must be set up to send and receive frames with proper length and ID and at a proper baud rate. See the Code Examples chapter on page 31 for more details on the LIN slave component configuration of this kit example project.

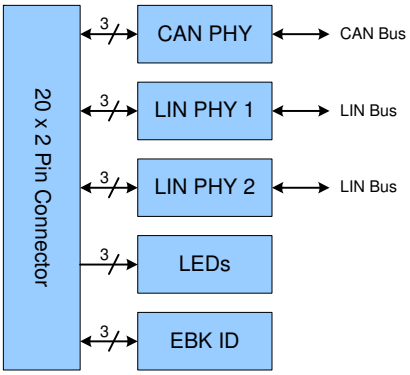
4. Hardware

4.1 System Block Diagram

The CAN/LIN EBK hardware consists of the following functional blocks:

- CAN transceiver circuit (TJA1050)
- LIN transceiver circuit in master configuration (TJA1020)
- LIN transceiver circuit in slave configuration (TJA1020)
- Three indicator LEDs
- EBK identification circuit (not populated)
- 40-pin (2x20) connector (Sullins Connector Solutions, S2111E-20-ND)

Figure 4-1. EBK Block Diagram



The primary functional blocks of this EBK are the three transceiver circuits: one CAN transceiver circuit and two LIN transceiver circuits. Each of these transceiver circuits enables a digital CMOS PSoC device to interface with a physical CAN or LIN bus, respectively. Without these transceiver circuits, it is impossible for CMOS devices to communicate with other CAN or LIN nodes on a CAN or LIN bus.

The EBK ID functional block consists of two circuits that are not populated, and can therefore be ignored. These circuits exist to provide forward compatibility with automatic EBK identification features that do not yet exist.

The LEDs functional block consists of three, active-low LEDs that can provide indications. These LEDs are driven by PSoC pins.

The 40-pin (2x20) connector connects the configured PSoC I/O pins to the various circuits on the expansion board.

4.2 CAN Physical Layer Transceiver Circuit

The CAN physical layer transceiver circuit on the expansion board uses a TJA1050 CAN transceiver device. This device translates differential CAN bus signals to and from digital CMOS signals with standard TTL voltage levels.

Three signals are used between this circuit and the PSoC controller. These signals are CAN_RX, CAN_TX, and CAN_EN. Any data signals on the CAN bus are driven at the CAN_RX signal. Any data signals driven on the CAN_TX signal is driven onto the CAN bus. The CAN_EN signal enables and disables the CAN transceiver.

A pull-up resistor footprint (R1) is provided on the CAN_EN net. This can be populated if it is desired to have a pull-up resistor on the CAN_EN net.

See the TJA1050 device datasheet for details on each of these three pins of the CAN transceiver.

4.2.1 CAN Bus Clock Accuracy

For accurate CAN communication, a CAN controller device must typically have a clock source with a frequency tolerance of 0.5% or less. Therefore, the PSoC device used as the CAN controller must meet this requirement. If the native internal oscillator tolerance of the PSoC device is greater than 0.5%, then some external clock source that is more accurate must be used. For example, an external oscillator or an external crystal can be used with the PSoC to improve the accuracy of the clocks in the PSoC device.

Footprints for oscillator and crystal devices (and any necessary passive components) are provided on the CY8CKIT-001 DVK board and PSoC processor module boards. Newer versions of these boards already have these footprints populated with components. If you have an older PSoC processor module that does not have a populated crystal circuit, you can populate it yourself or contact Cypress Technical Support for assistance.

See Design Wide Resources on page 48 for details on the clock configuration of this kit's code examples.

4.2.2 CAN Bus Connector

The following table shows the pinout of the CAN DB9 connector (P2) on the expansion board.

Table 4-1. CAN Connector Pinout

Pin	Signal
1	NC
2	CAN_L
3	GND
4	NC
5	NC
6	GND
7	CAN_H
8	NC
9	NC (VIN)

By default, pin 9 of the CAN connector is left floating. However, if the CANEXTPWR jumper (JP3) is populated, pin 9 of the CAN connector is connected to the VIN power rail of the DVK and EBK. This

is useful if you want to power up other CAN nodes through the CAN connector or if you want to power up the DVK and EBK from the power supply of some other CAN node.

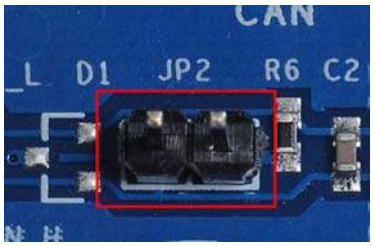
Warning: Take extra care when populating the CANEXTPWR jumper. The DVK or EBK can be damaged if the DVK and EBK have their own power supply powering the VIN rail and a different power supply is connected through pin 9 of the CAN connector.

4.2.3 CAN Bus Termination

The CAN specification requires that CAN nodes located physically at the end of the CAN bus must terminate the CAN differential signals with 120 Ω. The EBK features a 120-Ω termination resistor (R6) that can be enabled or disabled by using jumper JP2. If JP2 is populated, the termination resistor is active. If JP2 is not populated, the termination resistor is not active.

The termination resistor is active (JP2 is populated) by default.

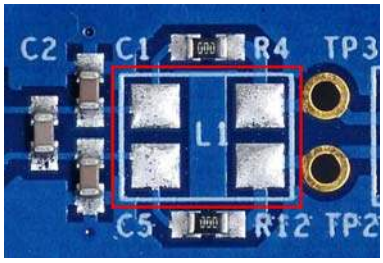
Figure 4-2. JP2 Jumper



4.2.4 Choke Footprint

A footprint for a common-mode signal suppression choke is available on the expansion board, but it is not populated. This footprint can be populated with a B82789C0 (or equivalent) choke to suppress common-mode signals on the CAN bus. If a choke component is mounted on the L1 footprint, resistors R4 and R12 must be removed from the board. The choke component has no effect if these two resistors are not removed.

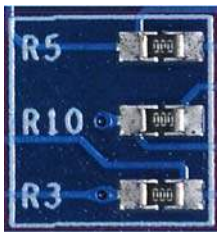
Figure 4-3. CAN Choke Footprint



4.2.5 CAN Circuit Isolation

The CAN circuit can be completely isolated from the rest of the CY8CKIT-001 DVK by removing resistors R3, R5, and R10. Each of these resistors is 0 Ω. This is useful if the CAN circuit is not needed, but other circuits on the EBK are. In this case, isolating the CAN circuit ensures that the CAN circuit does not interfere with any other circuits that share the same pins. The footprints for R3, R5, and R10 are designed so that they can be reconnected easily with a ‘solder jumper’ instead of repopulating the footprints with a 0-Ω resistor.

Figure 4-4. CAN Circuit



4.3 LIN Physical Layer Transceiver Circuits

The two LIN physical layer transceiver circuits on the expansion board use a TJA1020 LIN transceiver device. This device translates high-voltage LIN bus signals to and from digital CMOS signals with standard TTL voltage levels.

Three signals are used between each LIN circuit and the PSoC controller. For the LIN1 circuit, the three signals are LIN1_RX, LIN1_TX, and LIN1_NSLP. For the LIN2 circuit, the signals are LIN2_RX, LIN2_TX, and LIN2_NSLP. Any data signals on the LIN bus are sent to the PSoC by the LINx_RX signal. Any data signals driven by the PSoC on the LINx_TX signal will be driven onto the LIN bus. The LINx_NSLP signals are used to put the LIN transceiver device in and out of sleep.

By default, there are no pull-up resistors on the LINx_TX signals. A pull-up resistor footprint is provided (R14 on LIN1_TX and R16 on LIN2_TX) on each LINx_TX signal if you want to have a pull-up resistor on this signal.

See the TJA1020 device datasheet for details on each of the three signals of the LIN transceiver.

4.3.1 LIN Bus Connectors

The following table shows the pinout of both of the 3-pin LIN connectors (P4 and P5) on the expansion board.

Table 4-2. LIN Connector Pinouts

Pin	LIN1	LIN2
1	NC (VBAT)	NC (VBAT)
2	LIN_BUS	LIN_BUS
3	GND	GND

By default, pin 1 of both LIN connectors (P4 and P5) are left floating. However, if the LIN1EXTPWR jumper (JP4) is populated, pin 1 of the LIN1 connector is connected to the VIN power rail of the DVK and EBK. If the LIN2EXTPWR jumper (JP5) is populated, pin 1 of the LIN2 connector is connected to the VIN power rail of the DVK and EBK. Populating either one of these jumpers is useful if you want to power up other LIN nodes through the either of the LIN connectors, or if you want to power up the DVK and EBK from the power supply of some other LIN node.

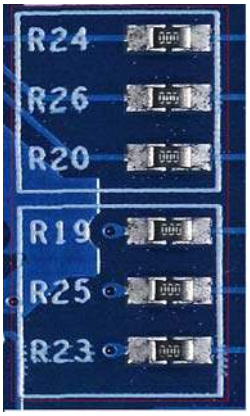
Warning: Take extra care when populating the LIN1EXTPWER or LIN2EXTPWR jumpers. The DVK or EBK can be damaged if the DVK and EBK are powered by a power supply at the VIN rail and a different power supply is connected through pin 1 of either LIN connector.

4.3.2 LIN Circuit Isolation

Both LIN circuits can be completely isolated from the rest of the CY8CKIT-001 DVK by removing resistors R19, R23, and R25 (to isolate the LIN1 circuit) or R20, R24, and R26 (to isolate the LIN2

circuit). Each of these resistors is 0 Ω. This is useful if either of the LIN circuits are not needed, but other circuits on the EBK are. In this case, isolating a LIN circuit ensures that the LIN circuit does not interfere with any other circuits that share the same pins. The footprints for R19, R23, R25, R20, R24, and R26 are designed so that they can be reconnected easily with a 'solder jumper' instead of repopulating the footprints with a 0-Ω resistor.

Figure 4-5. LIN Circuit



4.3.3 Using the LIN Transceiver NWAKE Pins

Both TJA1020 LIN transceiver devices have an NWAKE input that can be used to wake up a sleeping LIN bus without interaction from the PSoC LIN controller. If the NWAKE input needs to be used for either LIN circuit, the signal can be accessed using the test points on the board labeled with NWAKE1 and NWAKE2, respectively.

4.3.4 LIN Master and Slave Configurations

By default, the LIN1 circuit has a LIN master configuration and the LIN2 circuit has a LIN slave configuration. Table 4-3 shows how footprints in each circuit are populated by default to make a master or slave configuration. The components can be replaced as shown by Table 4-3 to change the circuit from a master to slave or vice versa.

Table 4-3. Configurations

Circuit	Diode	Resistor	Capacitor
LIN1 (master)	D7 is PMLL4148L	R27 is 1 kΩ	C8 is 1 nF
LIN2 (slave)	D8 is not populated	R28 is not populated	C9 is 220 pF

4.4 Indicator LEDs

The expansion board has three indicator LEDs: red, amber, and green. The LEDs are active-low, and they must each be driven with a sink current of approximately 2 mA to turn them on.

All three LEDs can be completely isolated from the rest of the CY8CKIT-001 DVK by removing resistor R2, which is a 0-Ω resistor. This is useful if the LEDs are not needed, but other circuits on the EBK are. In this case, isolating the LEDs ensures that they do not interfere with other circuits that share the same pins.

4.5 Port Options with CY8CKIT-001 DVK

The CAN/LIN expansion board connects to the CY8CKIT-001 PSoC DVK through the 20×2-pin connector. It hooks up to the DVK through one of the following ports: port A, port B, or port C. Table 4-4 shows how the signals on the CAN/LIN expansion board map to the pins on port A, port B, or port C of the DVK board. Sections 4.5.1 Jumper Settings of CY8CKIT-001 DVK for Using Port B and 4.5.2 Debugging Restrictions When Using Port B explain the limitations of using the EBK with port B.

Table 4-4. Port Pin Connections

Pin	Port A	Port B	Port C	CAN/LIN EBK
1	P3_7	P1_7	P9_7	EBK_SEL
2	P3_6	P1_6	P9_6	ERR_LED
3	P3_5	P1_5	P9_5	OK_LED
4	P3_4	P1_4	P9_4	CAN_RX
5	P3_3	P1_3	P9_3	CAN_TX
6	P3_2	P1_2	P9_2	CAN_EN
7	P3_1	P1_1	P9_1	NC
8	P3_0	P1_0	P9_0	NC
9	GND	GND	GND	GND
10	RESRV 11	RESRV 3	RESRV 14	NC
11	P5_7	P2_7	P8_7	NC
12	P5_6	P2_6	P8_6	LIN1_TX
13	P5_5	P2_5	P8_5	LIN2_TX
14	P5_4	P2_4	P8_4	LIN1_RX
15	P5_3	P2_3	P8_3	LIN2_NSLP
16	P5_2	P2_2	P8_2	LIN1_NSLP
17	P5_1	P2_1	P8_1	LIN2_RX
18	P5_0	P2_0	P8_0	WARN_LED
19	GND	GND	GND	GND
20	RESRV 10	RESRV 2	RESRV 13	NC
21	P4_7	P0_7	P7_7	NC
22	P4_6	P0_6	P7_6	NC
23	P4_5	P0_5	P7_5	NC
24	P4_4	P0_4	P7_4	NC
25	P4_3	P0_3	P7_3	NC
26	P4_2	P0_2	P7_2	NC
27	P4_1	P0_1	P7_1	NC
28	P4_0	P0_0	P7_0	NC
29	GND	GND	GND	GND
30	RESRV 9	RESRV 1	RESRV 12	NC
31	P12_3	P12_3	P12_3	NC
32	P12_2	P12_2	P12_2	NC
33	P12_1	P12_1	P12_1	SDA

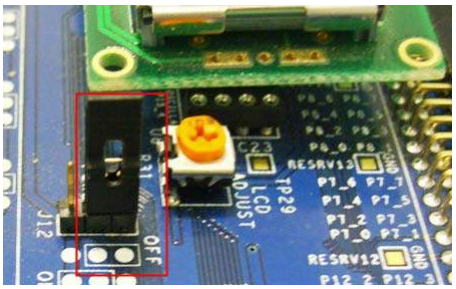
Pin	Port A	Port B	Port C	CAN/LIN EBK
34	P12_0	P12_0	P12_0	SCL
35	V3_3	V3_3	V3_3	V3_3
36	VADJ	VADJ	VADJ	VADJ
37	GND	GND	GND	GND
38	V5_0	V5_0	V5_0	V5_0
39	VIN	VIN	VIN	VIN
40	GND	GND	GND	GND

4.5.1 Jumper Settings of CY8CKIT-001 DVK for Using Port B

Port B uses the port 2 GPIO pins to connect to two of the LEDs and the two LIN transceiver circuits on the expansion board. Therefore, switch the jumper J12 to 'Off' position; this switches off the power for the character LCD that is connected to port 2 of the CY8CKIT-001 DVK. This means that the LCD on the DVK cannot be used when the port 2 GPIO pins are used and the expansion board is occupying port B of the DVK.

It is possible to use the EBK for CAN on port B of the DVK along with the LCD of the DVK. In this case, the LEDs and both LIN circuits on the EBK should be isolated from the 2x20 header by removing the corresponding 0-Ω resistors on the expansion board.

Figure 4-6. J12 Jumper in Off Position



4.5.2 Debugging Restrictions When Using Port B

When the CY8CKIT-017 EBK occupies port B, debugging is possible only through Serial Wire Debug (SWD) interface. It is not possible through JTAG and Serial Wire Viewer (SWV) debug interfaces because expansion board circuits conflict with GPIO pins of these (JTAG and SWV) interfaces.

