Freescale Semiconductor
Application Note

Document Number:AN4850
Rev 1, Mar 2014

Getting Started with Qorivva Fast Start Kit for TRK-MPC5606B

by: Sudhakar Srinivasa

Contents

1 Introduction

This application note provides an overview of software tools provided with the Fast Start Kit for the TRK-MPC5606B. The application note also provides an example to use the software tools included in the Fast Start Kit. The example software application uses:

- RAppID init tool for configuring microcontroller and for auto generation of the initialization code
- CodeWarrior Development Studio for Microcontrollers v10.5 for building the project
- RAppID Boot loader for programming the code on to the target
- FreeMASTER utility for monitor and debug purposes

2 Overview of Qorivva Fast Start Kit for TRK-MPC5606B

The Fast Start Kit for TRK-MPC5606B contains TRK-MPC5606B evaluation board and many Freescale's software tools to help you get started with your application development. The software tools consist of:

- RAppID initialization tool,
- CodeWarrior Development Studio v10.5 (Special Edition),
- RAppID Boot loader utility,
- FreeMASTER utility,

- CodeWarrior Project Maker utility to add RAppID generated source code to an empty CodeWarrior project,
- Driver code for the MPC5606B,
- Example projects to help you to get started with the Fast Start Kit.

An overview of some of the software tools provided with Fast Start Kit is described below.

2.1 RAppID Init overview

RAppID Init is a family of graphical development tools for Freescale's MPC56xx Qorivva microcontrollers that enable the user to quickly and easily configure the microcontroller and generate complete initialization code and documentation. It is also a learning tool that can help gain understanding of the microcontroller and its peripherals. Some of the product highlights include:

- Intuitive, easy-to-use graphical user interface (GUI)
- Comprehensive initialization of the CPU, memory and peripherals
- Automatic DMA register setting from peripherals for basic modes
- Built-in consistency checks to minimize incorrect settings
- Automatic report generation of settings
- Efficient C and assembly code generation for compilers from companies such as Wind River®, Green Hills® and CodeWarrior
- Online documentation and built-in tool tips
- Installation comes with many example projects
- Generates complete infrastructure code for MCU startup
- Provisions for revision management
- Automatic date and time stamps on generated code and reports
- Modular code generation - generate code for any or all peripherals
- Option to generate code for RAM or Flash
- Flexible Initialization sequence
- Project import/export capability for distributed development teams
- Wizards for eMIOS initialization and function settings

2.2 RAppID Boot Loader utility

The RAppID Boot Loader tool is developed by Freescale and helps you develop software for Freescale microcontrollers by providing a method to update software of these microcontrollers through a serial link. The RAppID Boot Loader works with the built-in Boot Assist Module (BAM) included in the Freescale's Qorivva family of parts. The Boot Loader provides a streamlined method for programming code into FLASH or RAM on either target EVBs or custom boards. The Boot Loader has two modes of operation; for use as a stand-alone PC desktop GUI utility, or for integration with different user required tools chains through a command line interface (i.e. Eclipse Plug-in, MatLab/SimuLink etc.).

2.3 FreeMASTER utility

FreeMASTER is a user-friendly real-time debug monitor and data visualization tool for application development and information management. FreeMASTER supports completely non-intrusive monitoring of variables on a running system. You can display multiple variables changing over time on an oscilloscope-like display, or view the data in text form.

2.4 Low-level and high-level drivers

The Fast Start Kit for TRK-MPC5606B includes low-level and high-level drivers to help make the application development easier. The included drivers are for peripherals ADC, GPIO, UART, and CAN. The high level drivers included are for the Potentiometer, Photo Sensor, and System Basis Chip (SBC) that are on the TRK-MPC5606B board.

2.5 Overview of example application

The example provided in this application note demonstrates the software tools provided with the Fast Start Kit and makes use of the input buttons, LEDs, and analog inputs provided in the TRK-MPC5606B evaluation board. This example uses multiple peripherals on the MPC5606B microcontroller like ADC, SIU, PIT, eMIOS, LINFlex (UART), and CAN. The example project turns on/off the LEDs based on different input commands:

- LED1 is turned on/off based on Potentiometer input value
- LED2 is turned on/off based on Photo sensor input value
- LED3 is turned on/off based on CAN command
- LED4 is driven by PWM signal where the duty cycle of PWM output can be increased by input pressing button S3 and decreased by pressing input button S4.

The application communicates with FreeMASTER utility via UART and monitor variables used in the example project.

The next few sections in this application note describe the steps to configure microcontroller, generate code, and build and run a simple project on TRK-MPC5606B target. The figure below depicts the example software in a block diagram.

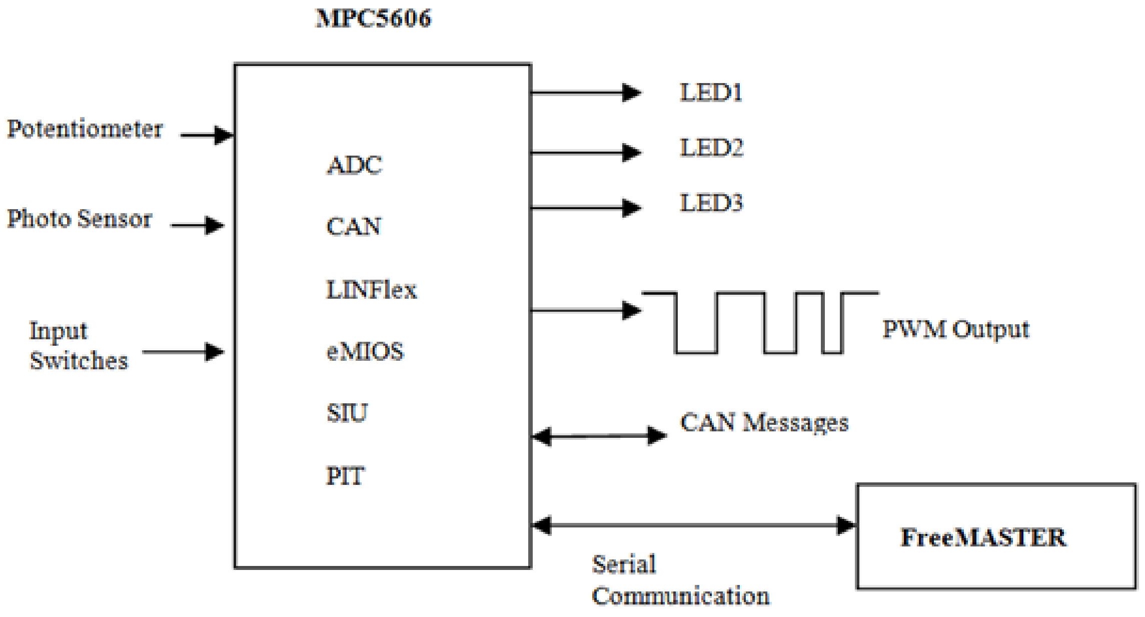


Figure 1. Block diagram of example application

3 Using RAppID tool to configure MPC5606B and generate code

This section describes the steps to configure MPC5606B microcontroller and generate initialization code for the example project using the RAppID tool.

3.1 Pin configuration

The table below shows TRK-MPC5606B board pin connections for the features used in the example project

Table 1. TRK-MPC5606B Pin connections used in example project

Feature	Pin	Comments
Potentiometer	PB4	ADC input - ANP0
Photo Sensor	PB5	ADC input ANP1
UART Tx	PB2	Connected to virtual serial port Tx
UART Rx	PB3	Connected to virtual serial port Rx
Switch S1	PE0	Not used in this example
Switch S2	PE1	Not used in this example
Switch S3	PE2	Used to increment duty cycle of LED4 PWM signal
Switch S4	PE3	Used to decrement duty cycle of LED4 PWM signal
LED 1	PE4	Turns on/off based on Potentiometer input value
LED 2	PE5	Turns on/off based on Photo sensor input value
LED 3	PE6	Turns on/off based on CAN message
LED 4	PE7	Output driven by PWM signal
CAN Tx	PC10	Connected to CAN1 Tx via SBC
CAN Rx	PC11	Connected to CAN1 Rx via SBC

The Pin configuration table above shows the relevant pin connection for the functions you will use in this project. You will use the RAppID tool to configure the pins and peripherals for this project.

To create, configure, and generate code for the project using RAppID Init:

1. Double-click the RAppID desktop icon to launch the RAppID application.

Figure 2. RAppID desktop icon

The RAppID window appears.

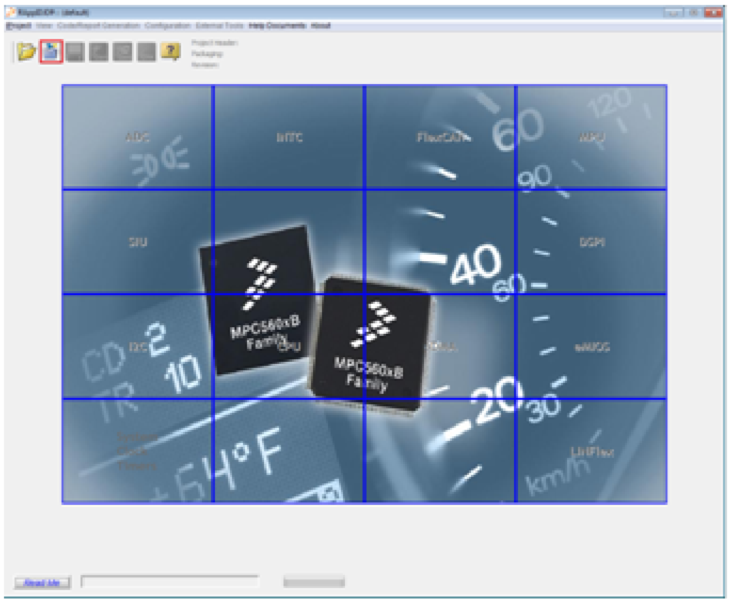
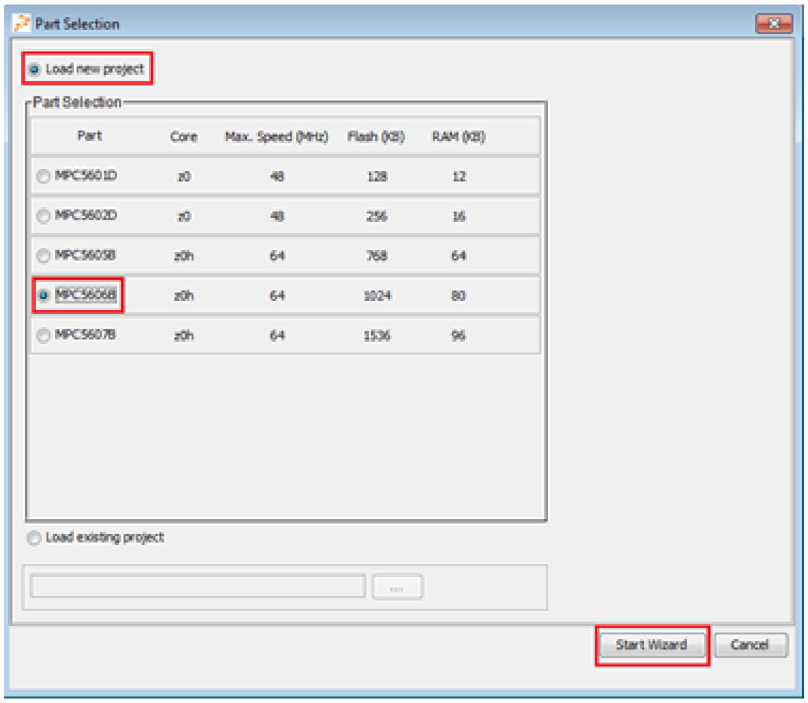


Figure 3. RAppID main window

2. Click the new project wizard button to start a new project.

3.2 Select part and package

1. Select *MPC5606B* and click the **Start Wizard** button.



Part Selection

[✓ Load new project]