4.6 Power Supply Configurations

By default, the CAN/LIN expansion board is powered from the CY8CKIT-001 DVK through the 40-pin (2×20) connector.

The expansion board has a selection jumper on it (JP6) that must be set correctly to match the power settings of the DVK. This jumper can be set to power up the circuits on the expansion board with the V5_0, V3_3, or VADJ power supplies from the DVK board.

The CAN transceiver circuit only works when it is powered with 5 V. Therefore, JP6 must be set to the V5_0 setting when using CAN. The LIN circuits can works at either the V5_0 or the V3_3 power supplies.

The expansion board has a pin on each CAN and LIN header (P2, P4, and P5) that can be connected to the VIN power supply of the DVK. One reason to use this is if there are two CY8CKIT-

001 boards (both with CY8CKIT-017 EBKs) that are connected to each other between their CAN connectors or LIN connectors. In this case, it is useful to allow the VIN power supply of one DVK to power up the other DVK (and its EBK) through the LIN or CAN header.

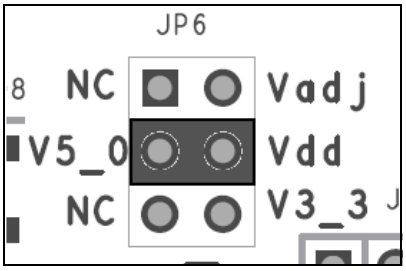
The CANEXTPWER jumper (JP3) is used to connect the VIN supply to a CAN connector pin. The LIN1EXTPWER jumper (JP4) is used to connect the VIN supply to a LIN1 connector pin. The LIN2EXTPWER jumper (JP5) is used to connect the VIN supply to a LIN2 connector pin.

Warning: Take extra care when populating any of the JP3, JP4, or JP5 jumpers. These jumpers should only be populated when only one VIN power supply exists in the system.

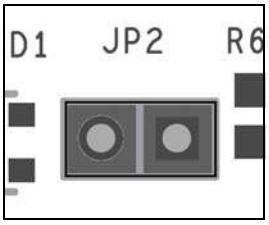
4.7 Default Switch and Jumper Settings

Jumpers on the CY8CKIT-017 CAN/LIN expansion board have a default setting for 5 V operation. For default configuration, each of the jumpers must be set according to these instructions (any jumpers on the board not mentioned below should have no jumper installed).

JP6 - VDD Select. Default Position: 5V (middle two pins)



JP2 - CAN Termination Resistor. Default Position: Installed



5. Code Examples

The CAN_Example_1 project demonstrates the implementation of a CAN node using a CY8CKIT-001 DVK, a CY8CKIT-009 PSoC 3 processor module, and a CY8CKIT-017 CAN/LIN EBK. The test setup shown in Figure 3-8 on page 17 consists of two CAN nodes, created using two CY8CKIT-001 DVKs, two CY8CKIT-009 PSoC 3 processor modules, and two CY8CKIT-017 CAN/LIN EBKs connected by a DB9 cable.

The CAN_Example_2 project is programmed into the second CY8CKIT-001 DVK making it a CAN node that communicates with the CAN node created by programming CAN_Example_1 into the first CY8CKIT-001 DVK. The two code examples are identical except that their transmit and receive ID's are reversed.

The LIN_Example project demonstrates the LIN slave operation on PSoC 3. This uses the CY8CKIT-001 DVK, CY8CKIT-009 PSoC 3 processor module, LIN bus analyzer, and the CY8CKIT-017 CAN/LIN EBK.

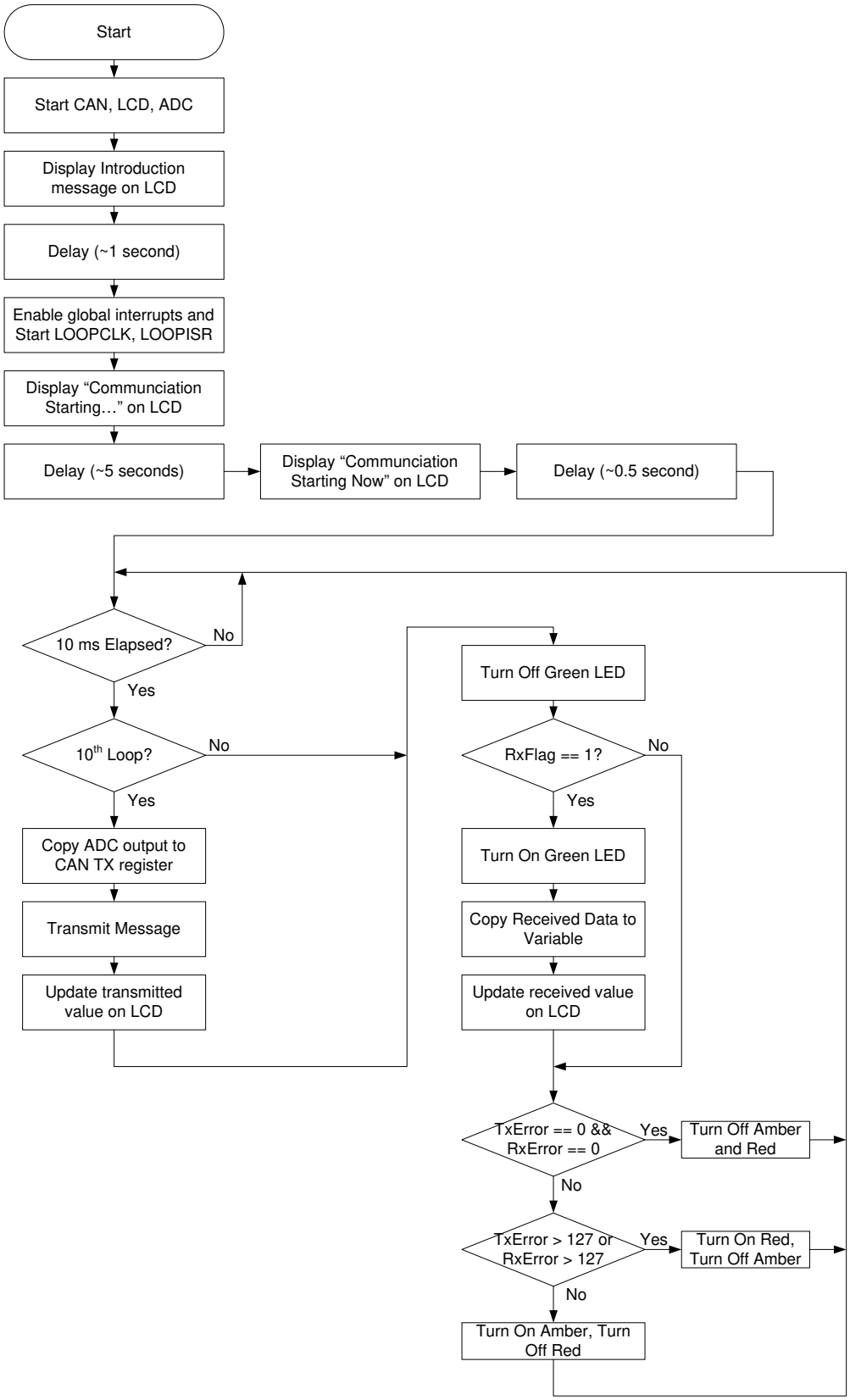
5.1 Code Example 1: CAN_Example_1

In the CAN_Example_1 project, the CAN component is configured to transmit messages at a baud rate of 500 kbps in full TX/RX mode. This CAN node has a transmit ID of 0x2FF and a receive ID of 0x3FF. The potentiometer present on the first CY8CKIT-001 DVK is sampled by a Sigma-Delta ADC that is configured with 8-bit resolution and a sampling rate of 10 ksps. The result of the conversion is stored in a variable in the ISR at the end of each ADC sample.

CAN messages are scheduled to be transmitted every 100 ms. The ADC samples are transmitted in these CAN messages. Every 10 ms a “CAN message received” flag is polled. This flag is set every time a CAN message is received. If the flag is set, the received data is copied from the receive buffer to a variable. The LCD on the first CY8CKIT-001 DVK displays the updated value of both the transmitted and received data. Three LEDs (green, amber, and red) present on the CY8CKIT-017 show the status of the CAN transmission.

The green LED flashes any time a CAN message is received successfully. The amber LED turns on continuously any time either CAN error counter is between 0 and 127, inclusive. The red LED turns on whenever either CAN error counter exceeds 127.

Figure 5-1. Firmware Flowchart



5.1.1 Running the Code Example

To program the PSoC 3 device with the CAN_Example_1 project, follow the instructions described in Programming PSoC 3 Device on page 13.

5.1.2 Hardware Connections

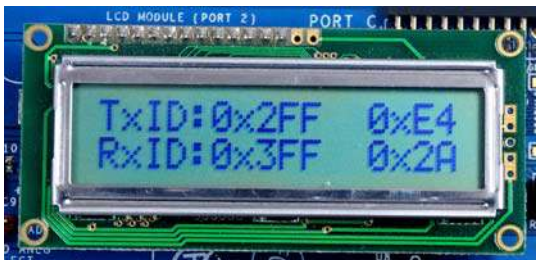
This code example is designed to work only when the EBK is connected to port A of the CY8CKIT-001 DVK. You can modify the code example to work on other ports of the DVK. See Design Wide Resources on page 48 for details on changing the project's pinout.

For more information on hardware connections, see Hardware Connections on page 16.

5.1.3 Verifying Output

As you vary the potentiometer of the first CY8CKIT-001 DVK, observe the change in the transmitted byte information on the first line of LCD present on the CY8CKIT-001 DVK. The same value is reflected as the received byte information on the second line of the LCD present on the second CY8CKIT-001 DVK.

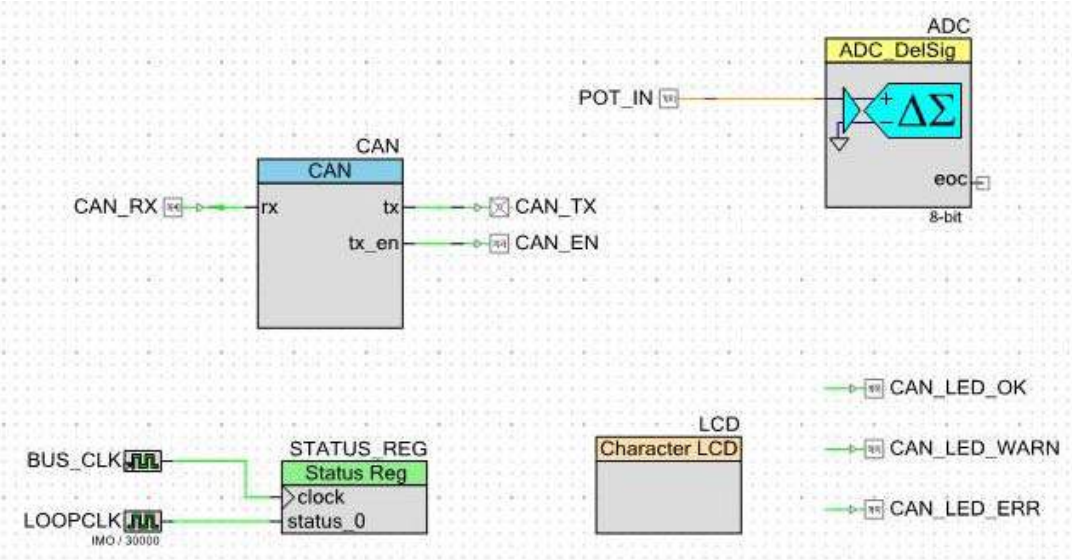
Figure 5-2. CAN_Example_1 Project Verification



5.1.4 PSoC Creator Project Details

PSoC Creator offers a flexible hardware and software co-design environment to create and configure the programmable peripherals.

Figure 5-3. PSoC Creator Top Level Design For CAN_Example_1 Project



5.1.4.1 CAN

CAN is the core component in this code example. The component enables you to set the message IDs and the transmission mode.

In the CAN_Example_1 project, the transmit message ID is set to 0x2FF, the receive message ID is set to 0x3FF, and the baud rate is set to 500 kbps. The transmission mode is set to full TX/RX mode. The CAN_Example_2 project has the same settings, except that the transmit message ID is set to 0x3FF and the receive message ID is set to 0x2FF.

Notes

- For details of parameters, refer to the component datasheets.
- The component figure shows only tabs in which settings have been changed from default states or in which critical settings exist for proper operation. Any tabs not shown have default settings. This is valid for all components of all code examples.

Figure 5-4. CAN Configuration: General Tab

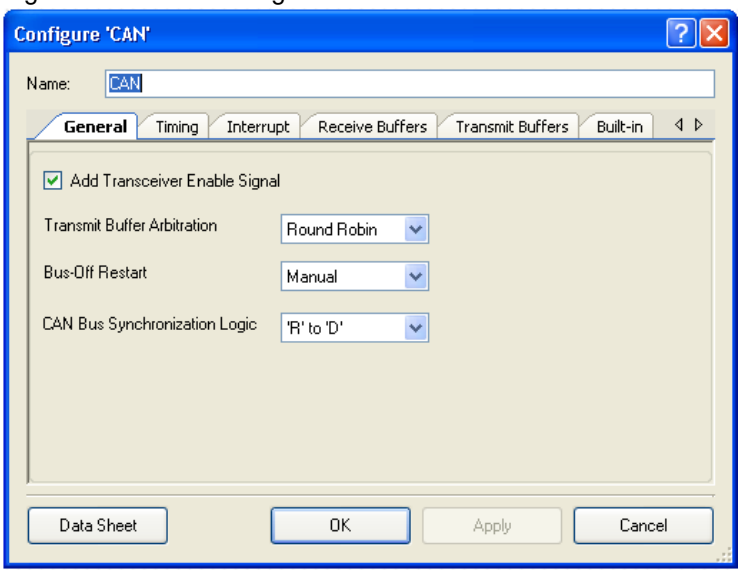


Figure 5-5. CAN Configuration: Timing Tab

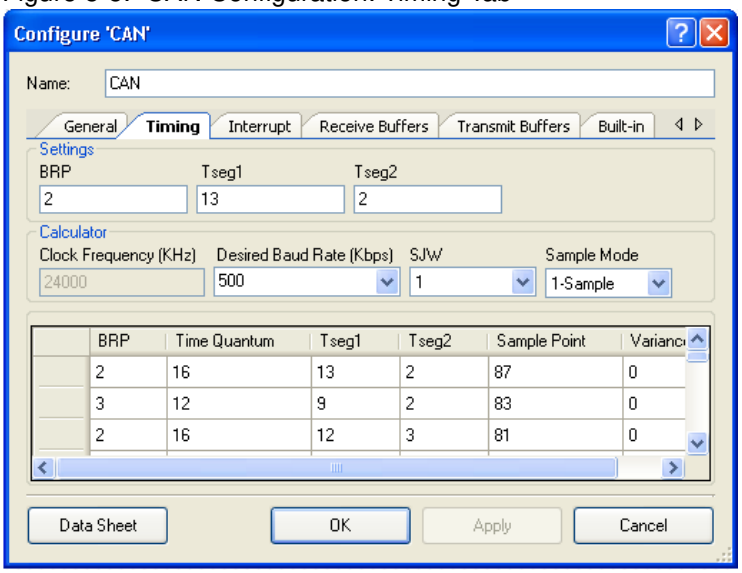


Figure 5-6. CAN Configuration: Interrupt Tab

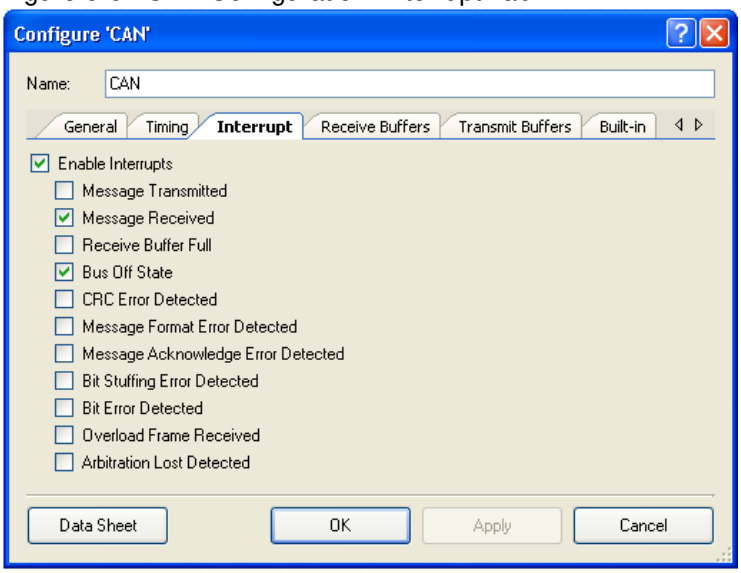


Figure 5-7. CAN Configuration: Receive Buffers Tab

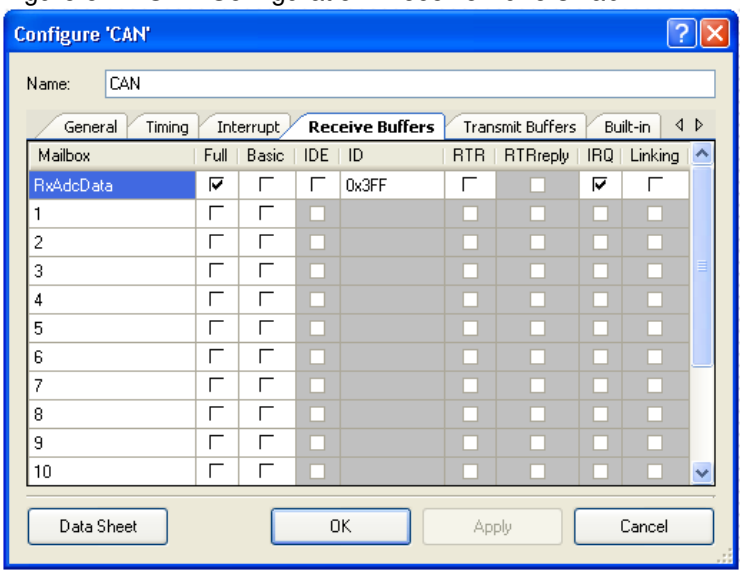
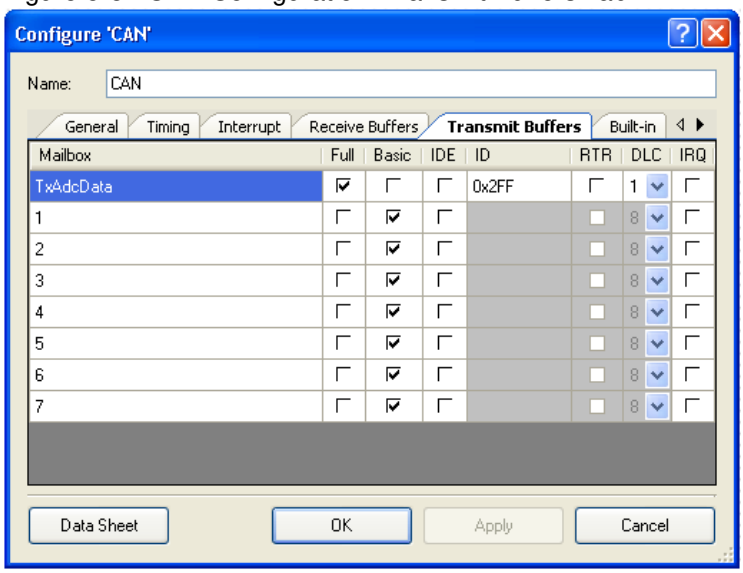


Figure 5-8. CAN Configuration: Transmit Buffers Tab



5.1.4.2 ADC

The ADC component is used to sample the potentiometer input.

Figure 5-9. ADC Configuration: Configure Tab

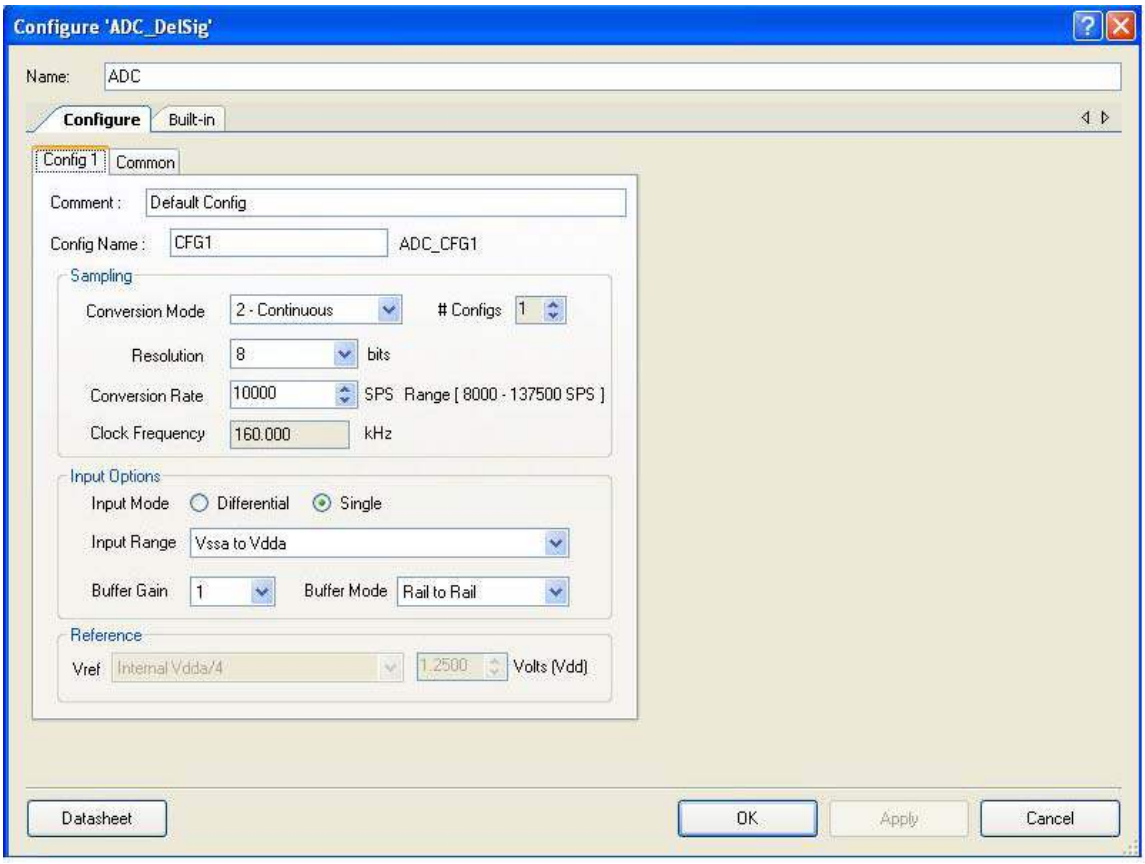
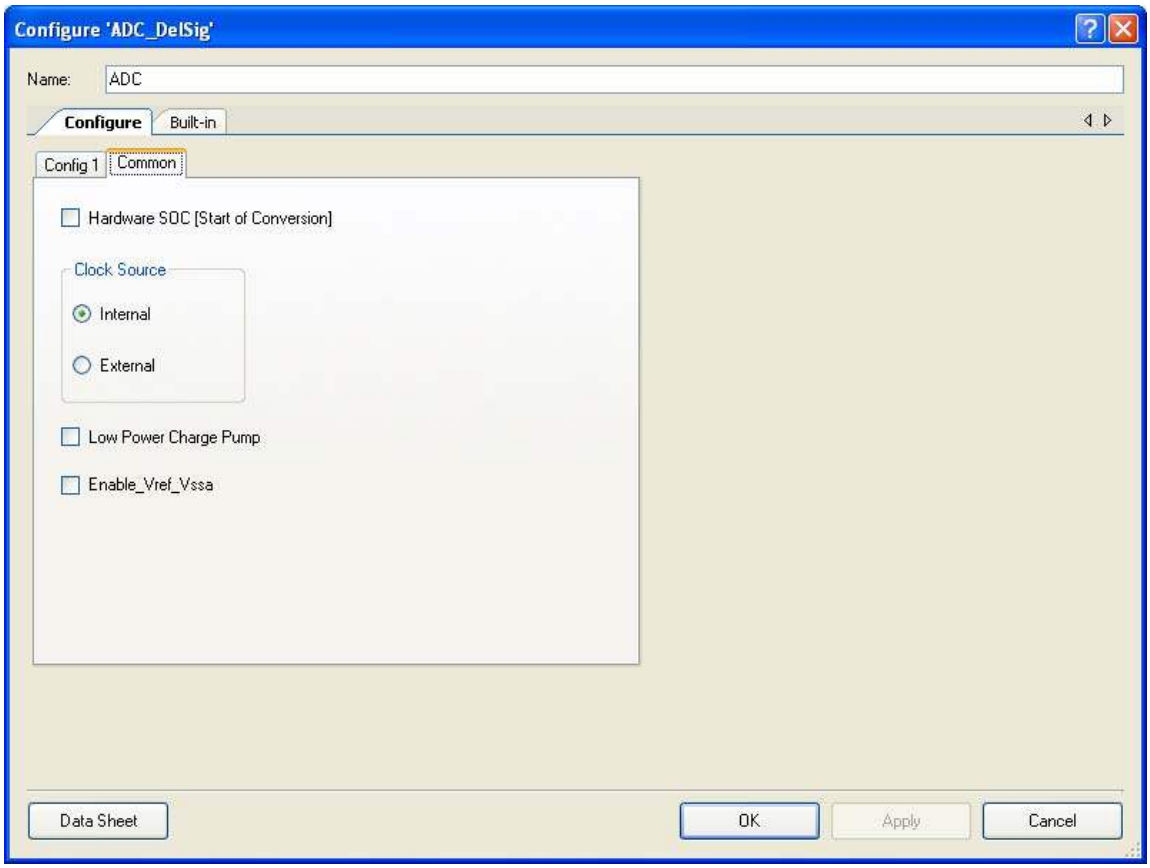


Figure 5-10. ADC Configuration: Common Tab



5.1.4.3 POT_IN

The POT_IN pin is used to input the analog signal from the potentiometer. The pin's drive mode is configured as High Impedance Analog, which is the default value.

Figure 5-11. POT_IN Configuration: Pins > Type Tab

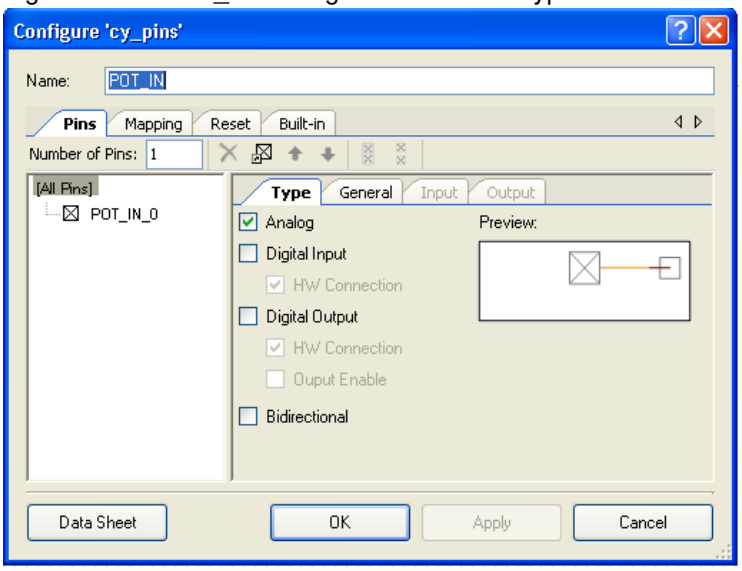
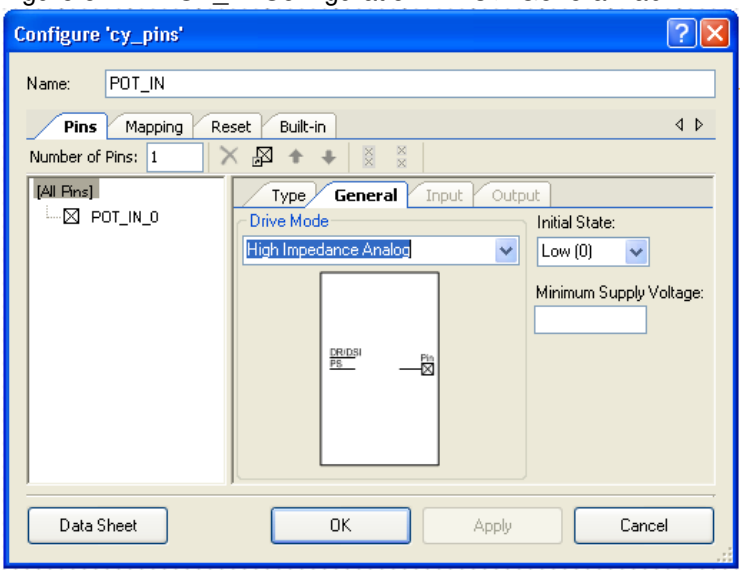


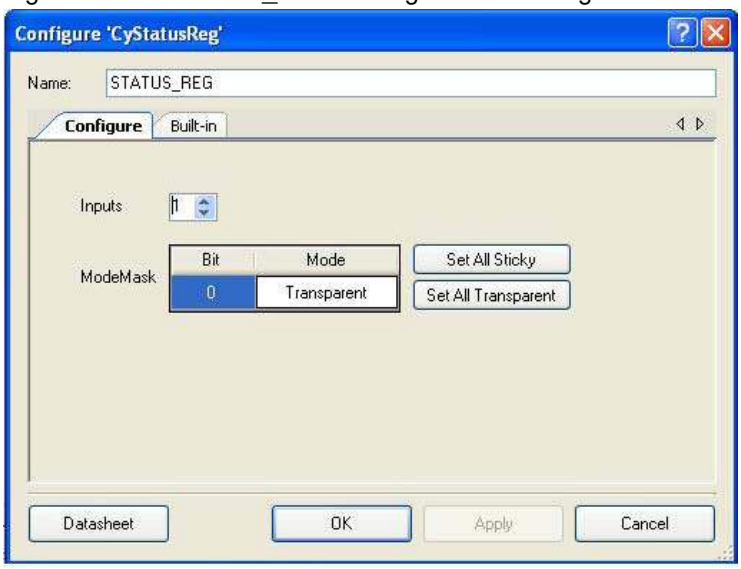
Figure 5-12. POT_IN Configuration: Pins > General Tab



5.1.4.4 STATUS_REG

The status register is used to read the state of the LOOPCLK clock component. The output of this register is used to detect the rising edge of the LOOPCLK in this project.