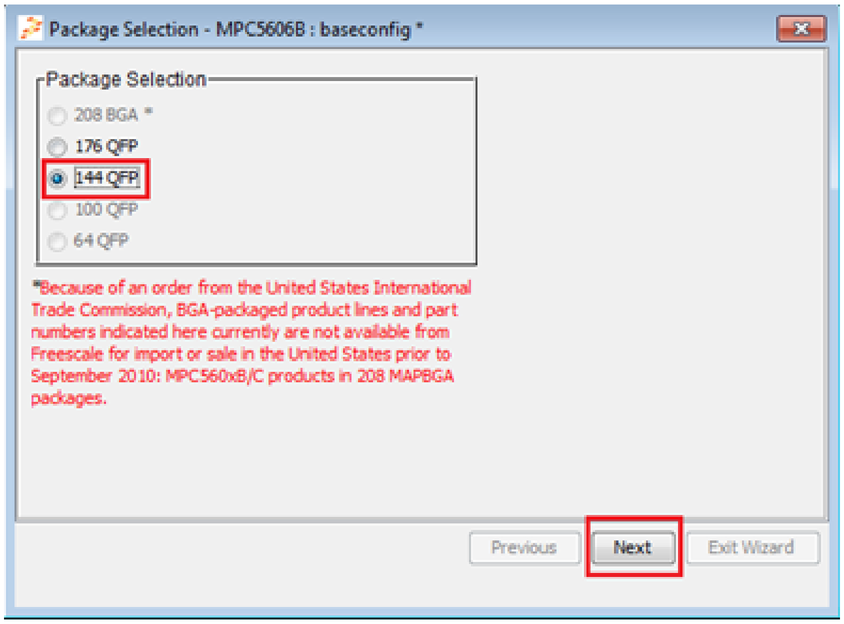
Part Selection

Part	Core	Max. Speed (MHz)	Flash (KB)	RAM (KB)
MPC5601D	z0	48	128	12
MPC5602D	z0	48	256	16
MPC5605B	z0h	64	768	64
[✓ MPC5606B]	z0h	64	1024	80
MPC5607B	z0h	64	1536	96

Load existing project

Figure 4. Part selection

2. Select the *144 QFP* package and click the **Next** button.



Package Selection

- 208 BGA *
- 176 QFP
- [✓ 144 QFP]
- 100 QFP
- 64 QFP

*Because of an order from the United States International Trade Commission, BGA-packaged product lines and part numbers indicated here currently are not available from Freescale for import or sale in the United States prior to September 2010: MPC560xB/C products in 208 MAPBGA packages.

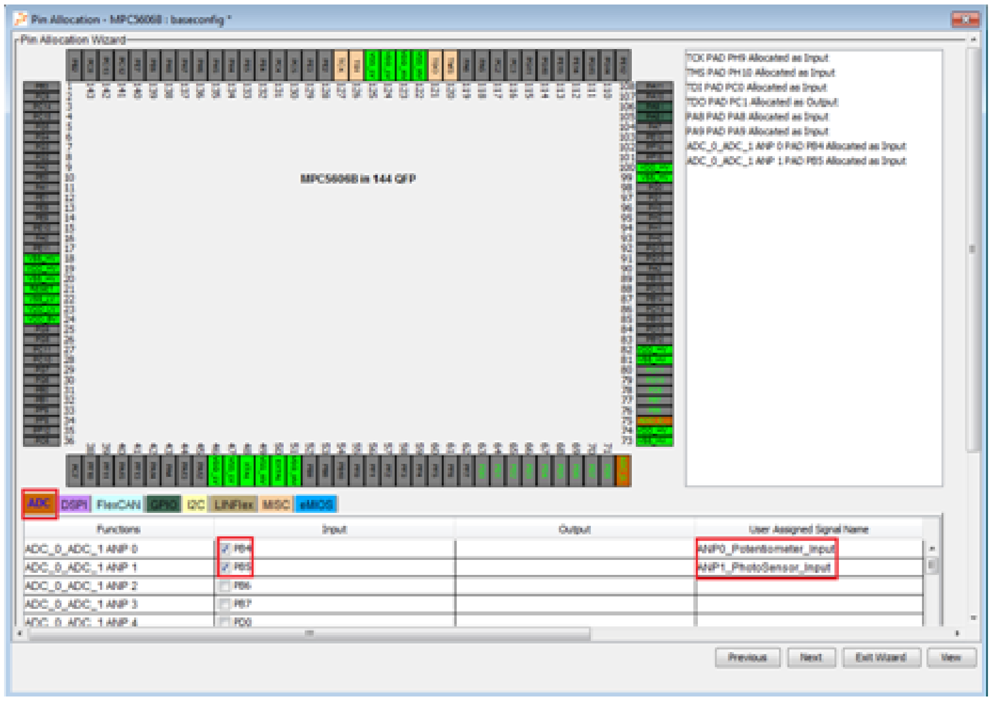
Figure 5. Package selection

3.3 Configure ADC pins

In TRK-MPC5606B, the potentiometer is connected to input ANP0 (PB4) and the photo sensor is connected to ANP1 (PB5).

To configure ADC inputs:

1. Select the ADC tab in the Pin Allocation window.
2. Select PB4 and PB5 as inputs and enter the user signal names as shown below.



TCK PAD PH9 Allocated as Input
TMS PAD PH10 Allocated as Input
TDI PAD PC0 Allocated as Input
TDO PAD PC1 Allocated as Output
PA8 PAD PA8 Allocated as Input
PA9 PAD PA9 Allocated as Input
ADC_0_ADC_1 ANP 0 PAD PB4 Allocated as Input
ADC_0_ADC_1 ANP 1 PAD PB5 Allocated as Input

Functions	Input	Output	User Assigned Signal Name
ADC_0_ADC_1 ANP 0	✓ PB4		ANP0_Potentiometer_Input
ADC_0_ADC_1 ANP 1	✓ PB5		ANP1_PhotoSensor_Input
ADC_0_ADC_1 ANP 2	PB6		
ADC_0_ADC_1 ANP 3	PB7		
ADC_0_ADC_1 ANP 4	PD0		

Figure 6. Configure ADC pins

3.4 Configure DSPI pins

TRK-MPC5606B contains MCZ3390S5EK system basis chip (SBC) with integrated CAN transceiver and LIN 2.0 interface. Since DSPI 1 of the MPC5606B is connected to SBC, you can use the DSPI 1 peripheral to configure the SBC and enable CAN communication by sending appropriate commands via DSPI 1.

The figure below shows the connections between the SBC and DSPI 1 peripheral.

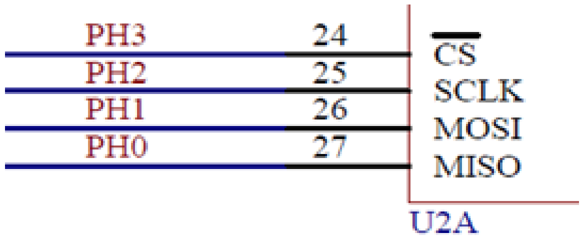
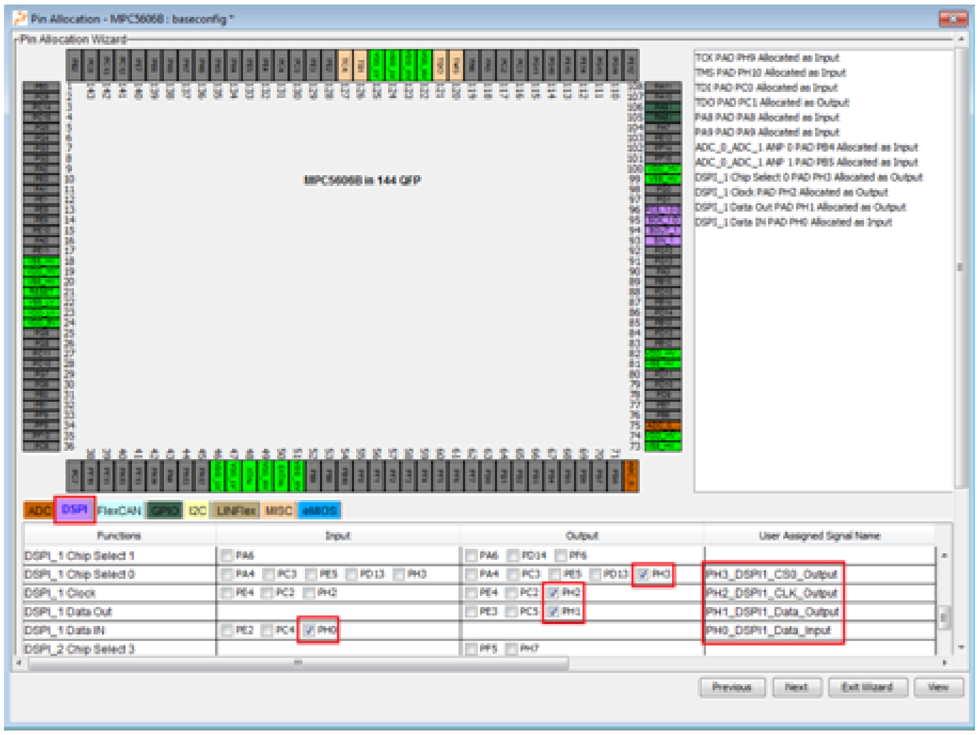


Figure 7. Connection between SBC and DSPI 1

Configure DSPI 1 pins PH3, PH2, PH1, and PH0 using RAppID as shown below.



TCK PAD PH9 Allocated as Input
TMS PAD PH10 Allocated as Input
TDI PAD PC0 Allocated as Input
TDO PAD PC1 Allocated as Output
PA8 PAD PA8 Allocated as Input
PA9 PAD PA9 Allocated as Input
ADC_0_ADC_1 ANP 0 PAD PB4 Allocated as Input
ADC_0_ADC_1 ANP 1 PAD PB5 Allocated as Input
DSPI_1 Chip Select 0 PAD PH3 Allocated as Output
DSPI_1 Clock PAD PH2 Allocated as Output
DSPI_1 Data Out PAD PH1 Allocated as Output
DSPI_1 Data IN PAD PH0 Allocated as Input

Functions	Input	Output	User Assigned Signal Name
DSPI_1 Chip Select 1	☐ PA6	☐ PA6 ☐ PD14 ☐ PF6	
DSPI_1 Chip Select 0	☐ PA4 ☐ PC3 ☐ PE5 ☐ PD13 ☐ PH3	☐ PA4 ☐ PC3 ☐ PE5 ☐ PD13 ✓ PH3	PH3_DSPI1_CS0_Output
DSPI_1 Clock	☐ PE4 ☐ PC2 ☐ PH2	☐ PE4 ☐ PC2 ✓ PH2	PH2_DSPI1_CLK_Output
DSPI_1 Data Out		☐ PE3 ☐ PC5 ✓ PH1	PH1_DSPI1_Data_Output
DSPI_1 Data IN	☐ PE2 ☐ PC4 ✓ PH0		PH0_DSPI1_Data_Input
DSPI_2 Chip Select 3		☐ PF5 ☐ PH7	

Figure 8. Configure DSPI 1 pins

3.5 Configure FlexCAN pins

The CAN TX and CAN RX pins of the SBC are connected to the pins PC10 and PC11 of CAN 1 peripheral of the microcontroller as shown in the figure below.