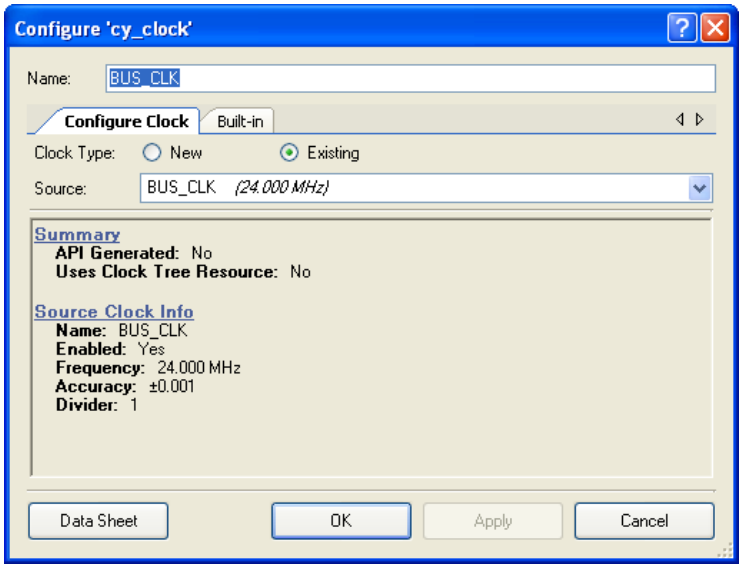
Figure 5-13. STATUS_REG Configuration: Configure Tab



5.1.4.5 BUS_CLK

The BUS_CLK is used as the latching clock for the STATUS_REG component. This is an existing, high-frequency clock in the chip.

Figure 5-14. BUS_CLK Configuration: Configure Clock Tab



5.1.4.6 LOOPCLK

LOOPCLK is configured to generate a 100 Hz clock, which is used to generate a 10 ms period in the firmware.

Figure 5-15. LOOPCLK Configuration: Configure Clock Tab

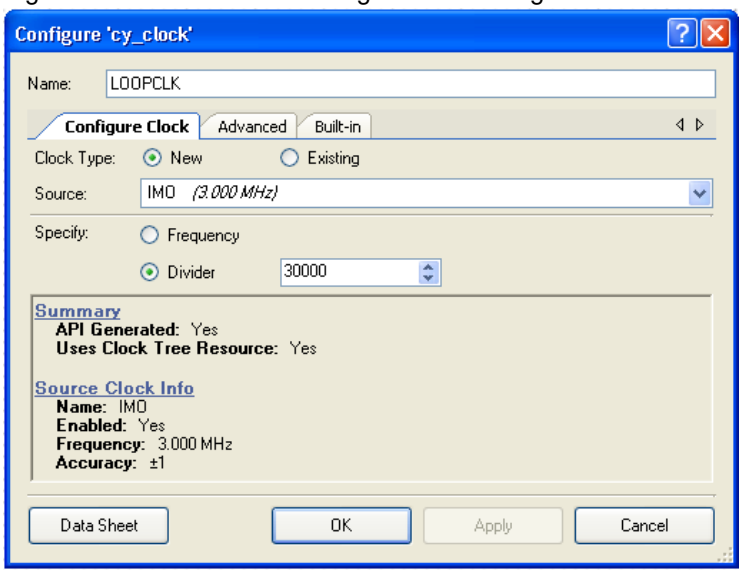
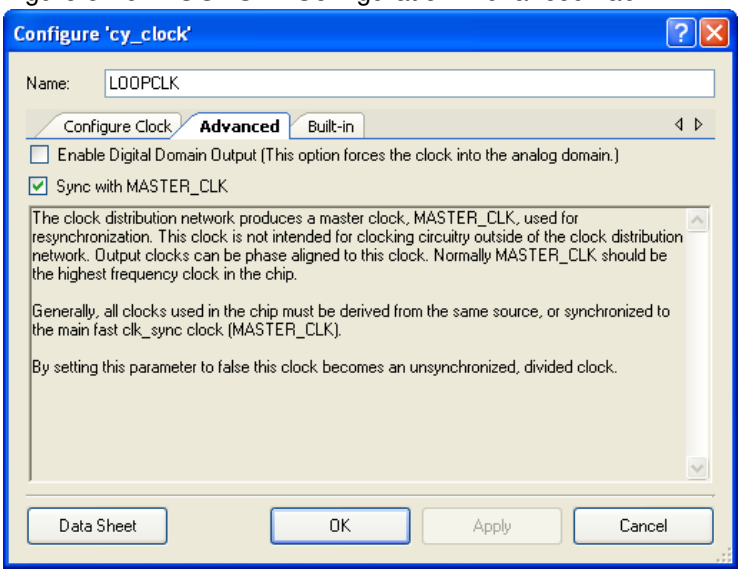


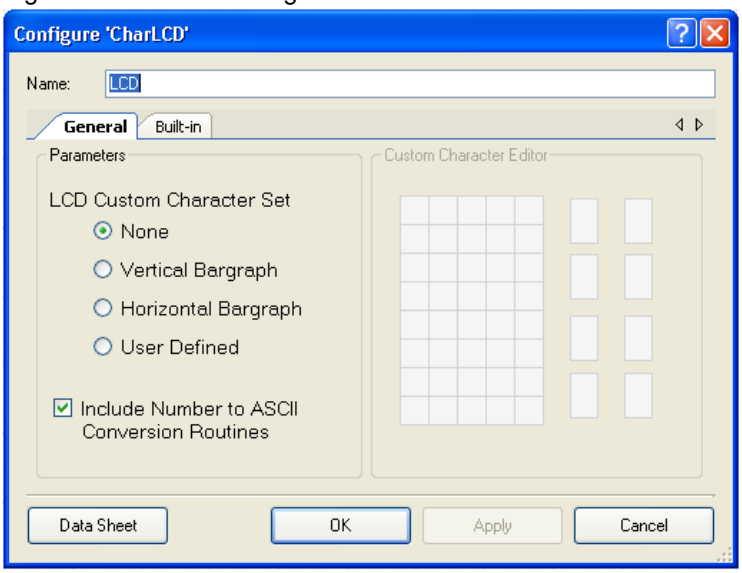
Figure 5-16. LOOPCLK Configuration: Advanced Tab



5.1.4.7 LCD

The Character LCD is used to display the updated information of the transmitted and received bytes along with their transmit and receive IDs.

Figure 5-17. LCD Configuration: General Tab



5.1.4.8 CAN_TX

CAN_TX is the CAN bus transmit signal pin. This pin is configured as an output pin with a strong drive mode. It must be connected to the CAN TX input of the external CAN transceiver.

Figure 5-18. CAN_TX Configuration: Pins > Type Tab

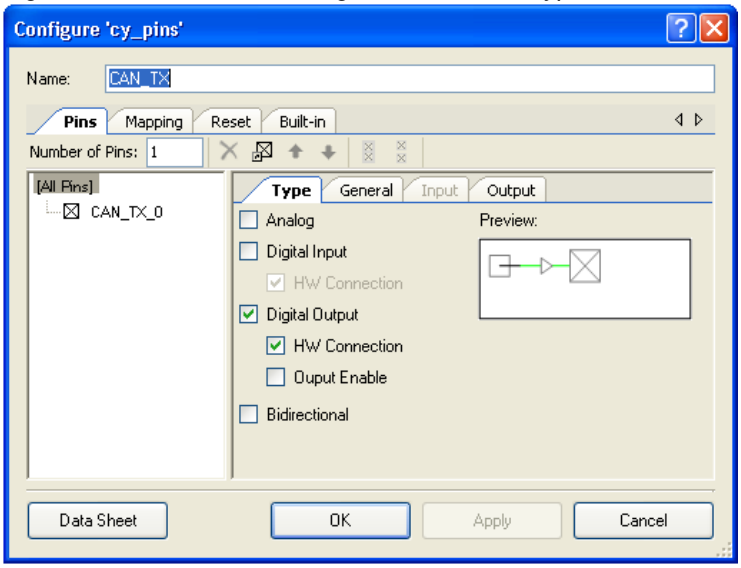
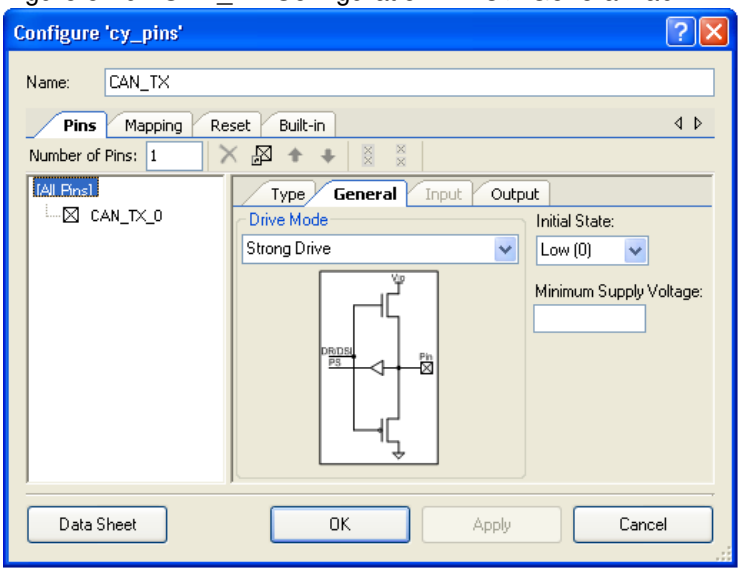


Figure 5-19. CAN_TX Configuration: Pins > General Tab



5.1.4.9 *CAN_RX*

CAN_RX is the CAN bus receive signal pin. This pin is configured as an input pin with a high impedance drive mode. It must be connected to the CAN RX pin of the external CAN transceiver.

Figure 5-20. CAN_RX Configuration: Pin > Type Tab

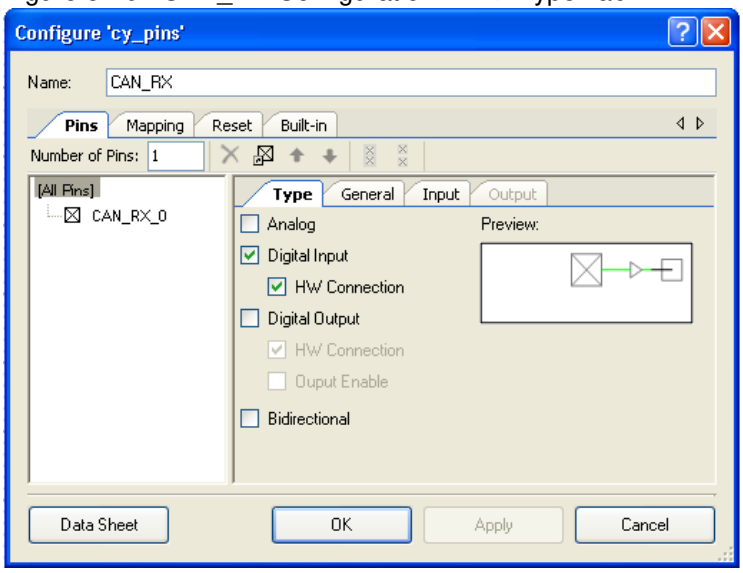


Figure 5-21. CAN_RX Configuration: Pins > General Tab

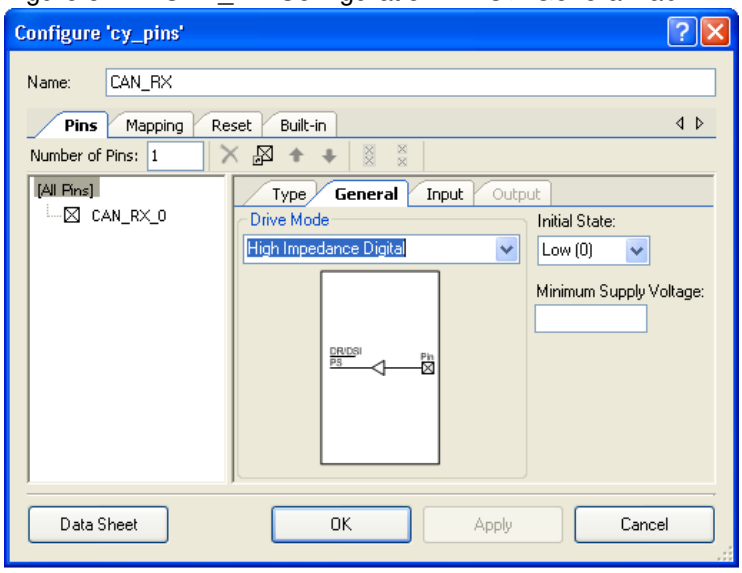
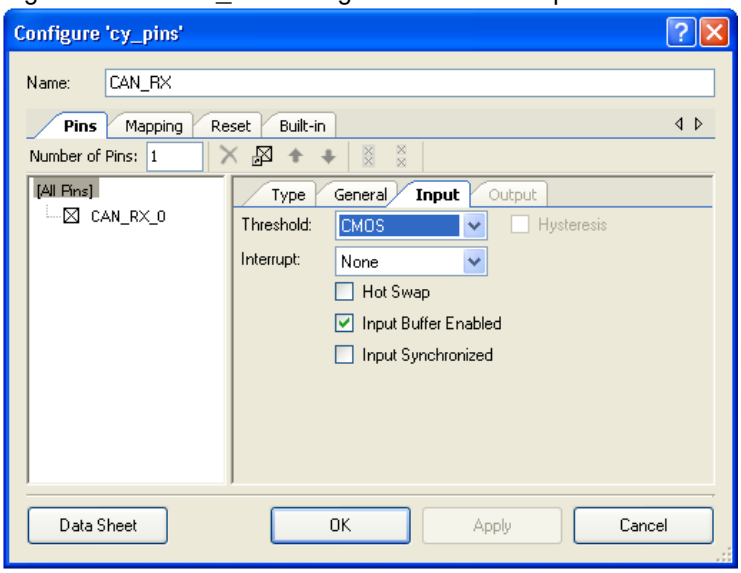


Figure 5-22. CAN_RX Configuration: Pins > Input Tab



5.1.4.10 CAN_EN

CAN_EN is external CAN transceiver enable signal pin. This pin is configured as an output pin with strong drive mode.

Figure 5-23. CAN_EN Configuration: Pins > Type Tab

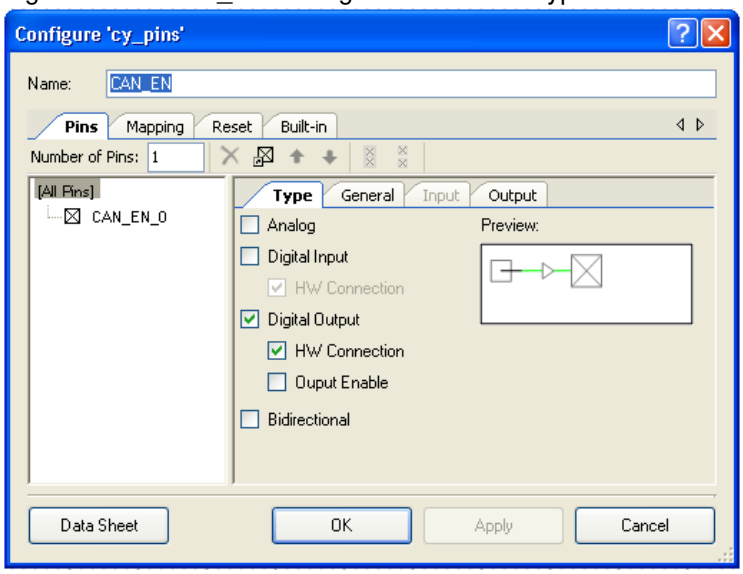
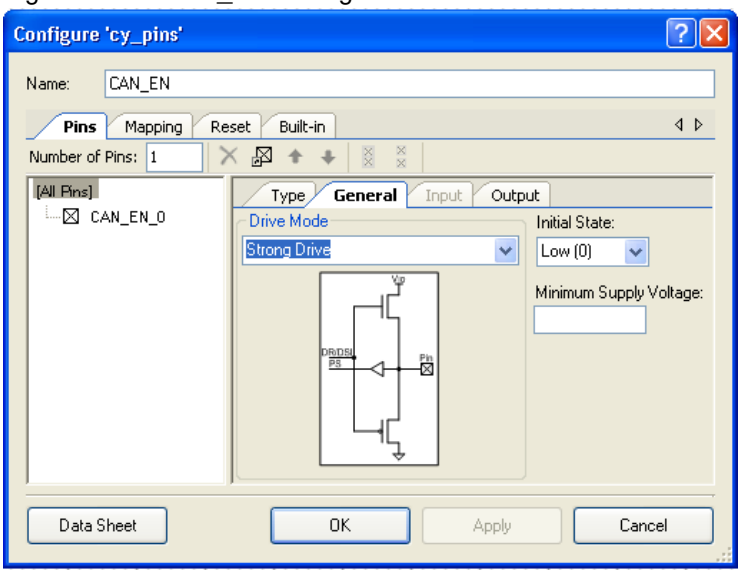


Figure 5-24. CAN_EN Configuration: Pins > General Tab



5.1.4.11 CAN_LED_OK

CAN_LED_OK is configured as a software controlled output pin with strong drive mode and initial state as high. This pin is connected to the green LED on the CY8CKIT-017 CAN/LIN EBK.