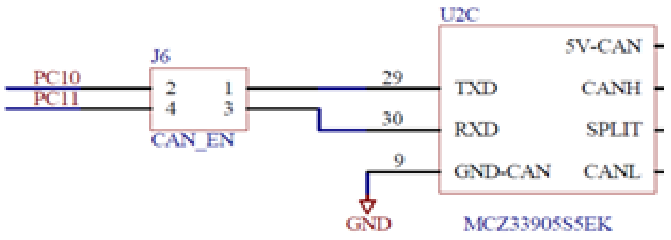
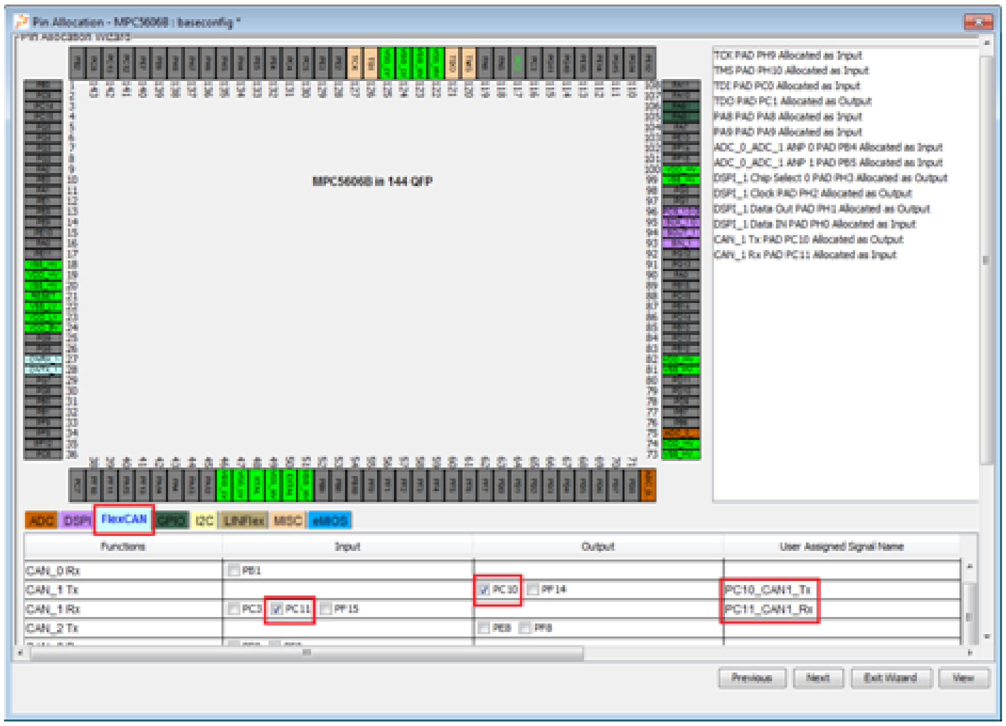


Figure 9. CAN TX and CAN RX pins of SBC connected to pins PC10 and PC11

Configure FlexCAN 1 pins PC10 and PC11 using RAppID as shown below.



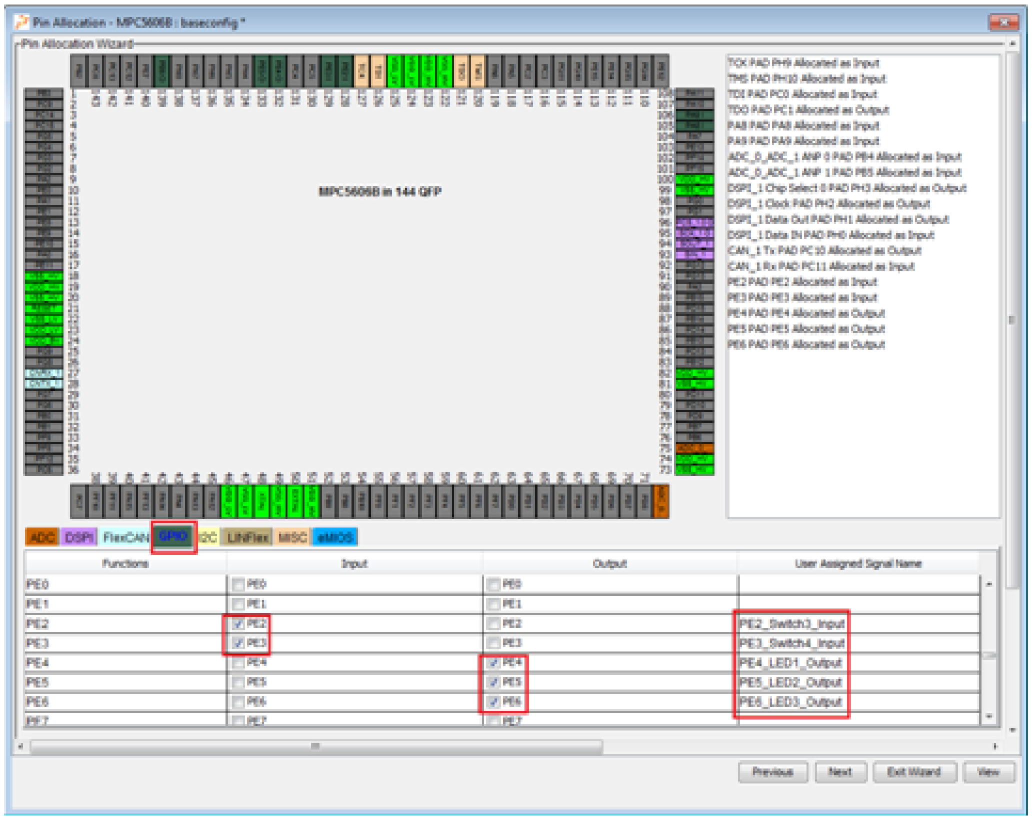
TCK PAD PH9 Allocated as Input
TMS PAD PH10 Allocated as Input
TDI PAD PC0 Allocated as Input
TDO PAD PC1 Allocated as Output
PA8 PAD PA8 Allocated as Input
PA9 PAD PA9 Allocated as Input
ADC_0_ADC_1 ANP 0 PAD PB4 Allocated as Input
ADC_0_ADC_1 ANP 1 PAD PB5 Allocated as Input
DSPI_1 Chip Select 0 PAD PH3 Allocated as Output
DSPI_1 Clock PAD PH2 Allocated as Output
DSPI_1 Data Out PAD PH1 Allocated as Output
DSPI_1 Data IN PAD PH0 Allocated as Input
CAN_1 Tx PAD PC10 Allocated as Output
CAN_1 Rx PAD PC11 Allocated as Input

Functions	Input	Output	User Assigned Signal Name
CAN_0 Rx	☐ PB1		
CAN_1 Tx		☑ PC10 ☐ PF14	PC10_CAN1_Tx
CAN_1 Rx	☐ PC3 ☑ PC11 ☐ PF15		PC11_CAN1_Rx
CAN_2 Tx		☐ PE8 ☐ PF8	

Figure 10. Configure FlexCAN pins

3.6 Configure GPIO pins

In this example, you use Switch S3 and S4 as inputs and LED1, LED2, and LED3 as general purpose outputs. The switches S3 and S4 are connected to PE2 and PE3 pins of the microcontroller and LED1, LED2, and LED3 are connected to PE4, PE5, and PE6 pins of microcontroller. Configure GPIO pins using RAppID as shown below.



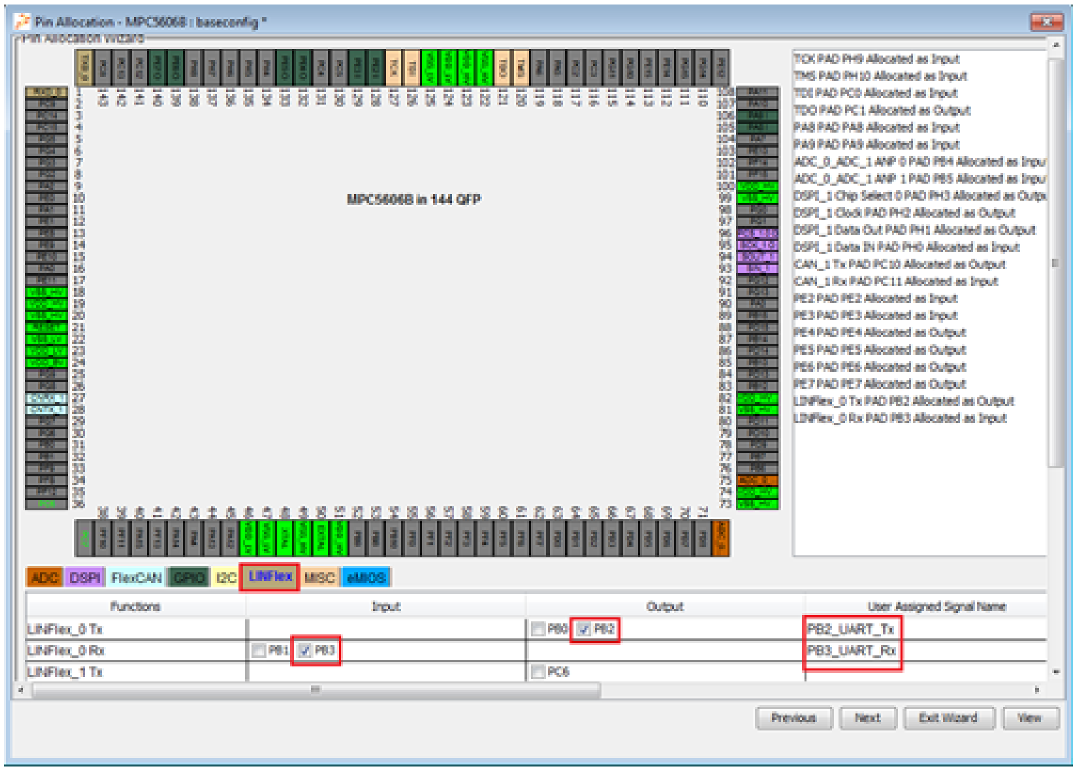
TCK PAD PH9 Allocated as Input
TMS PAD PH10 Allocated as Input
TDI PAD PC0 Allocated as Input
TDO PAD PC1 Allocated as Output
PA8 PAD PA8 Allocated as Input
PA9 PAD PA9 Allocated as Input
ADC_0_ADC_1 ANP 0 PAD PB4 Allocated as Input
ADC_0_ADC_1 ANP 1 PAD PB5 Allocated as Input
DSPI_1 Chip Select 0 PAD PH3 Allocated as Output
DSPI_1 Clock PAD PH2 Allocated as Output
DSPI_1 Data Out PAD PH1 Allocated as Output
DSPI_1 Data IN PAD PH0 Allocated as Input
CAN_1 Tx PAD PC10 Allocated as Output
CAN_1 Rx PAD PC11 Allocated as Input
PE2 PAD PE2 Allocated as Input
PE3 PAD PE3 Allocated as Input
PE4 PAD PE4 Allocated as Output
PE5 PAD PE5 Allocated as Output
PE6 PAD PE6 Allocated as Output

Functions	Input	Output	User Assigned Signal Name
PE0	PE0	PE0	
PE1	PE1	PE1	
PE2	✓ PE2	PE2	PE2_Switch3_Input
PE3	✓ PE3	PE3	PE3_Switch4_Input
PE4	PE4	✓ PE4	PE4_LED1_Output
PE5	PE5	✓ PE5	PE5_LED2_Output
PE6	PE6	✓ PE6	PE6_LED3_Output
PE7	PE7	PE7	

Figure 11. Configure GPIO pins

3.7 Configure LINFlex (UART) pins

In this example, use the virtual serial port of TRK-MPC5606B board for serial communication. The PB2 and PB3 pins of microcontroller in TRK-MPC5606B board are connected to TX and RX pins of virtual serial port. Configure LINFlex 0 pins using RAppID as shown below.



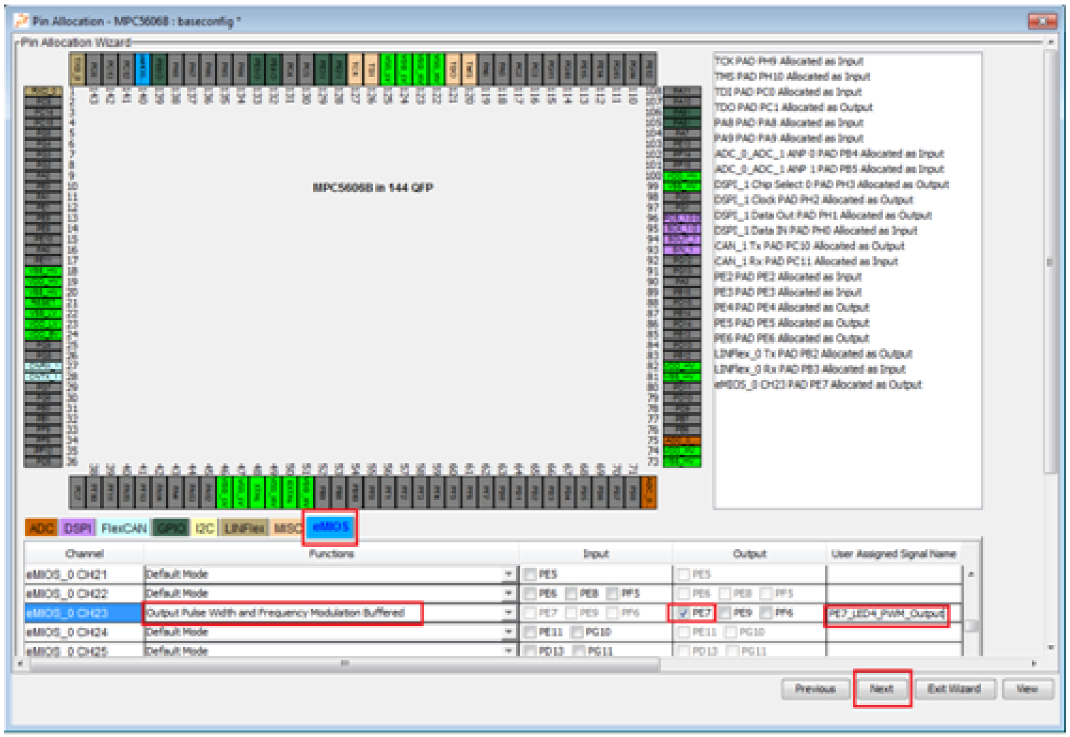
TCK PAD PH9 Allocated as Input
TMS PAD PH10 Allocated as Input
TDI PAD PC0 Allocated as Input
TDO PAD PC1 Allocated as Output
PA8 PAD PA8 Allocated as Input
PA9 PAD PA9 Allocated as Input
ADC_0_ADC_1 ANP 0 PAD PB4 Allocated as Input
ADC_0_ADC_1 ANP 1 PAD PB5 Allocated as Input
DSPI_1 Chip Select 0 PAD PH3 Allocated as Output
DSPI_1 Clock PAD PH2 Allocated as Output
DSPI_1 Data Out PAD PH1 Allocated as Output
DSPI_1 Data IN PAD PH0 Allocated as Input
CAN_1 Tx PAD PC10 Allocated as Output
CAN_1 Rx PAD PC11 Allocated as Input
PE2 PAD PE2 Allocated as Input
PE3 PAD PE3 Allocated as Input
PE4 PAD PE4 Allocated as Output
PE5 PAD PE5 Allocated as Output
PE6 PAD PE6 Allocated as Output
PE7 PAD PE7 Allocated as Output
LINFlex_0 Tx PAD PB2 Allocated as Output
LINFlex_0 Rx PAD PB3 Allocated as Input

Functions	Input	Output	User Assigned Signal Name
LINFlex_0 Tx		☐ PB0 ☑ PB2	PB2_UART_Tx
LINFlex_0 Rx	☐ PB1 ☑ PB3		PB3_UART_Rx
LINFlex_1 Tx		☐ PC6	

Figure 12. Configure LINFlex (UART) pins

3.8 Configure eMIOS pins

Use LED4 as PWM output in this example. LED4 is connected to PE7 pin of the microcontroller. Configure eMIOS function and pin using RAppID as shown below.



TCK PAD PH9 Allocated as Input
TMS PAD PH10 Allocated as Input
TDI PAD PC0 Allocated as Input
TDO PAD PC1 Allocated as Output
PA8 PAD PA8 Allocated as Input
PA9 PAD PA9 Allocated as Input
ADC_0_ADC_1 ANP 0 PAD PB4 Allocated as Input
ADC_0_ADC_1 ANP 1 PAD PB5 Allocated as Input
DSPI_1 Chip Select 0 PAD PH3 Allocated as Output
DSPI_1 Clock PAD PH2 Allocated as Output
DSPI_1 Data Out PAD PH1 Allocated as Output
DSPI_1 Data IN PAD PH0 Allocated as Input
CAN_1 Tx PAD PC10 Allocated as Output
CAN_1 Rx PAD PC11 Allocated as Input
PE2 PAD PE2 Allocated as Input
PE3 PAD PE3 Allocated as Input
PE4 PAD PE4 Allocated as Output
PE5 PAD PE5 Allocated as Output
PE6 PAD PE6 Allocated as Output
LINFlex_0 Tx PAD PB2 Allocated as Output
LINFlex_0 Rx PAD PB3 Allocated as Input
eMIOS_0 CH23 PAD PE7 Allocated as Output

Channel	Functions	Input	Output	User Assigned Signal Name
eMIOS_0 CH21	Default Mode	PE5	PE5	
eMIOS_0 CH22	Default Mode	PE6 PE8 PF5	PE6 PE8 PF5	
eMIOS_0 CH23	Output Pulse Width and Frequency Modulation Buffered	PE7 PE9 PF6	✓ PE7 PE9 PF6	PE7_LED4_PWM_Output
eMIOS_0 CH24	Default Mode	PE11 PG10	PE11 PG10	
eMIOS_0 CH25	Default Mode	PD13 PG11	PD13 PG11	

Figure 13. Configure eMIOS pins

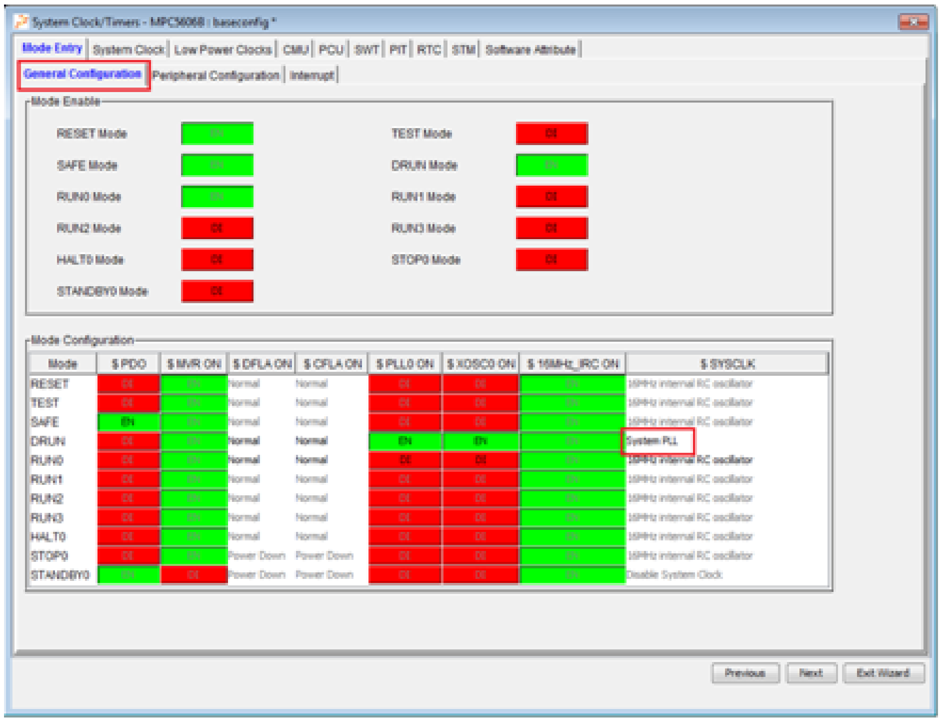
Now we are finished with the entire pin configuration required for this project. Next, configure Mode Entry. Select *Next* button and skip Core Configuration and Memory Protection Unit configuration windows by selecting *Next* button for the next two windows.

3.9 Mode entry configuration

RAppID generates code to put the target in DRUN mode. This example uses PLL as clock source in the DRUN mode.

To configure mode entry:

1. Select the General Configuration tab.
2. Select System PLL from the drop-down menu as SYSCLK source for DRUN.


System Clock/Timers - MPC5606B : baseconfig *
Mode Entry | System Clock | Low Power Clocks | CMU | PCU | SWT | PIT | RTC | STM | Software Attribute
General Configuration | Peripheral Configuration | Interrupt
Previous
Next
Exit Wizard

Mode Enable

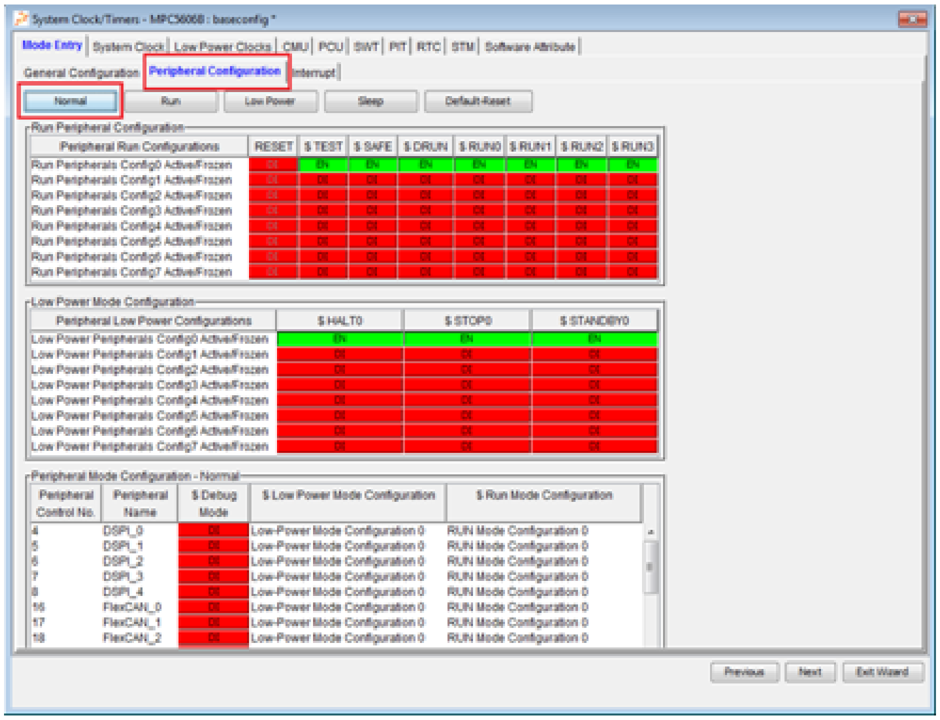
RESET Mode	EN	TEST Mode	DI
SAFE Mode	EN	DRUN Mode	EN
RUN0 Mode	EN	RUN1 Mode	DI
RUN2 Mode	DI	RUN3 Mode	DI
HALT0 Mode	DI	STOP0 Mode	DI
STANDBY0 Mode	DI		

Mode Configuration

Mode	$ PDO	$ MVR ON	$ DFLA ON	$ CFLA ON	$ PLL0 ON	$ XOSC0 ON	$ 16MHz_IRC ON	$ SYSCLK
RESET	DI	EN	Normal	Normal	DI	DI	EN	16MHz internal RC oscillator
TEST	DI	EN	Normal	Normal	DI	DI	EN	16MHz internal RC oscillator
SAFE	EN	EN	Normal	Normal	DI	DI	EN	16MHz internal RC oscillator
DRUN	DI	EN	Normal	Normal	EN	EN	EN	System PLL
RUN0	DI	EN	Normal	Normal	DI	DI	EN	16MHz internal RC oscillator
RUN1	DI	EN	Normal	Normal	DI	DI	EN	16MHz internal RC oscillator
RUN2	DI	EN	Normal	Normal	DI	DI	EN	16MHz internal RC oscillator
RUN3	DI	EN	Normal	Normal	DI	DI	EN	16MHz internal RC oscillator
HALT0	DI	EN	Normal	Normal	DI	DI	EN	16MHz internal RC oscillator
STOP0	DI	EN	Power Down	Power Down	DI	DI	EN	16MHz internal RC oscillator
STANDBY0	EN	DI	Power Down	Power Down	DI	DI	EN	Disable System Clock

Figure 14. Mode Entry clock configuration

3. Configure peripheral behavior during run modes and non-run modes. RAppID provides an easy way to configure peripheral behavior during these two modes by providing one click buttons: Normal, Run, Low Power, and Stop modes. The Normal button sets Run Peripheral Configuration 0 and Low Power Configuration 0 to be enabled in all modes and selects Configuration 0 during run and non-run modes.
4. Click the Normal button to select Peripheral Configuration 0 for run and non-run modes.