Figure 5-25. CAN_LED_OK Configuration: Pins > Type Tab

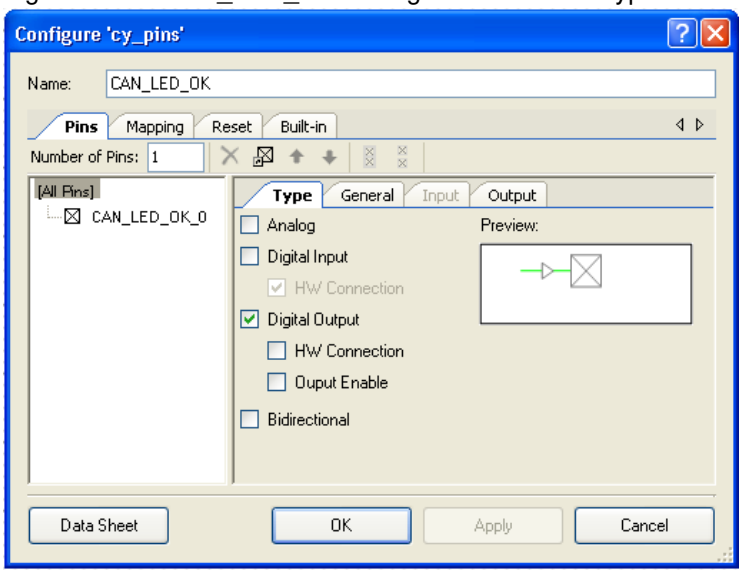
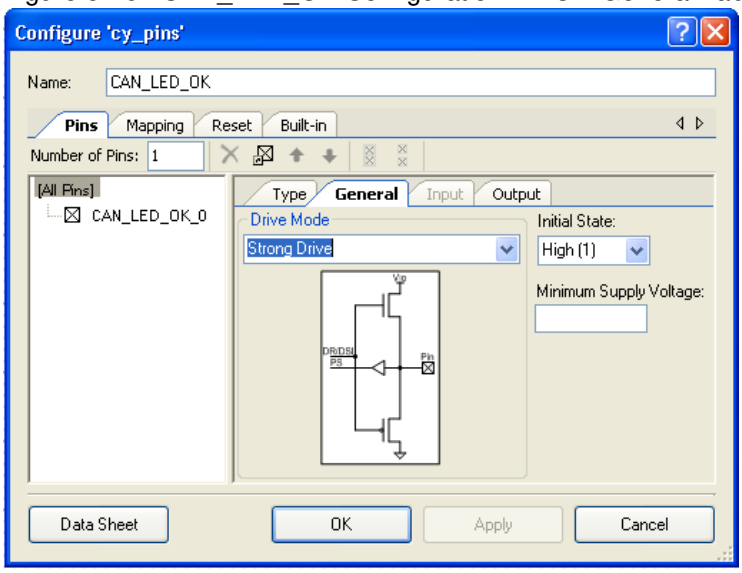


Figure 5-26. CAN_LED_OK Configuration: Pins > General Tab



5.1.4.12 CAN_LED_WARN

CAN_LED_WARN is configured as a software controlled output pin with strong drive mode and initial state as high. This pin is connected to the amber LED on the CY8CKIT-017 CAN/LIN EBK.

Figure 5-27. CAN_LED_WARN Configuration: Pins > Type Tab

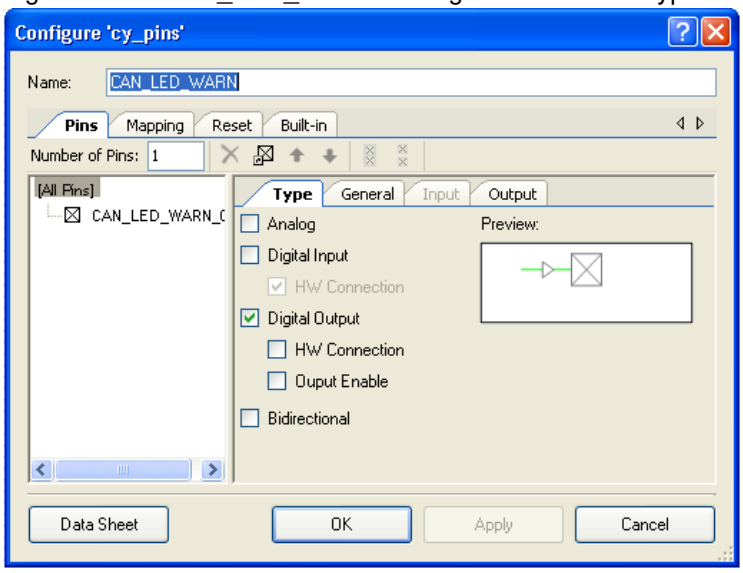
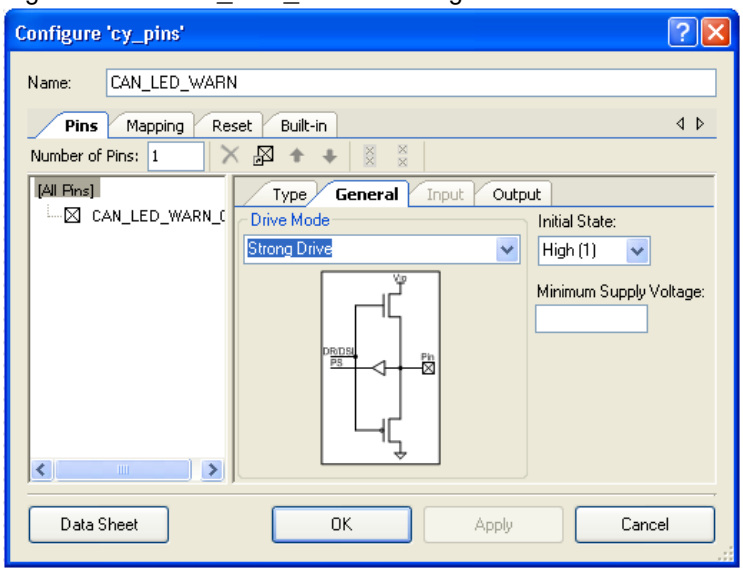


Figure 5-28. CAN_LED_WARN Configuration: Pins > General Tab



5.1.4.13 CAN_LED_ERR

CAN_LED_ERR is configured as a software controlled output pin with strong drive mode and initial state as high. This pin is connected to the red LED on the CY8CKIT-017 CAN/LIN EBK.

Figure 5-29. CAN_LED_ERR Configuration: Pins > Type Tab

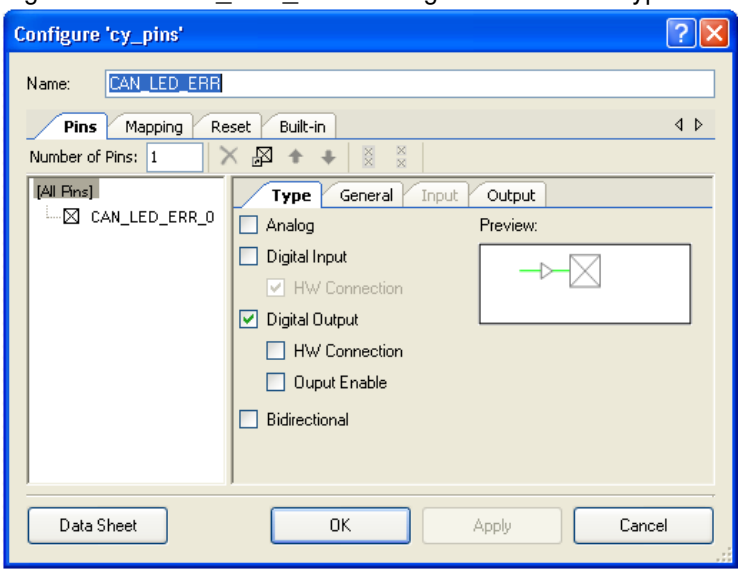
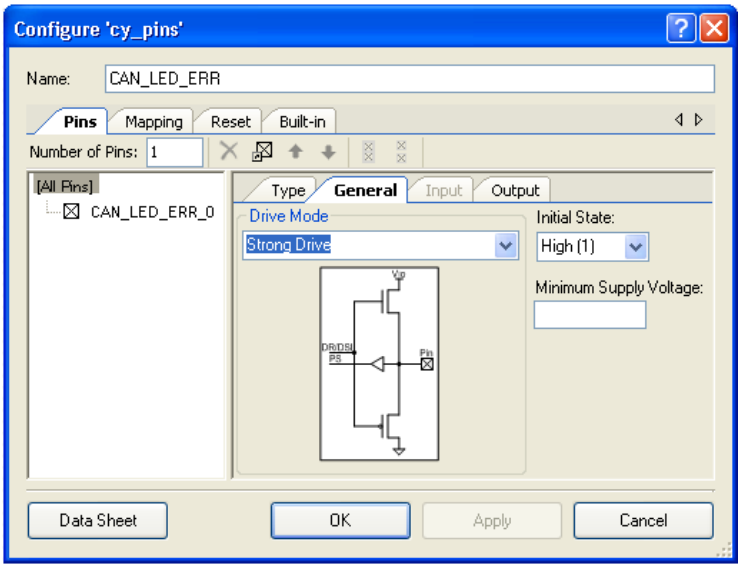


Figure 5-30. CAN_LED_WARN Configuration: Pins > General Tab



5.1.4.14 Design Wide Resources

The pin assignment in this code example matches port A of the CY8CKIT-001 DVK. To use this EBK on port B or port C of the DVK, open the code example and change the pin assignment in PSoC Creator (in the .cydwr file) to match port B or port C according to Table 4-4 on page 28. The pin assignment for this code example 1 is shown in Figure 5-31.

Figure 5-31. Pin Connection Mapping for Port A of CY8CKIT-001 DVK

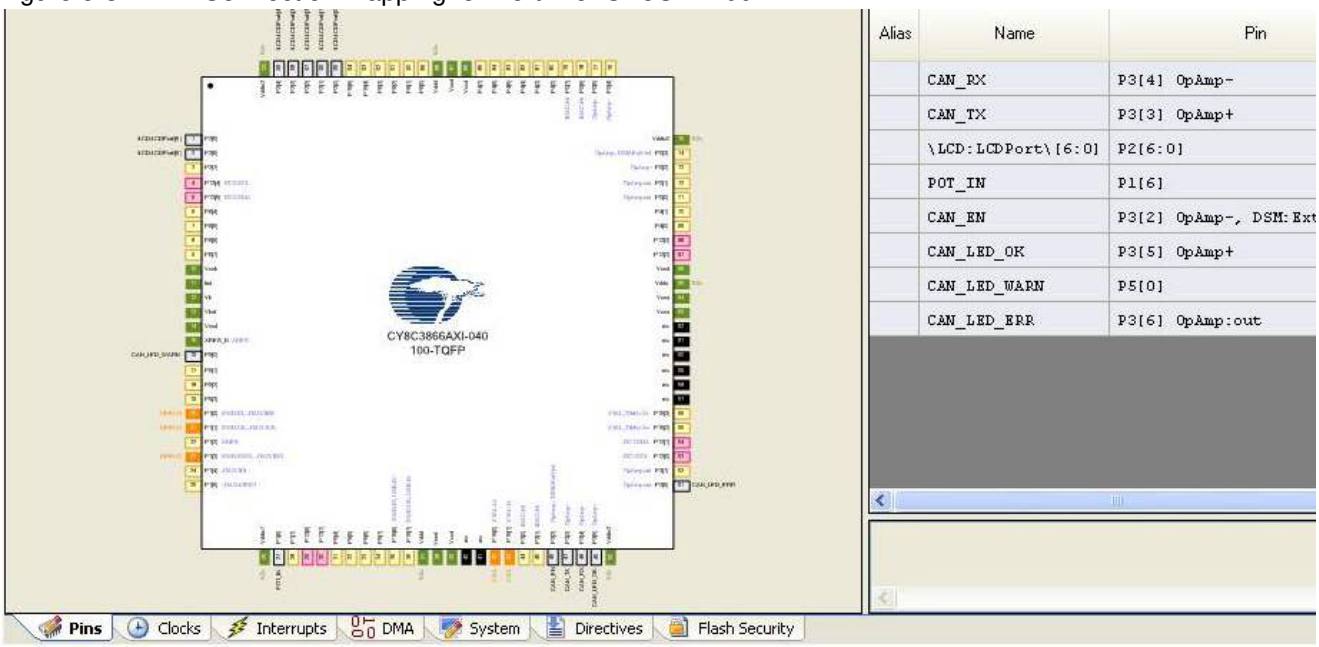


Table 5-1. Pin Assignment Details of CAN Example

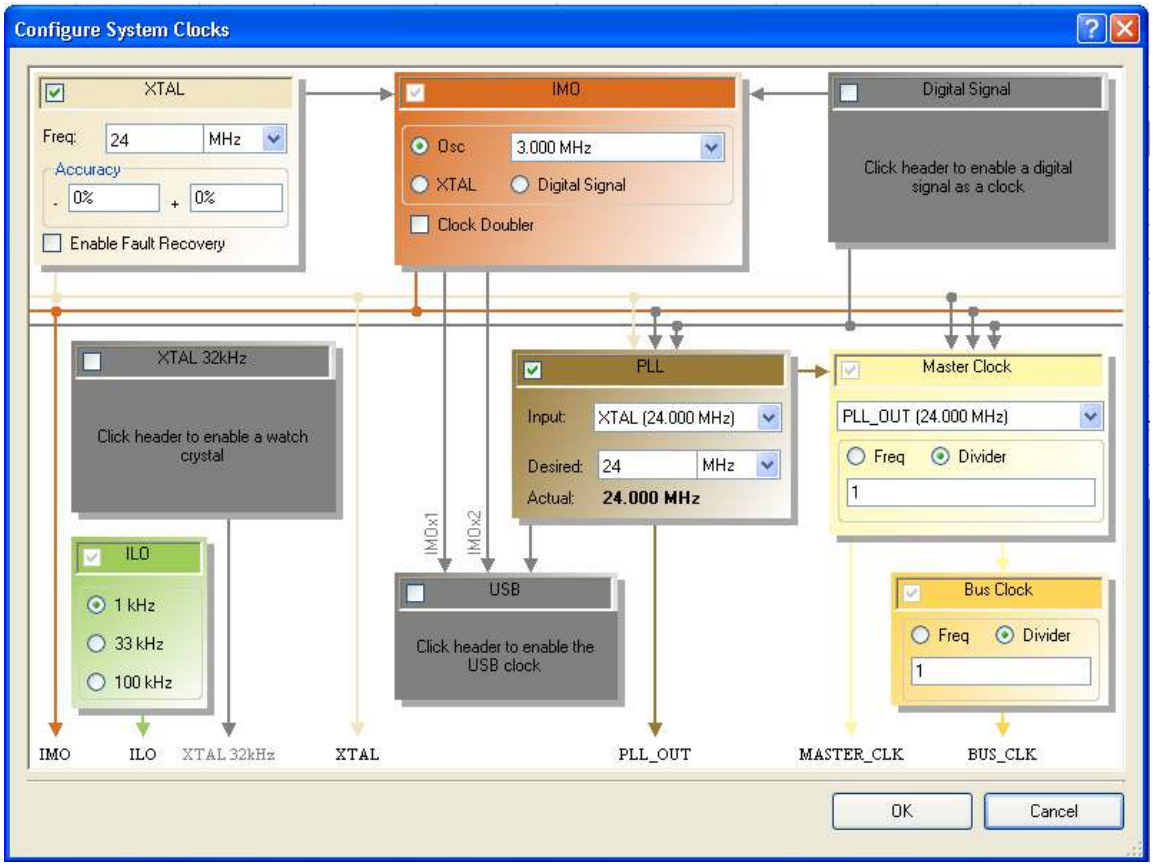
Signal	CY8CKIT-001 DVK (Port A)	CY8CKIT-030 DVK (PORT E)
CAN_LED_WARN	P5[0]	P0[0]
CAN_LED_OK	P3[5]	P3[5]
CAN_LED_ERR	P3[6]	P3[6]
CAN_RX	P3[4]	P3[4]
CAN_TX	P3[3]	P3[3]
LCD	P2[6:0]	P2[6:0]
POT_IN	P1[6]	P6[5]
CAN_EN	P3[2]	P3[2]

This code example only works if the PSoC 3 device is using its External Crystal Oscillator (ECO) circuit with a 24 MHz external crystal. See CAN Bus Clock Accuracy on page 24 for more information on clock source requirements for the PSoC CAN controller. All clock settings of this code example are shown in Figure 5-32 and Figure 5-33 on page 49.