System Clock/Timers - MPC5606B : baseconfig *
Mode Entry | System Clock | Low Power Clocks | CMU | PCU | SWT | PIT | RTC | STM | Software Attribute
General Configuration | Peripheral Configuration | Interrupt
Normal
Run
Low Power
Sleep
Default-Reset
Previous
Next
Exit Wizard

Run Peripheral Configuration

Peripheral Run Configurations	RESET	$ TEST	$ SAFE	$ DRUN	$ RUN0	$ RUN1	$ RUN2	$ RUN3
Run Peripherals Config0 Active/Frozen	DI	EN	EN	EN	EN	EN	EN	EN
Run Peripherals Config1 Active/Frozen	DI	DI	DI	DI	DI	DI	DI	DI
Run Peripherals Config2 Active/Frozen	DI	DI	DI	DI	DI	DI	DI	DI
Run Peripherals Config3 Active/Frozen	DI	DI	DI	DI	DI	DI	DI	DI
Run Peripherals Config4 Active/Frozen	DI	DI	DI	DI	DI	DI	DI	DI
Run Peripherals Config5 Active/Frozen	DI	DI	DI	DI	DI	DI	DI	DI
Run Peripherals Config6 Active/Frozen	DI	DI	DI	DI	DI	DI	DI	DI
Run Peripherals Config7 Active/Frozen	DI	DI	DI	DI	DI	DI	DI	DI

Low Power Mode Configuration

Peripheral Low Power Configurations	$ HALT0	$ STOP0	$ STANDBY0
Low Power Peripherals Config0 Active/Frozen	EN	EN	EN
Low Power Peripherals Config1 Active/Frozen	DI	DI	DI
Low Power Peripherals Config2 Active/Frozen	DI	DI	DI
Low Power Peripherals Config3 Active/Frozen	DI	DI	DI
Low Power Peripherals Config4 Active/Frozen	DI	DI	DI
Low Power Peripherals Config5 Active/Frozen	DI	DI	DI
Low Power Peripherals Config6 Active/Frozen	DI	DI	DI
Low Power Peripherals Config7 Active/Frozen	DI	DI	DI

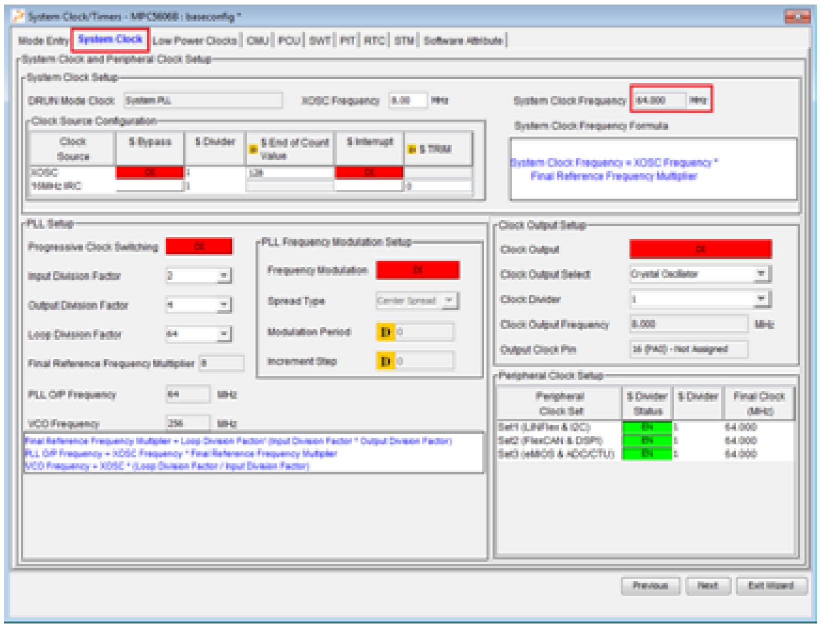
Peripheral Mode Configuration - Normal

Peripheral Control No.	Peripheral Name	$ Debug Mode	$ Low Power Mode Configuration	$ Run Mode Configuration
4	DSPI_0	DI	Low-Power Mode Configuration 0	RUN Mode Configuration 0
5	DSPI_1	DI	Low-Power Mode Configuration 0	RUN Mode Configuration 0
6	DSPI_2	DI	Low-Power Mode Configuration 0	RUN Mode Configuration 0
7	DSPI_3	DI	Low-Power Mode Configuration 0	RUN Mode Configuration 0
8	DSPI_4	DI	Low-Power Mode Configuration 0	RUN Mode Configuration 0
16	FlexCAN_0	DI	Low-Power Mode Configuration 0	RUN Mode Configuration 0
17	FlexCAN_1	DI	Low-Power Mode Configuration 0	RUN Mode Configuration 0
18	FlexCAN_2	DI	Low-Power Mode Configuration 0	RUN Mode Configuration 0

Figure 15. Mode Entry peripheral configuration

3.10 System clock

In the Mode Entry configuration window, when you select System PLL as the DRUN clock, the system clock is set to 64 MHz by default. In this example, you will use system clock of 64MHz, so no changes are required in the System Clock tab.



System Clock and Peripheral Clock Setup

System Clock Setup

DRUN Mode Clock: System PLL
XOSC Frequency: 8.00 MHz
System Clock Frequency: 64.000 MHz

Clock Source Configuration

Clock Source	$ Bypass	$ Divider	$ End of Count Value	$ Interrupt	$ TRIM
XOSC	DI	1	128	DI	
16MHz IRC		1			0

System Clock Frequency Formula

System Clock Frequency = XOSC Frequency * Final Reference Frequency Multiplier

PLL Setup

Progressive Clock Switching: DI
Input Division Factor: 2
Output Division Factor: 4
Loop Division Factor: 64
Final Reference Frequency Multiplier: 8
PLL O/P Frequency: 64 MHz
VCO Frequency: 256 MHz

PLL Frequency Modulation Setup

Frequency Modulation: DI
Spread Type: Center Spread
Modulation Period: 0
Increment Step: 0

Final Reference Frequency Multiplier = Loop Division Factor/ (Input Division Factor * Output Division Factor)
PLL O/P Frequency = XOSC Frequency * Final Reference Frequency Multiplier
VCO Frequency = XOSC * (Loop Division Factor / Input Division Factor)

Clock Output Setup

Clock Output: DI
Clock Output Select: Crystal Oscillator
Clock Divider: 1
Clock Output Frequency: 8.000 MHz
Output Clock Pin: 16 (PA0) - Not Assigned

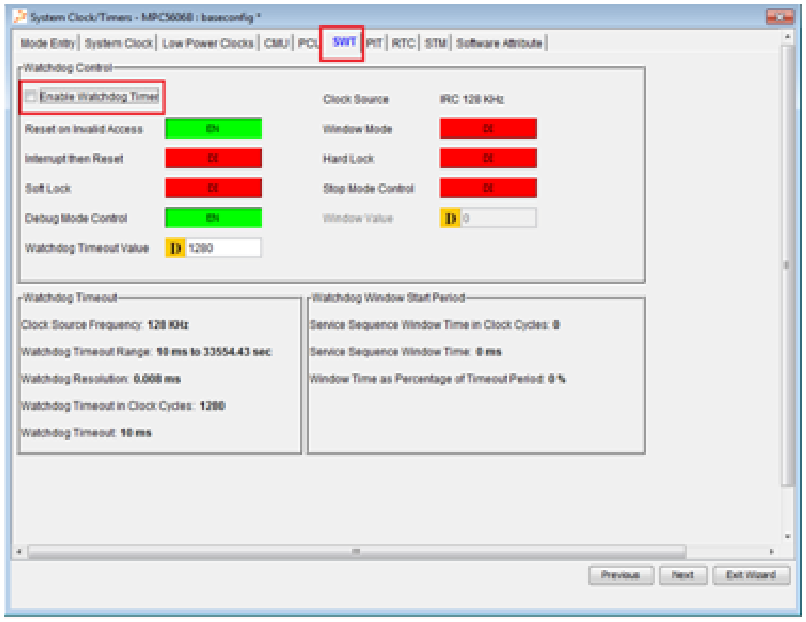
Peripheral Clock Setup

Peripheral Clock Set	$ Divider Status	$ Divider	Final Clock (MHz)
Set1 (LINFlex & I2C)	EN	1	64.000
Set2 (FlexCAN & DSPI)	EN	1	64.000
Set3 (eMIOS & ADC/CTU)	EN	1	64.000

Figure 16. System clock configuration

3.11 Watchdog configuration

By default, the watchdog is enabled in MPC5606B. This example does not use the Watchdog feature Thus, you need to disable the Watchdog Timer in SWT tab as shown below.



Watchdog Control

[ ] Enable Watchdog Timer

		Clock Source	IRC 128 KHz
Reset on Invalid Access	EN	Window Mode	DI
Interrupt then Reset	DI	Hard Lock	DI
Soft Lock	DI	Stop Mode Control	DI
Debug Mode Control	EN	Window Value	0
Watchdog Timeout Value	1280		

Watchdog Timeout

Clock Source Frequency: **128 KHz**

Watchdog Timeout Range: **10 ms to 33554.43 sec**

Watchdog Resolution: **0.008 ms**

Watchdog Timeout in Clock Cycles: **1280**

Watchdog Timeout: **10 ms**

Watchdog Window Start Period

Service Sequence Window Time in Clock Cycles: **0**

Service Sequence Window Time: **0 ms**

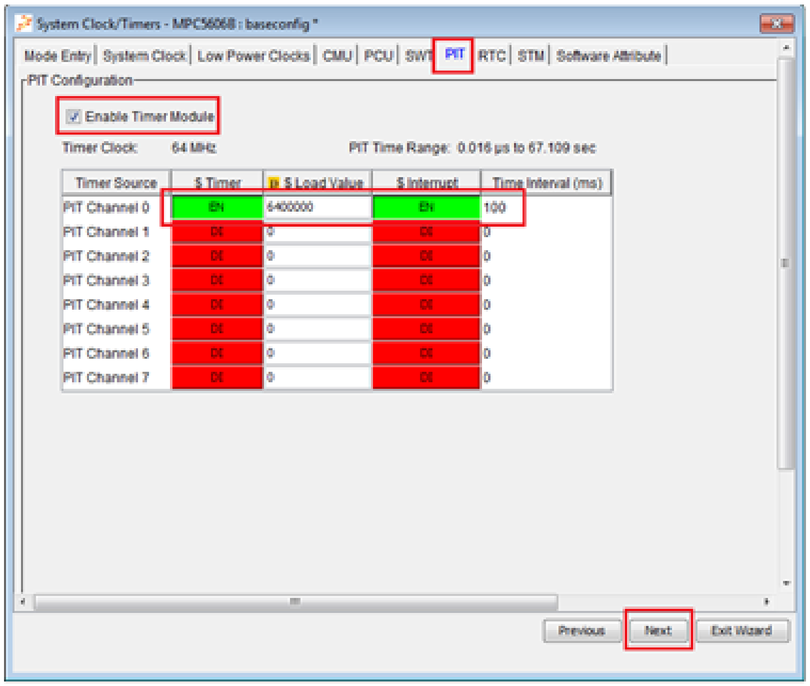
Window Time as Percentage of Timeout Period: **0 %**

Figure 17. SWT configuration

3.12 PIT configuration

You will use the PIT channel 0 interrupt to check the state of input switches S3 and S4 and increment/decrement PWM duty cycle of LED4 output based on these two switch state. Configure PIT interrupt at 100 ms using RAppID as follows:

1. Enable timer module
2. Enable channel 0 Timer and enter the Load Value as 6,400,000. This will result in PIT timeout value at 100 ms with system time at 64 MHz.
3. Enable Channel 0 interrupt.
4. This completes the basic system configuration. Click Next to begin Peripheral configuration.