Figure 5-32. Clock Setting

Type	Name	Domain	Desired Frequency	Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on Reset	Source Clock
System	USB_CLK	DIGITAL	48.000 MHz	? MHz	±0	-	1	☐	IMOx2
System	Digital_Signal	DIGITAL	? MHz	? MHz	±0	-	0	☐	
System	XTAL_32KHZ	DIGITAL	32.768 kHz	? MHz	±0	-	0	☐	
System	ILO	DIGITAL	? MHz	1.000 kHz	-50, +100	-	0	☑	
System	IMO	DIGITAL	3.000 MHz	3.000 MHz	±1	-	0	☑	
System	BUS_CLK (CPU)	DIGITAL	? MHz	24.000 MHz	±0.001	-	1	☑	MASTER_CLK
System	MASTER_CLK	DIGITAL	? MHz	24.000 MHz	±0.001	-	1	☑	PLL_OUT
System	XTAL	DIGITAL	24.000 MHz	24.000 MHz	±0.001	-	0	☑	
System	PLL_OUT	DIGITAL	24.000 MHz	24.000 MHz	±0.001	-	0	☑	XTAL
Local	LOOPCLK	DIGITAL	? MHz	100.000 Hz	±1	-	30000	☑	IMO
Local	ADC_theACLK	ANALOG	160.000 kHz	160.000 kHz	±0.001	±12	150	☑	Auto: MASTER_CLK
Local	ADC_Ext_CP_Clk	DIGITAL	1.000 MHz	1.000 MHz	±0.001	±20	24	☑	Auto: MASTER_CLK
Local	CAN_Clock	DIGITAL	? MHz	24.000 MHz	±0.001	-	0	☑	BUS_CLK
Local	BUS_CLK	DIGITAL	? MHz	24.000 MHz	±0.001	-	0	☑	BUS_CLK

Figure 5-33. System Clock Configuration



5.2 Code Example 2: CAN_Example_2

The CAN_Example_2 project is programmed into the second CY8CKIT-009 PSoC 3 processor module making it a CAN node that can communicate with the CAN node created by programming CAN_Example_1 into the first CY8CKIT-009 PSoC 3 processor module. Because CAN communication requires two CAN nodes, two projects are provided.

CAN_Example_2 has a CAN transmit message ID of 0x3FF and a receive message ID of 0x2FF. The only difference between the two projects is the difference in their transmit and receive IDs. All sections describing the CAN_Example_1 project also apply to the CAN_Example_2 project except Figure 5-7 on page 35 and Figure 5-8 on page 36 where the transmit message ID and receive message ID are interchanged.

5.3 Code Example 3: LIN_Example

In the LIN_Example project, LIN Slave component is configured with a baud rate of 19200 and two unconditional frames of eight bytes length. One frame is of type Subscribe for data reception and the other frame is of type Publish for data transmission. The "InFrame" is of unconditional type with Frame ID 0x10, eight bytes long and direction set to Subscribe. The "OutFrame" is of unconditional type with Frame ID 0x11, eight bytes long and direction set to Publish.

It receives unconditional frame having eight bytes of data (Byte 1 is of scalar signal of 7-bit length and Byte 2 to 8 has a byte array signal of 7-byte length) with Frame ID equal to 0x11. The same is written to another unconditional frame. LIN master can read back the data by sending Frame ID 0x11.

LCD is used to display the user interface messages, and received and transmitted frames. The timer is set to a period of 6 seconds and interrupt is generated on the Terminal Count (TC). This will be used to change the user interface messages on LCD display.

5.3.1 Firmware Flowcharts

Figure 5-34. Main Firmware Flowchart

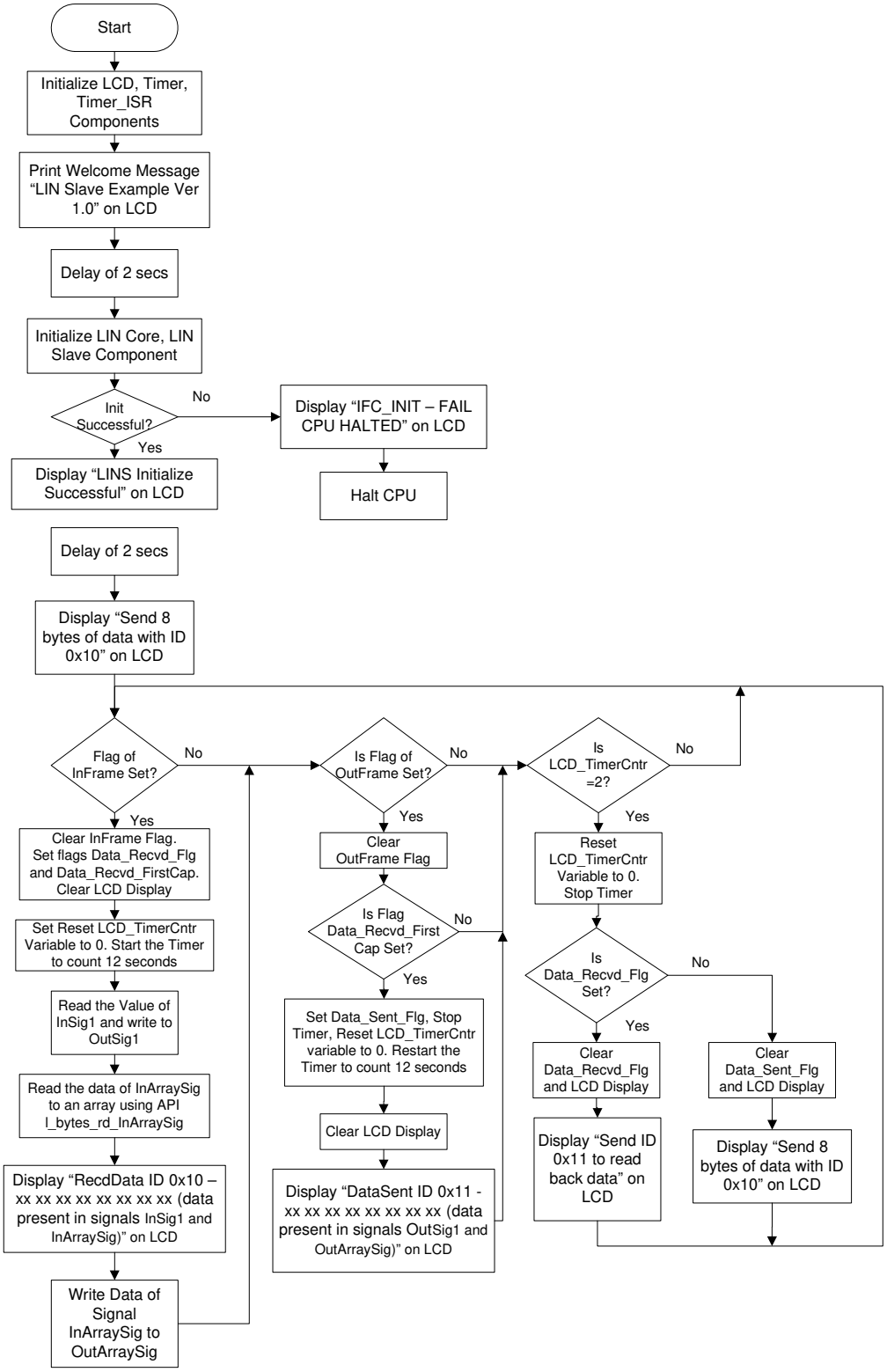
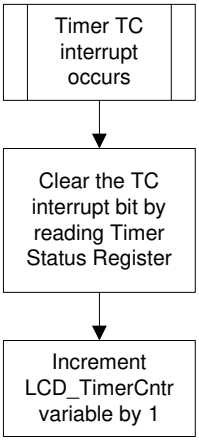


Figure 5-35. Timer ISR Function Flowchart



5.3.2 Running the Code Example

To program the PSoC 3 device with the CAN_Example_1 project, follow the instructions described in Programming PSoC 3 Device on page 13.

5.3.3 Hardware Connections

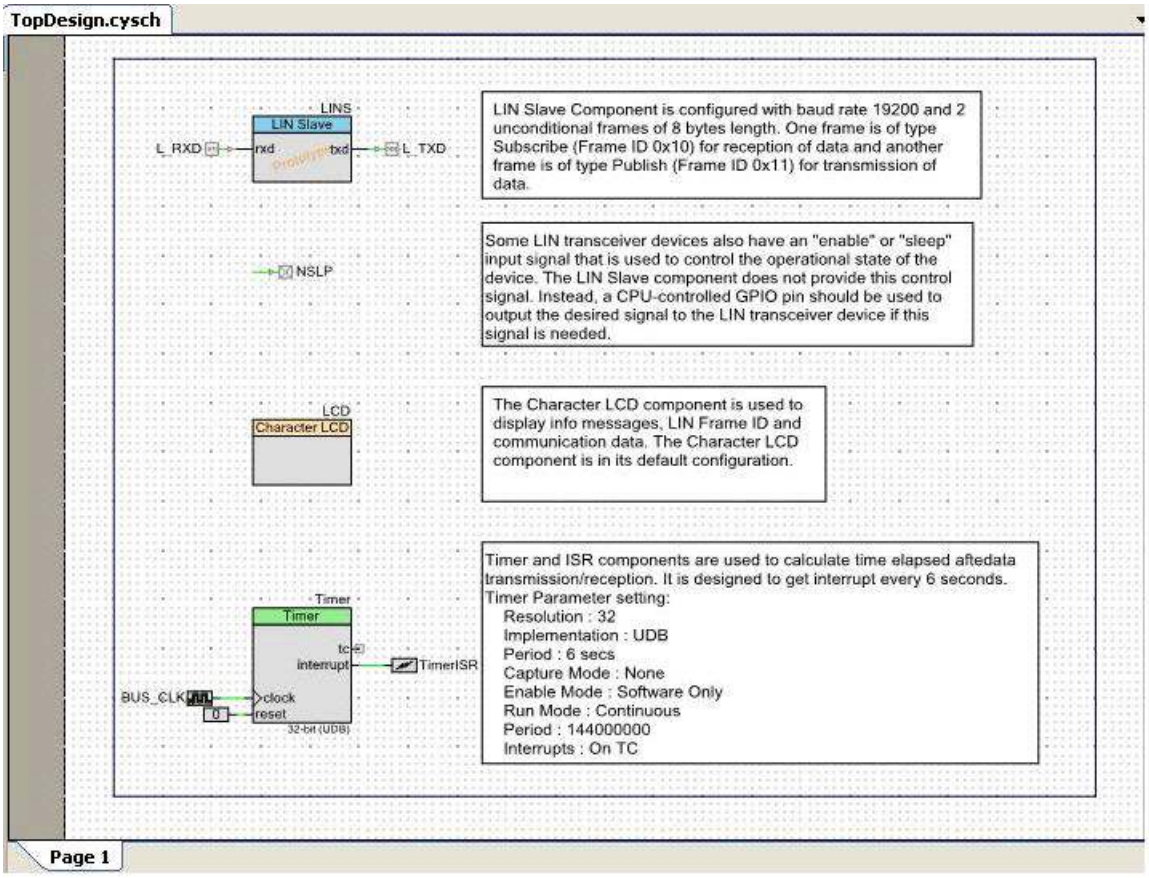
This code example is designed to work only when the EBK is connected to port A of the CY8CKIT-001 DVK. You can modify the code example to work on other ports of the DVK. See Design Wide Resources on page 48 for details on changing the project's pinout. For more information on hardware connections, see Hardware Connections on page 16.

5.3.4 Verifying Output

See the verify functionality described in section LIN Communication on page 20 for the LIN code example verification.

5.3.5 PSoC Creator Project Details

Figure 5-36. PSoC Creator Schematic for LIN Example Project



5.3.5.1 LIN Slave

LIN slave is the core component in this code example.

Notes

- For details of parameters, see the component datasheets.
- The component figure shows only tabs in which settings have been changed from default states or in which critical settings exist for proper operation. Any tabs not shown have default settings. This is valid for all components of all code examples.

Figure 5-37. LIN Configuration: General Tab

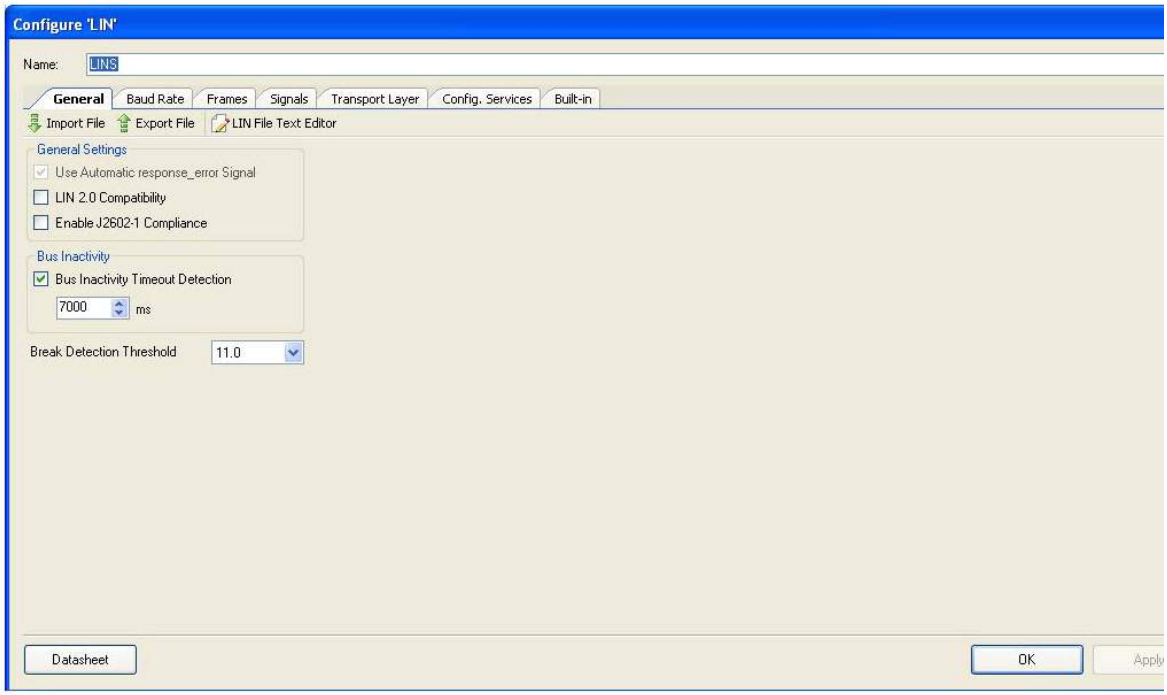


Figure 5-38. LIN Configuration: Baud Rate Tab

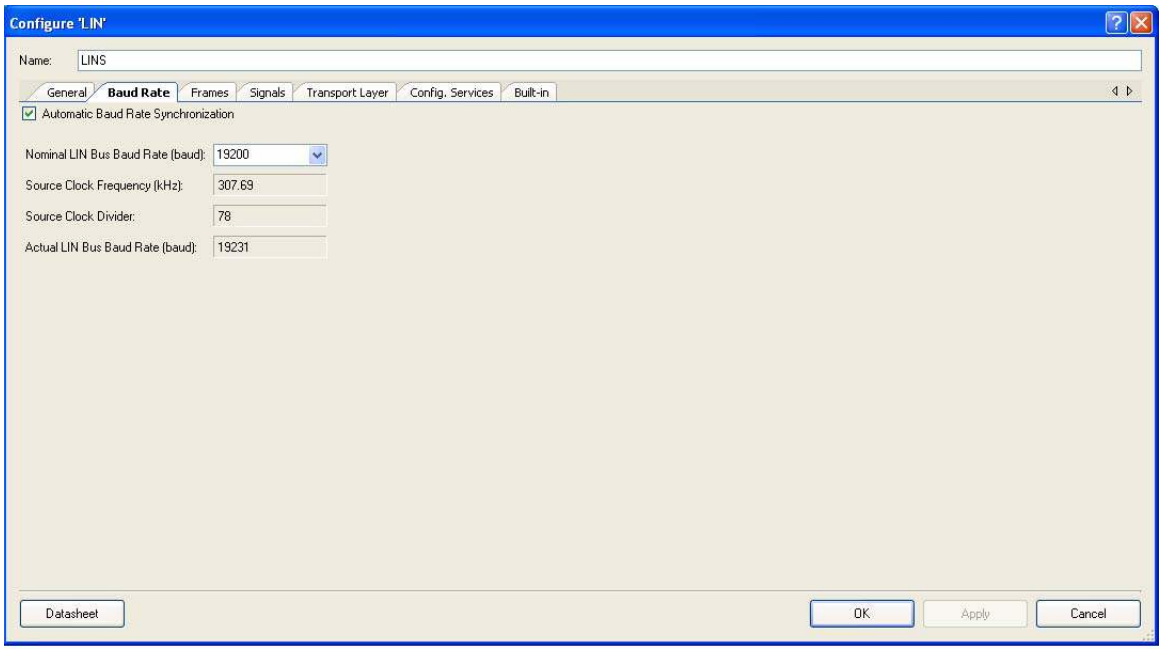


Figure 5-39. LIN Configuration: Frames Tab

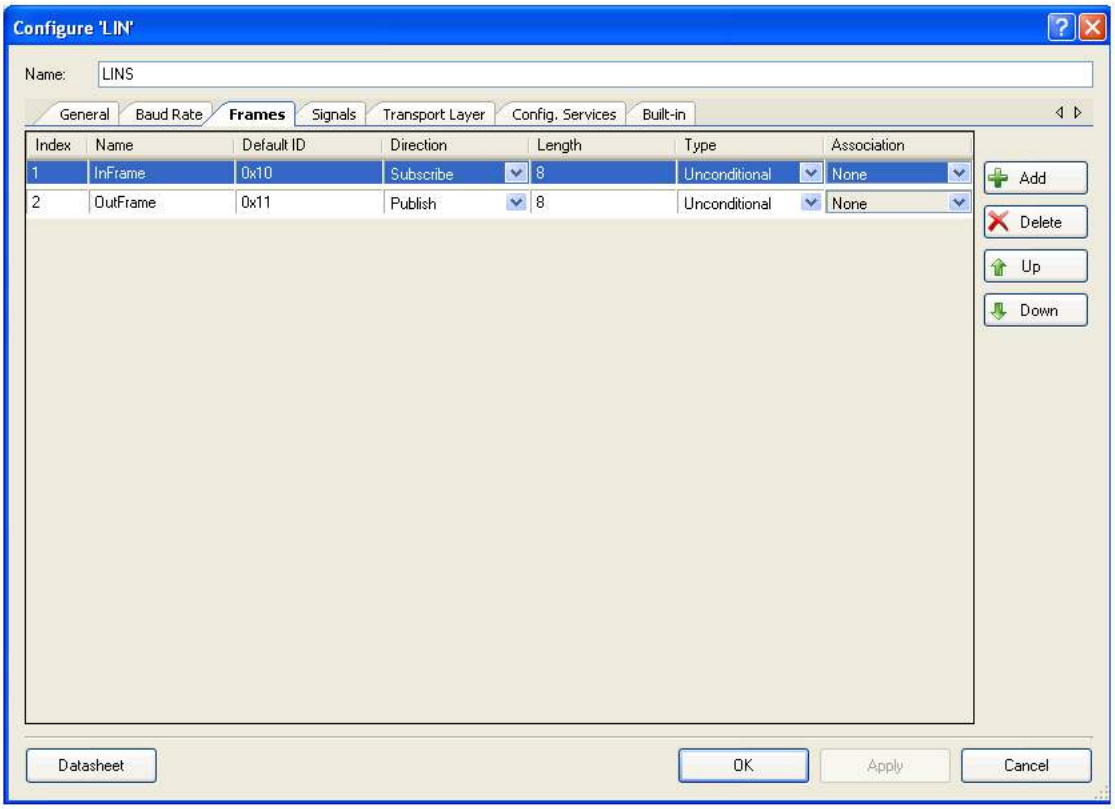
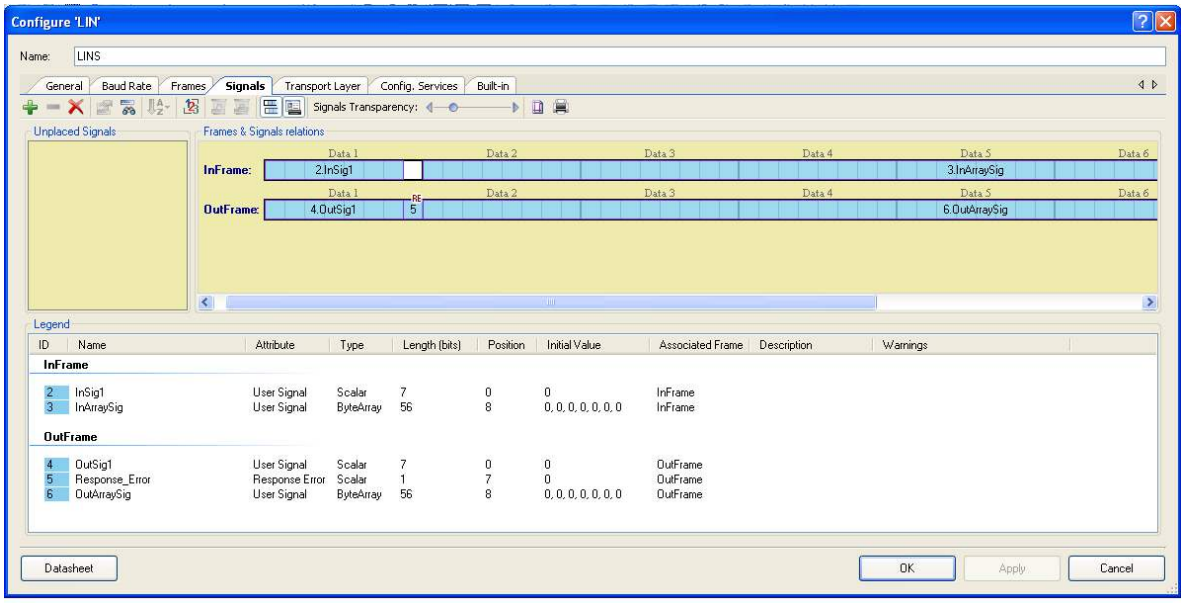


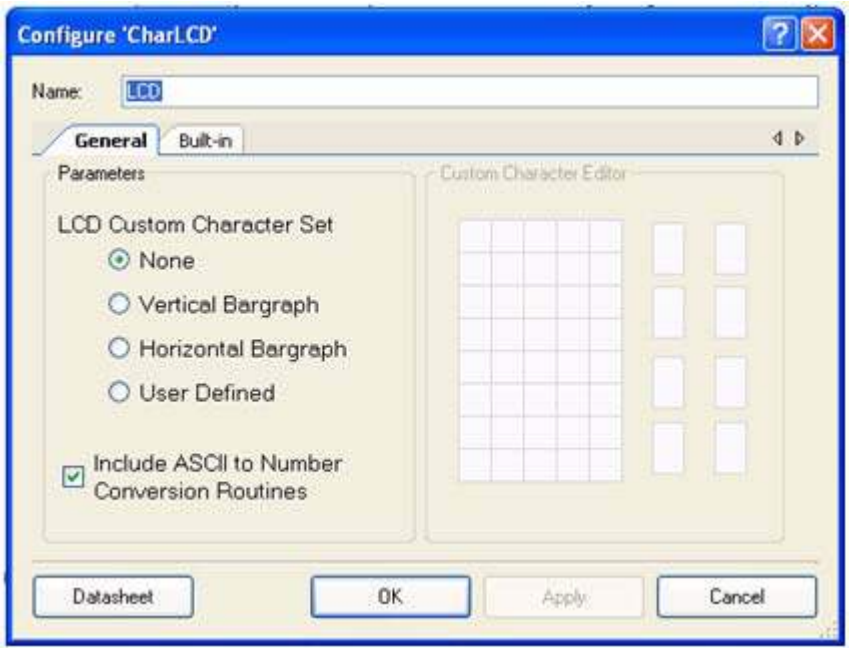
Figure 5-40. LIN Configuration: Signals Tab



5.3.5.2 Character LCD

The character LCD is used to display the user interface messages, received/transmitted data, and frame IDs.

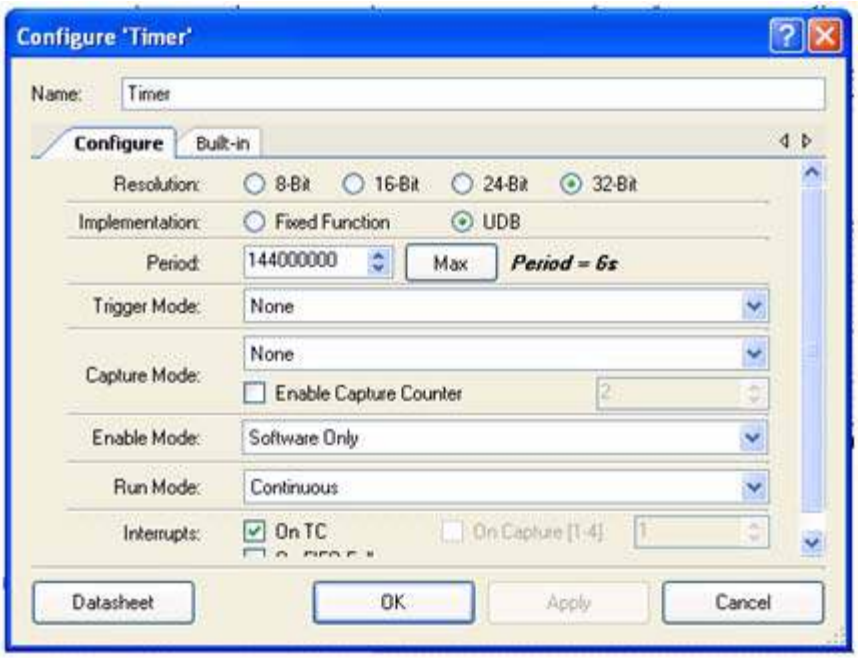
Figure 5-41. Character LCD Configuration: General Tab



5.3.5.3 Timer

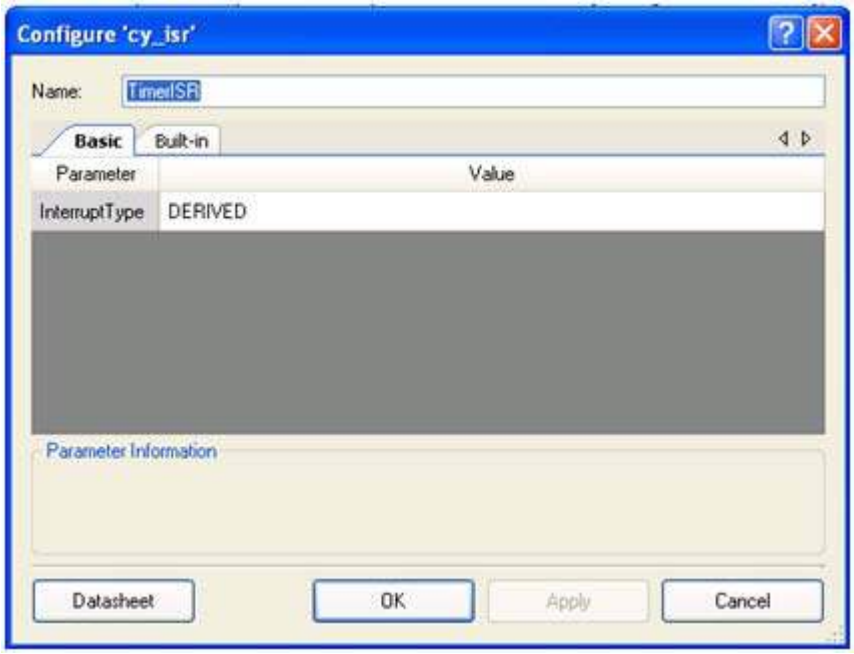
Timer uses UDB based implementation and period is configured to 6 seconds.

Figure 5-42. Timer Configuration: Configure Tab



5.3.5.4 ISR Component

Figure 5-43. ISR Component: Basic Tab



5.3.5.5 Design Wide Resources

The pin assignment in this code example matches port A of the CY8CKIT-001 DVK. To use this EBK on port B or port C of the DVK, open the code example and change the pin assignment in PSoC Creator (in the .cydwr file) to match port B or port C according to Table 4-4 on page 28. The pin assignment for this code example is shown in Figure 5-44.