PIT Configuration

☑ Enable Timer Module

Timer Clock: 64 MHz

PIT Time Range: 0.016 µs to 67.109 sec

Timer Source	$ Timer	$ Load Value	$ Interrupt	Time Interval (ms)
PIT Channel 0	EN	6400000	EN	100
PIT Channel 1	DI	0	DI	0
PIT Channel 2	DI	0	DI	0
PIT Channel 3	DI	0	DI	0
PIT Channel 4	DI	0	DI	0
PIT Channel 5	DI	0	DI	0
PIT Channel 6	DI	0	DI	0
PIT Channel 7	DI	0	DI	0

Figure 18. PIT configuration

3.13 Peripheral configuration

In the next window, RAppID displays all the peripherals you need to configure, based on the pins configured in the previous steps. To start configuring DSPI peripheral, select the DSPI tile shown below.

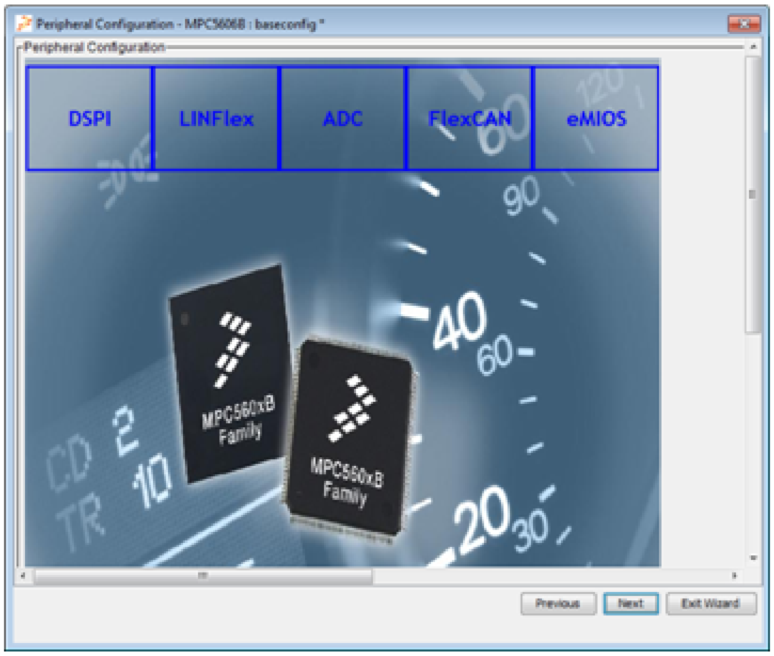
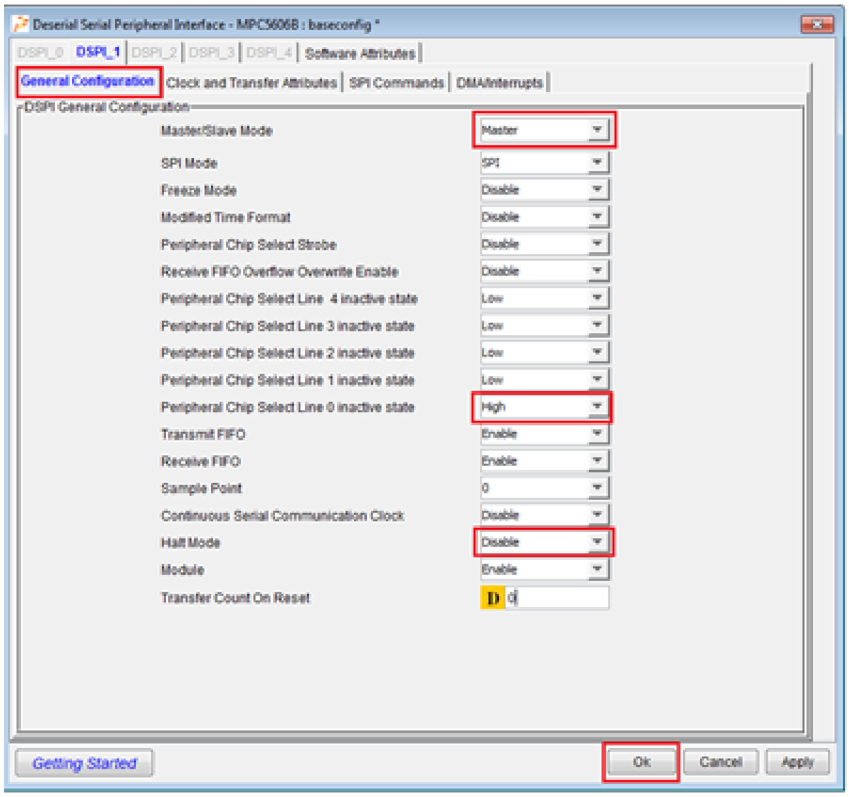


Figure 19. Select DSPI

3.14 DSPI 1 configuration

1. To send commands to SBC, set DSPI 1 to master mode.
2. Configure DSPI 1 peripheral as shown below.
3. Select Master mode, set Chip select 0 inactive state to *High* and *Disable* Halt mode.
4. Click *OK* to finish the DSPI 1 configuration



Master/Slave Mode	Master
SPI Mode	SPI
Freeze Mode	Disable
Modified Time Format	Disable
Peripheral Chip Select Strobe	Disable
Receive FIFO Overflow Overwrite Enable	Disable
Peripheral Chip Select Line 4 inactive state	Low
Peripheral Chip Select Line 3 inactive state	Low
Peripheral Chip Select Line 2 inactive state	Low
Peripheral Chip Select Line 1 inactive state	Low
Peripheral Chip Select Line 0 inactive state	High
Transmit FIFO	Enable
Receive FIFO	Enable
Sample Point	0
Continuous Serial Communication Clock	Disable
Halt Mode	Disable
Module	Enable
Transfer Count On Reset	D 0

Figure 20. DSPI configuration

3.15 LINFlex (UART) configuration

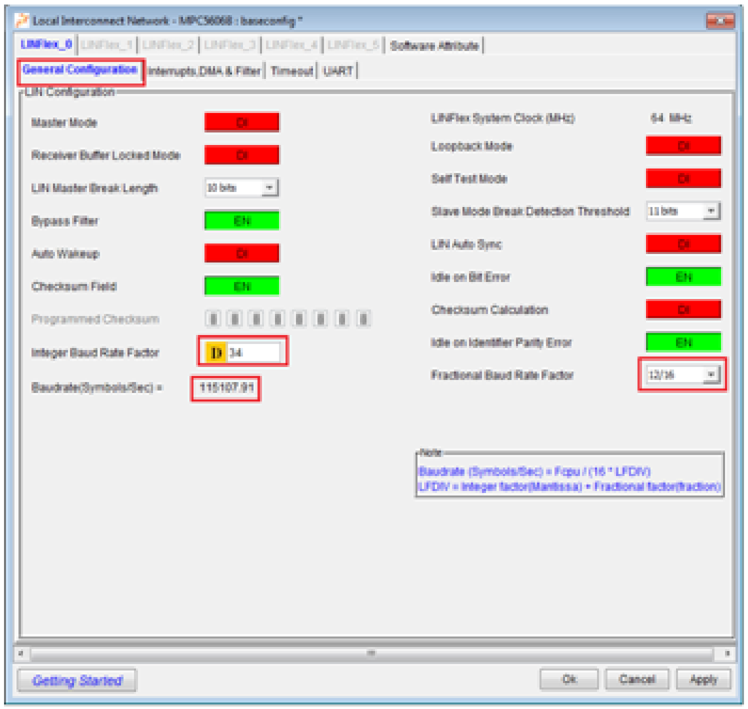
To configure the LINFlex peripheral:

1. Select LINFlex tile from the Peripheral Configuration window.

   You will use the virtual serial port of TRK-MPC5606B board to communicate with FreeMASTER utility at a baud rate of 115,200.

2. To select baud rate of 115,200, set Integer Baud Rate Factor to 34 and Fractional Baud Rate Factor to 12/16.

   When Baud Rate Factor and Fractional Baud Rate Factor values are selected, RAppID automatically calculates and displays the resulting baud rate. As indicated in the figure below, the resulting baud rate is 115,107 which is close enough to the required baud rate of 115,200.



Local Interconnect Network - MPC5606B : baseconfig *

LINFlex_0 | LINFlex_1 | LINFlex_2 | LINFlex_3 | LINFlex_4 | LINFlex_5 | Software Attribute

General Configuration | Interrupts,DMA & Filter | Timeout | UART

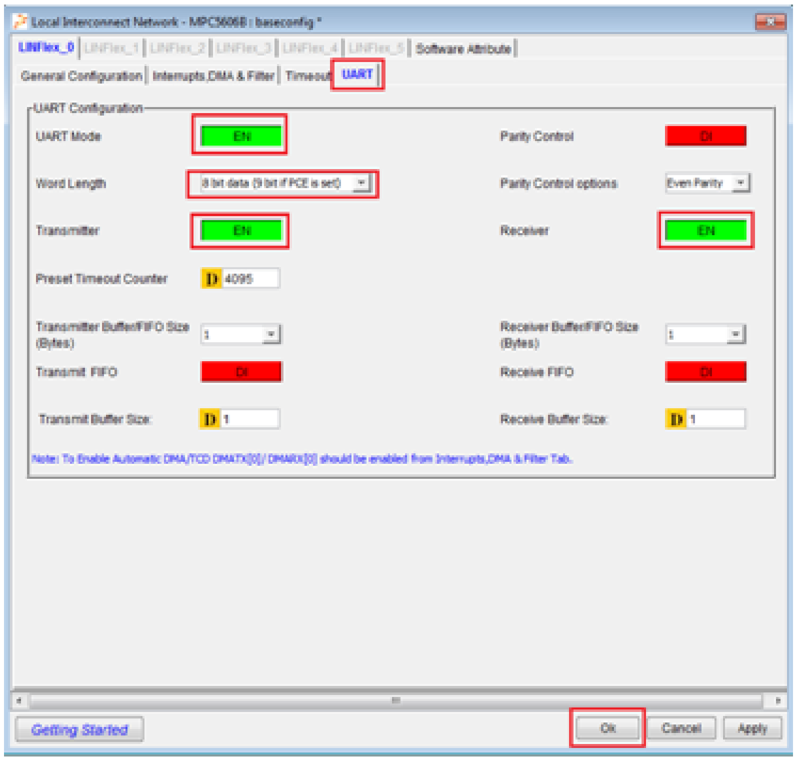
LIN Configuration

Master Mode	DI	LINFlex System Clock (MHz)	64 MHz
Receiver Buffer Locked Mode	DI	Loopback Mode	DI
LIN Master Break Length	10 bits	Self Test Mode	DI
Bypass Filter	EN	Slave Mode Break Detection Threshold	11 bits
Auto Wakeup	DI	LIN Auto Sync	DI
Checksum Field	EN	Idle on Bit Error	EN
Programmed Checksum		Checksum Calculation	DI
Integer Baud Rate Factor	34	Idle on Identifier Parity Error	EN
Baudrate(Symbols/Sec) =	115107.91	Fractional Baud Rate Factor	12/16

Note
Baudrate (Symbols/Sec) = Fcpu / (16 * LFDIV)
LFDIV = Integer factor(Mantissa) + Fractional factor(fraction)

Figure 21. LINFlex configuration

3. In the UART tab, *enable* UART, set Word Length to *8-bit* data and *enable* Transmitter and Receiver.
4. Select *OK* to finish LINFlex configuration.