Figure 5-44. Design Wide Resources

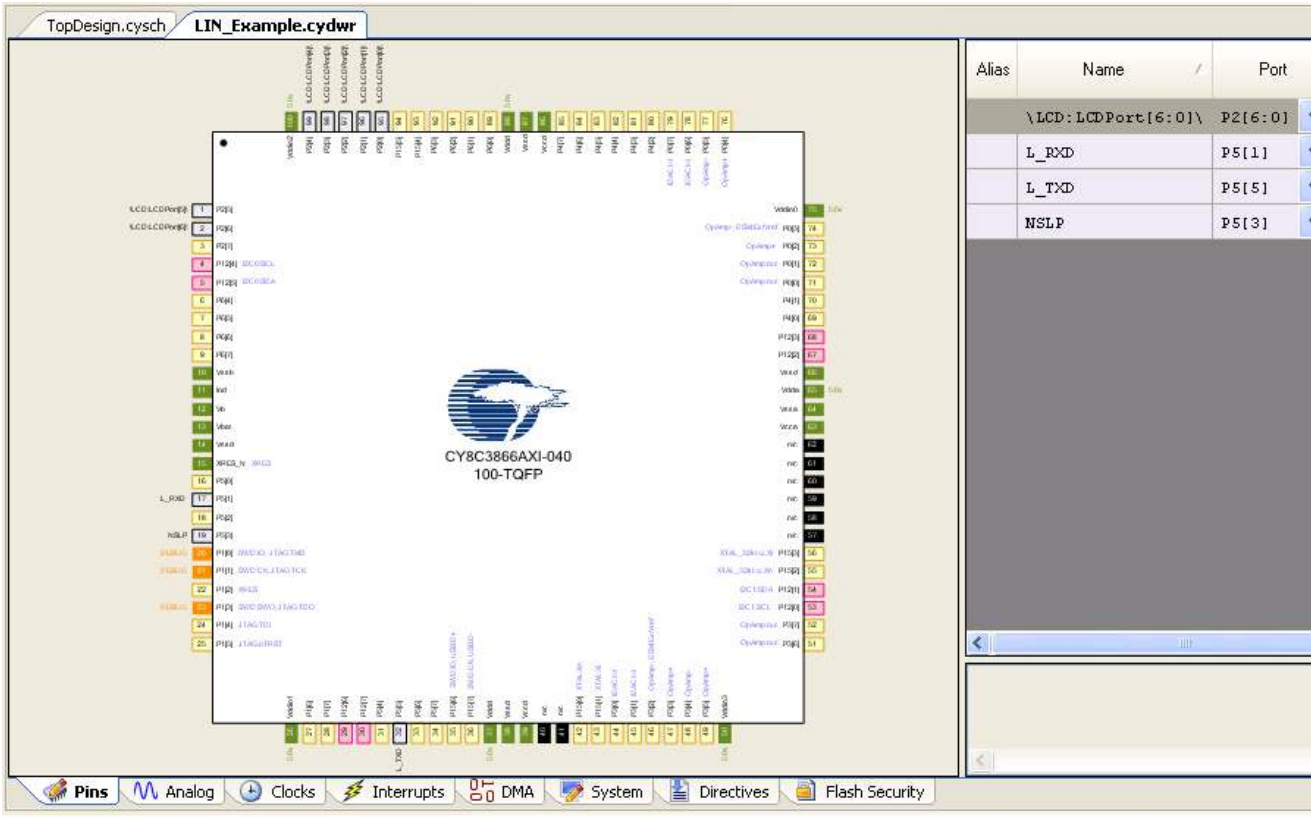


Table 5-2. Pin Assignments details of LIN Example Project

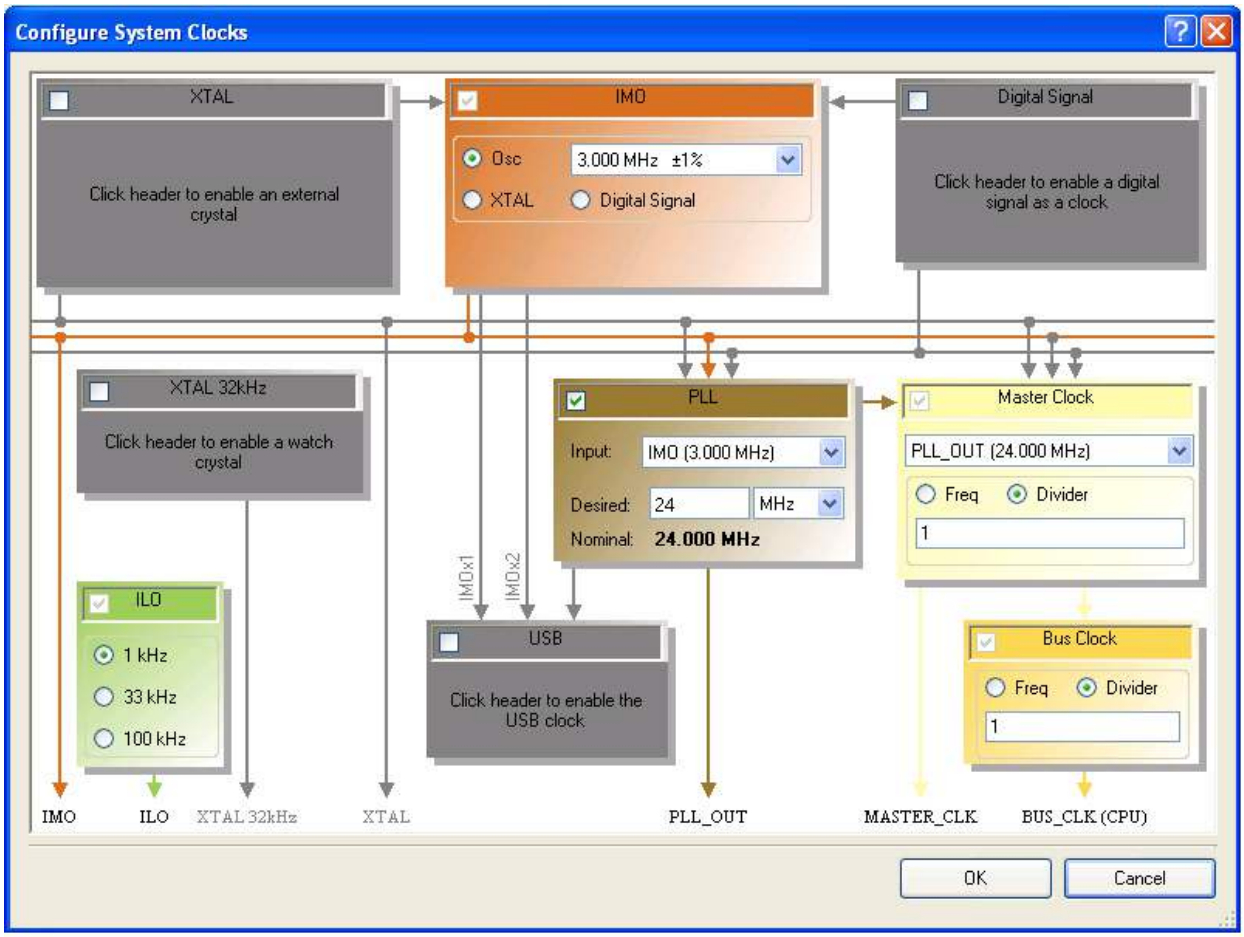
Signal	CY8CKIT-001 DVK (Port A)	CY8CKIT-030 DVK (Port E)
LCD	P2[6:0]	P2[6:0]
L_RXD	P5[1]	P0[1]
L_TXD	P5[5]	P0[5]
NSLP	P5[3]	P0[3]

Figure 5-45. Clock Settings

Add Design-Wide Clock... | Delete Design-Wide Clock | Edit Clock...

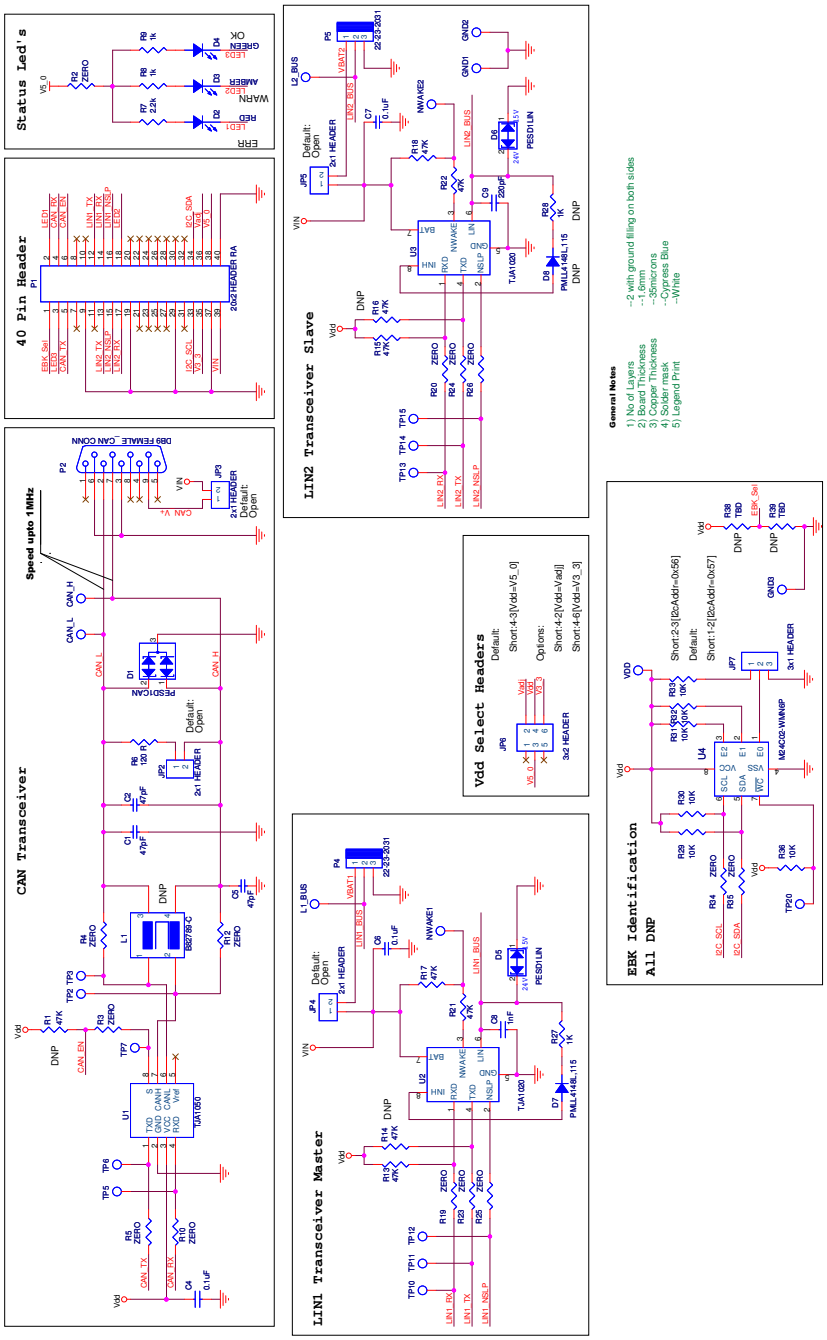
Type	Name	Domain	Desired Frequency	Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on Reset	Source Clock
System	USB_CLK	DIGITAL	48.000 MHz	? MHz	±0	-	1	☐	IMOx2
System	Digital Signal	DIGITAL	? MHz	? MHz	±0	-	0	☐	
System	XTAL 32kHz	DIGITAL	32.768 kHz	? MHz	±0	-	0	☐	
System	XTAL	DIGITAL	25.000 MHz	? MHz	±0	-	0	☐	
System	ILO	DIGITAL	? MHz	1.000 kHz	-50, +100	-	0	☑	
System	IMO	DIGITAL	3.000 MHz	3.000 MHz	±1	-	0	☑	
System	BUS_CLK (CPU)	DIGITAL	? MHz	24.000 MHz	±1	-	1	☑	MASTER_CLK
System	MASTER_CLK	DIGITAL	? MHz	24.000 MHz	±1	-	1	☑	PLL_OUT
System	PLL_OUT	DIGITAL	24.000 MHz	24.000 MHz	±1	-	0	☑	IMO
Local	LINS_IntClk	DIGITAL	307.200 kHz	307.692 kHz	±1	±14	78	☑	Auto: MASTER_CLK
Local	timer_clock	DIGITAL	? MHz	24.000 MHz	±1	-	0	☑	BUS_CLK

Figure 5-46. System Clock Configuration



A. Appendix

A.1 Schematic



A.2 Bill of Materials (BOM)

Item	Qty	Reference	Description	Manufacturer	Mfr Part Number
1	3	C1,C2,C5	CAP CER 47pF 10% 50V X7R 0603	AVX	06035A470KAT2A
2	3	C4,C6,C7	CAP CERAMIC .100UF 10% 50V X7R 0603	KEMET	C0603C104K5RACTU
3	1	C8	CAP CERAMIC 1000PF 5% 50V NPO 0603	KEMET	C0603C102J5GACTU
4	1	C9	CAP CER 220PF 5% 50V X7R 0603	AVX	06035C221JAT2A
5	1	D1	DIODE ESD PROTECTION SOT23-3(SST3)	NXP Semiconductors	PESD1CAN,215
6	1	D2	LED SUPER RED CLEAR 0805 SMD	Lite-On Inc	LTST-C170KRKT
7	1	D3	LED AMBER YELLOW CLEAR 0805 SMD	Lite-On Inc	LTST-C170KYKT
8	1	D4	LED GREEN CLEAR 0805 SMD	Lite-On Inc	LTST-C170KGKT
9	2	D5,D6	DIODE ESD PROTECTION SOD323(SC-76)	NXP Semiconductors	PESD1LIN,115
10	1	D7	DIODE SW GPP 75V 200MA SOD80C	NXP Semiconductors	PMLL4148L,115
11	4	JP2,JP3,JP4,JP5	CONN HEADR BRKWAY .100 02POS STR	Tyco Electronics	9-146280-0-02
12	1	JP6	CONN HEADER 6POS .100 STR 15AU	FCI	67996-206HLF
13	2	(JP2,JP6)	SHUNT GOLD W/HANDLE, BLACK	Kobiconn	151-8030-E
14	1	P1	CONN HEADER .100 DUAL R/A 40POS	Sullins Connector Solutions	PBC20DBAN
15	1	P2	CONN D-SUB RCPT R/A 9POS 30GOLD	AMP Division of TYCO	5747844-4
16	2	P4,P5	CONN HEADER 3POS .100 VERT TIN	Molex Inc	22-23-2031
17	2	(P4,P5)	CONN HOUSING 3POS .100 W/RAMP	Molex Inc	22-01-3037
18	12	R2,R3,R4,R5, R10,R12,R19, R20,R23,R24, R25,R26	RES ZERO OHM 1/10W 5% 0603 SMD	Panasonic -ECG	ERJ-3GEY0R00V
19	3	R8,R9,R27	RES 1.0K OHM 1/10W 1% 0603 SMD	Panasonic -ECG	ERJ-3EKF1001V
20	1	R6	RES 120 OHM 1/10W 1% 0603 SMD	Panasonic -ECG	ERJ-3EKF1200V
21	1	R7	RES2.2K OHM 1/10W 1% 0603 SMD	Panasonic -ECG	ERJ-3EKF2201V
22	6	R13,R15,R17, R18,R21,R22	RES 47.0K OHM 1/10W 1% 0603 SMD	Panasonic -ECG	ERJ-3EKF4702V
23	1	U1	IC TXRX CAN HS 5.25V 8-SOIC	NXP Semiconductors	TJA1050T/VM,118
24	2	U2,U3	IC LIN TRANSCEIVER 8-SOIC	NXP Semiconductors	TJA1020T/N1,112
25	1	TP19(GND..)	TEST POINT 43 HOLE 65 PLATED WHITE (0.040" (1.016mm) Hole Diameter)	Keystone Electronics	5002

A.3 Regulatory Compliance Information

CY8CKIT-017 has been tested and verified to comply with the following electromagnetic compatibility (EMC) regulations.

- CISPR 22 - Emissions
- EN 55022 Class A - Immunity (Europe)