Local Interconnect Network - MPC5606B : baseconfig *

LINFlex_0 | LINFlex_1 | LINFlex_2 | LINFlex_3 | LINFlex_4 | LINFlex_5 | Software Attribute

General Configuration | Interrupts,DMA & Filter | Timeout | UART

UART Configuration

UART Mode	EN	Parity Control	DI
Word Length	8 bit data (9 bit if PCE is set)	Parity Control options	Even Parity
Transmitter	EN	Receiver	EN
Preset Timeout Counter	4095		
Transmitter Buffer/FIFO Size (Bytes)	1	Receiver Buffer/FIFO Size (Bytes)	1
Transmit FIFO	DI	Receive FIFO	DI
Transmit Buffer Size	1	Receive Buffer Size	1

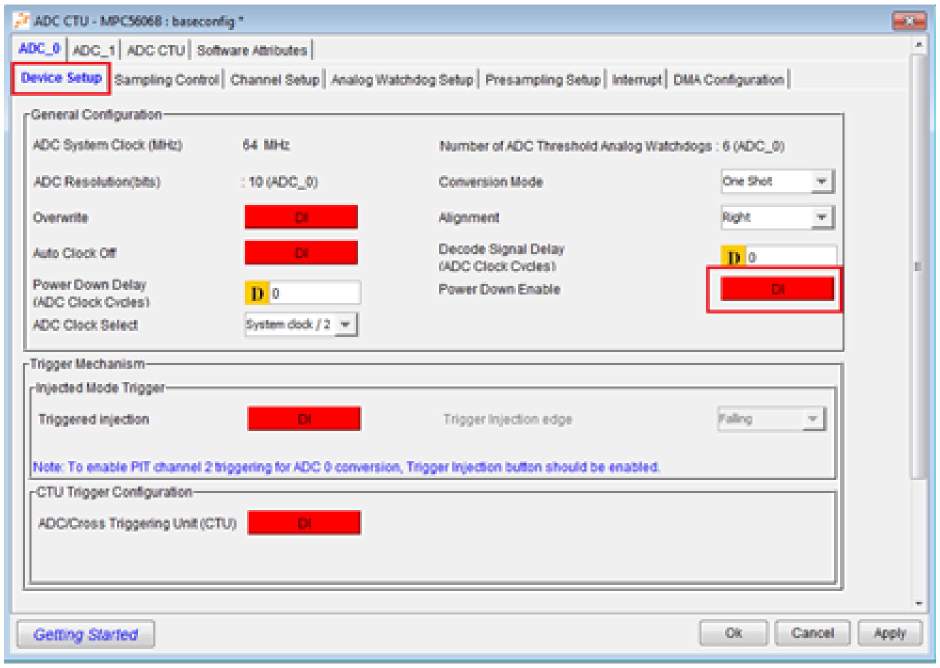
Note: To Enable Automatic DMA/TCD DMATX[0]/ DMARX[0] should be enabled from Interrupts,DMA & Filter Tab.

Figure 22. UART configuration

3.16 ADC configuration

To configure ADC peripheral:

1. Select ADC tile from the Peripheral Configuration window.
2. In the Device Setup tab, disable Power Down Enable option.



ADC CTU - MPC5606B : baseconfig *

ADC_0 | ADC_1 | ADC CTU | Software Attributes

Device Setup | Sampling Control | Channel Setup | Analog Watchdog Setup | Presampling Setup | Interrupt | DMA Configuration

General Configuration

ADC System Clock (MHz) 64 MHz

Number of ADC Threshold Analog Watchdogs : 6 (ADC_0)

ADC Resolution(bits) : 10 (ADC_0)

Conversion Mode: One Shot

Overwrite: DI

Alignment: Right

Auto Clock Off: DI

Decode Signal Delay (ADC Clock Cycles): 0

Power Down Delay (ADC Clock Cycles): 0

Power Down Enable: DI

ADC Clock Select: System clock / 2

Trigger Mechanism

Injected Mode Trigger

Triggered injection: DI

Trigger Injection edge: Falling

Note: To enable PIT channel 2 triggering for ADC 0 conversion, Trigger injection button should be enabled.

CTU Trigger Configuration

ADC/Cross Triggering Unit (CTU): DI

Figure 23. ADC configuration

3. In the Channel Setup tab, enable Channel 0 and Channel 1 in Normal Mode. These channels represent Potentiometer and Photo Sensor inputs.
4. Select OK to finish the ADC configuration.

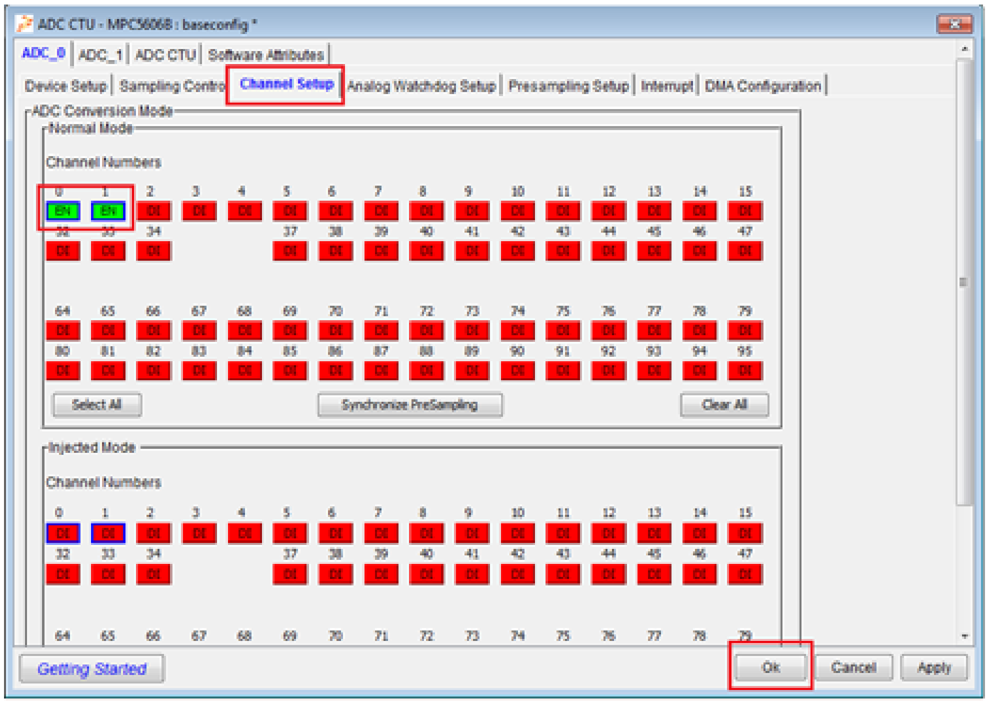


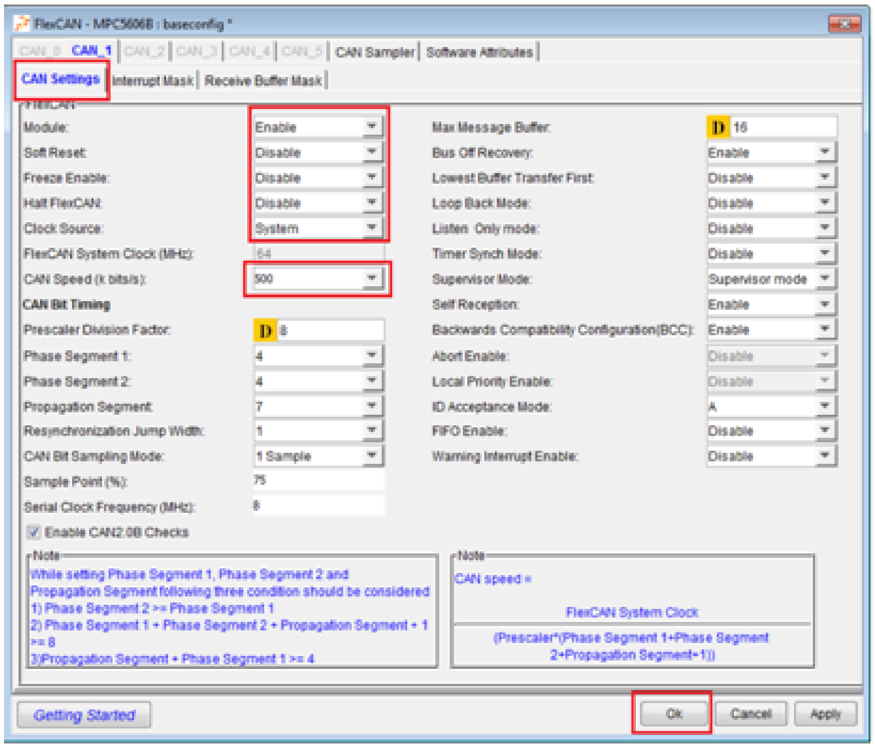
Figure 24. ADC channels configuration

3.17 FlexCAN configuration

To configure FlexCAN peripheral:

1. Select FlexCAN tile from the Peripheral Configuration window. In TRK-MPC5606B board, the CAN 1 peripheral is connected to CAN transceiver in SBC. In this example, use a CAN speed of 500 kbit/s.
2. To configure CAN1:
   - *Enable* CAN1 module
   - *Disable* Freeze and Halt modes
   - Set Clock Source to System
   - Set CAN speed to *500* kbit/s. RAppID init configures Phase segments and Propagation segment values automatically.

   `
3. Select *OK* to finish CAN 1 configuration.



Module:	Enable	Max Message Buffer:	16
Soft Reset:	Disable	Bus Off Recovery:	Enable
Freeze Enable:	Disable	Lowest Buffer Transfer First:	Disable
Halt FlexCAN:	Disable	Loop Back Mode:	Disable
Clock Source:	System	Listen Only mode:	Disable
FlexCAN System Clock (MHz):	64	Timer Synch Mode:	Disable
CAN Speed (k bits/s):	500	Supervisor Mode:	Supervisor mode
CAN Bit Timing		Self Reception:	Enable
Prescaler Division Factor:	8	Backwards Compatibility Configuration(BCC):	Enable
Phase Segment 1:	4	Abort Enable:	Disable
Phase Segment 2:	4	Local Priority Enable:	Disable
Propagation Segment:	7	ID Acceptance Mode:	A
Resynchronization Jump Width:	1	FIFO Enable:	Disable
CAN Bit Sampling Mode:	1 Sample	Warning Interrupt Enable:	Disable
Sample Point (%):	75		
Serial Clock Frequency (MHz):	8		

[✓] Enable CAN2.0B Checks

Note
While setting Phase Segment 1, Phase Segment 2 and Propagation Segment following three condition should be considered
1) Phase Segment 2 >= Phase Segment 1
2) Phase Segment 1 + Phase Segment 2 + Propagation Segment + 1 >= 8
3)Propagation Segment + Phase Segment 1 >= 4

Note
CAN speed = FlexCAN System Clock / (Prescaler*(Phase Segment 1+Phase Segment 2+Propagation Segment+1))

Figure 25. CAN configuration

3.18 eMIOS configuration

To configure eMIOS peripheral:

1. Select eMIOS tile from the Peripheral Configuration window. In this example, LED4 output is driven by PWM signal using eMIOS peripheral. Initially, PWM period is set to 1 ms and duty cycle of 50%. When the code is running on the target, the duty cycle can be changed using input switches S3 and S4.
2. In the Module Configuration tab, check *Global Time base Enable and Enable Global Prescaler* options.

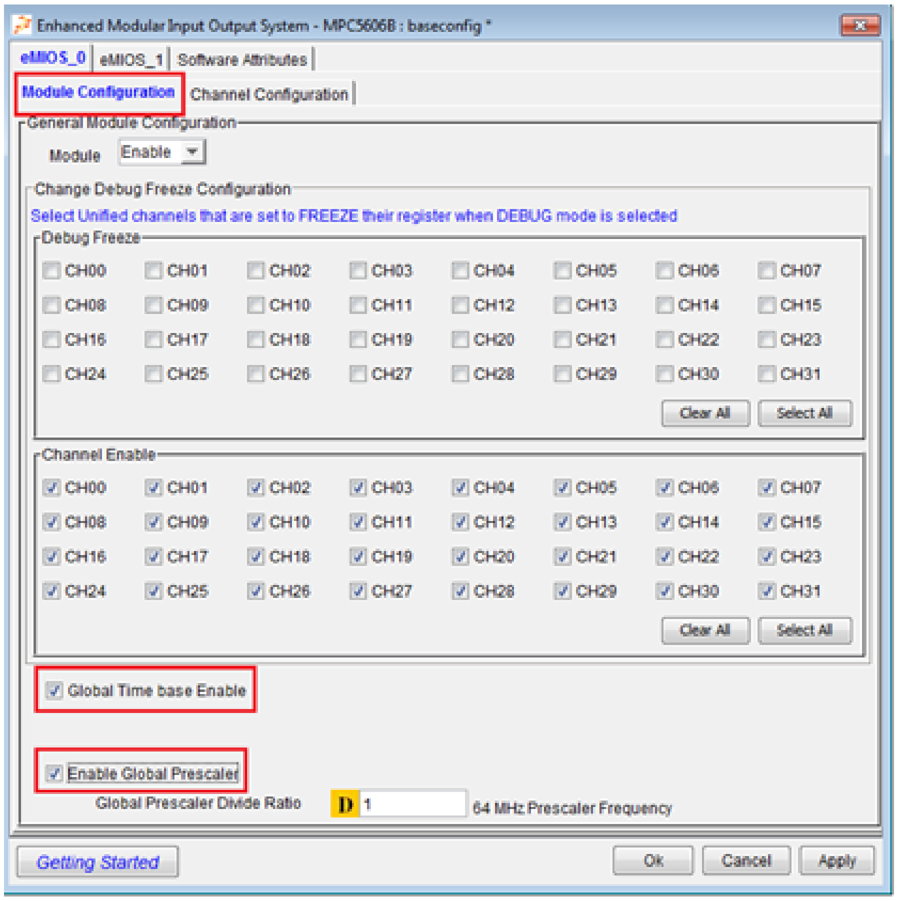
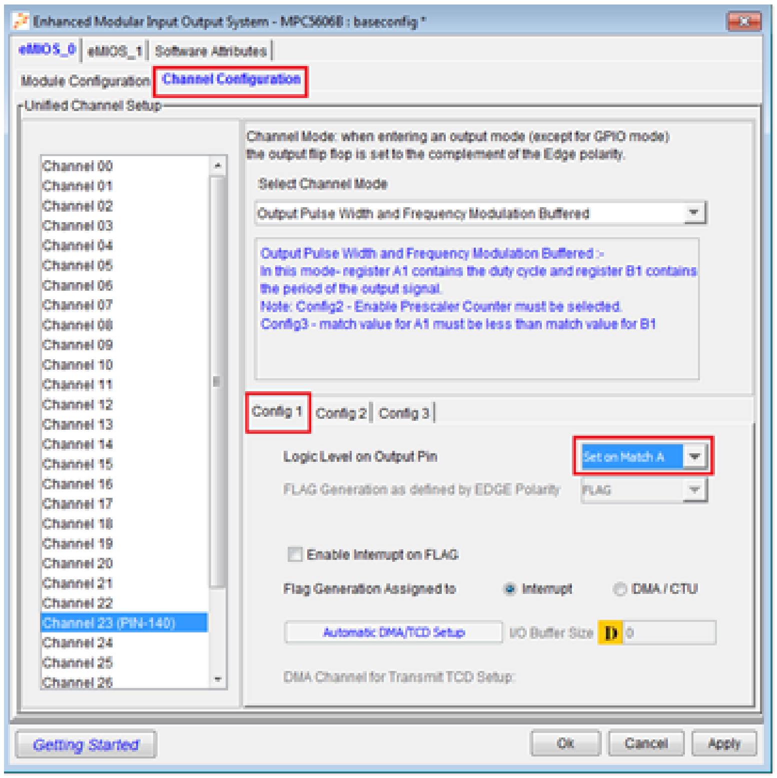

Enhanced Modular Input Output System - MPC5606B : baseconfig *
eMIOS_0
eMIOS_1
Software Attributes
Module Configuration
Channel Configuration
General Module Configuration
Module Enable
Change Debug Freeze Configuration
Select Unified channels that are set to FREEZE their register when DEBUG mode is selected
Debug Freeze
CH00 CH01 CH02 CH03 CH04 CH05 CH06 CH07
CH08 CH09 CH10 CH11 CH12 CH13 CH14 CH15
CH16 CH17 CH18 CH19 CH20 CH21 CH22 CH23
CH24 CH25 CH26 CH27 CH28 CH29 CH30 CH31
Clear All
Select All
Channel Enable
✓ CH00 ✓ CH01 ✓ CH02 ✓ CH03 ✓ CH04 ✓ CH05 ✓ CH06 ✓ CH07
✓ CH08 ✓ CH09 ✓ CH10 ✓ CH11 ✓ CH12 ✓ CH13 ✓ CH14 ✓ CH15
✓ CH16 ✓ CH17 ✓ CH18 ✓ CH19 ✓ CH20 ✓ CH21 ✓ CH22 ✓ CH23
✓ CH24 ✓ CH25 ✓ CH26 ✓ CH27 ✓ CH28 ✓ CH29 ✓ CH30 ✓ CH31
Clear All
Select All
✓ Global Time base Enable
✓ Enable Global Prescaler
Global Prescaler Divide Ratio 1
64 MHz Prescaler Frequency
Getting Started
Ok
Cancel
Apply

Figure 26. eMIOS configuration

3. In the Channel Configuration tab, select *Config1*, *Config2*, and *Config3* options as shown in figures below.


Enhanced Modular Input Output System - MPC5606B : baseconfig *

eMIOS_0 | eMIOS_1 | Software Attributes

Module Configuration | Channel Configuration

Unified Channel Setup

Channel 00
Channel 01
Channel 02
Channel 03
Channel 04
Channel 05
Channel 06
Channel 07
Channel 08
Channel 09
Channel 10
Channel 11
Channel 12
Channel 13
Channel 14
Channel 15
Channel 16
Channel 17
Channel 18
Channel 19
Channel 20
Channel 21
Channel 22
Channel 23 (PIN-140)
Channel 24
Channel 25
Channel 26

Channel Mode: when entering an output mode (except for GPIO mode) the output flip flop is set to the complement of the Edge polarity.

Select Channel Mode: Output Pulse Width and Frequency Modulation Buffered

Output Pulse Width and Frequency Modulation Buffered :-
In this mode- register A1 contains the duty cycle and register B1 contains the period of the output signal.
Note: Config2 - Enable Prescaler Counter must be selected.
Config3 - match value for A1 must be less than match value for B1

Config 1 | Config 2 | Config 3

Logic Level on Output Pin: Set on Match A

FLAG Generation as defined by EDGE Polarity: FLAG

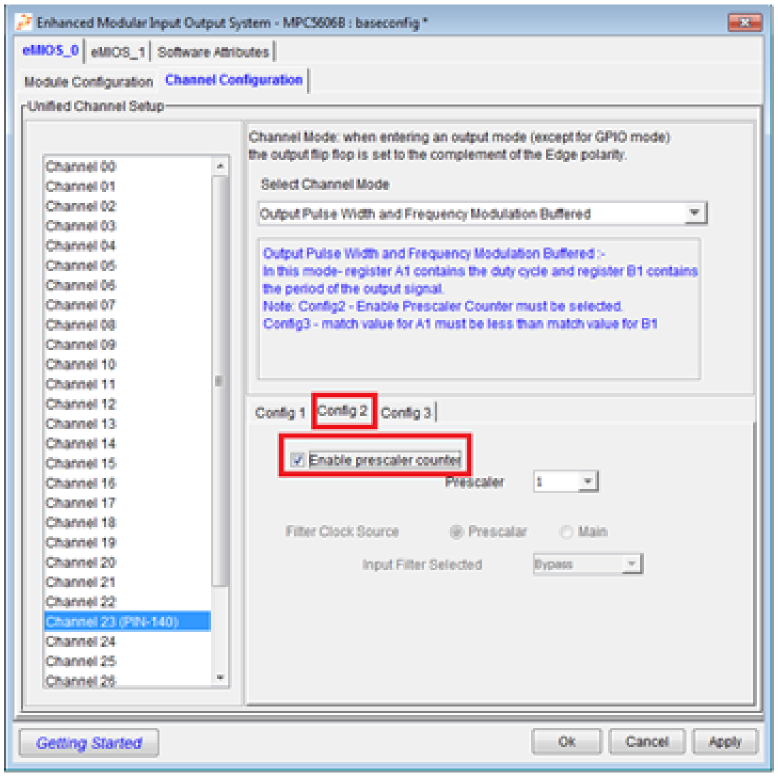
Enable Interrupt on FLAG

Flag Generation Assigned to: Interrupt / DMA / CTU

Automatic DMA/TCD Setup | I/O Buffer Size 0

DMA Channel for Transmit TCD Setup:

Figure 27. eMIOS Channel configuration I


Enhanced Modular Input Output System - MPC5606B : baseconfig *

eMIOS_0 | eMIOS_1 | Software Attributes

Module Configuration | Channel Configuration

Unified Channel Setup

Channel 00
Channel 01
Channel 02
Channel 03
Channel 04
Channel 05
Channel 06
Channel 07
Channel 08
Channel 09
Channel 10
Channel 11
Channel 12
Channel 13
Channel 14
Channel 15
Channel 16
Channel 17
Channel 18
Channel 19
Channel 20
Channel 21
Channel 22
Channel 23 (PIN-140)
Channel 24
Channel 25
Channel 26

Channel Mode: when entering an output mode (except for GPIO mode) the output flip flop is set to the complement of the Edge polarity.

Select Channel Mode: Output Pulse Width and Frequency Modulation Buffered

Output Pulse Width and Frequency Modulation Buffered :-
In this mode- register A1 contains the duty cycle and register B1 contains the period of the output signal.
Note: Config2 - Enable Prescaler Counter must be selected.
Config3 - match value for A1 must be less than match value for B1

Config 1 | Config 2 | Config 3

[✓] Enable prescaler counter

Prescaler: 1

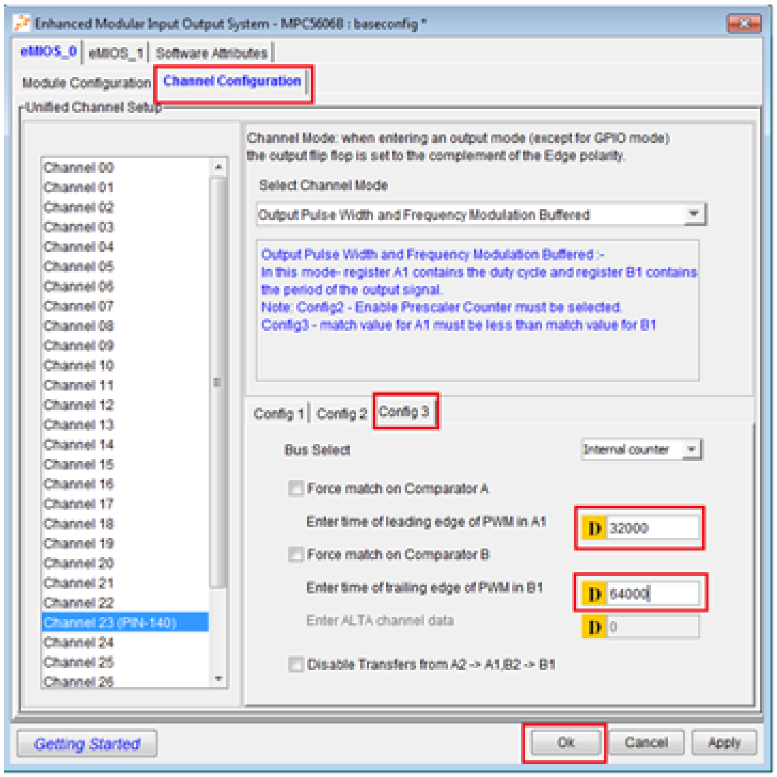
Filter Clock Source: Prescalar / Main

Input Filter Selected: Bypass

Figure 28. eMIOS Channel configuration II

In Config 3 options, the value in B1 represents PWM period. With clock set at 64 MHz and prescaler value at 1, a value of 64000 would result in 1 ms period. (B1*(1/ (eMIOS0 Peripheral Clock/Prescaler Divide Ratio))

A1 value of 32000 would result in 50% duty cycle (100*(B1-A1)/B1).



Channel Mode: when entering an output mode (except for GPIO mode) the output flip flop is set to the complement of the Edge polarity.

Select Channel Mode: Output Pulse Width and Frequency Modulation Buffered

Output Pulse Width and Frequency Modulation Buffered :-
In this mode- register A1 contains the duty cycle and register B1 contains the period of the output signal.
Note: Config2 - Enable Prescaler Counter must be selected.
Config3 - match value for A1 must be less than match value for B1

Bus Select: Internal counter

Force match on Comparator A

Enter time of leading edge of PWM in A1: 32000

Force match on Comparator B

Enter time of trailing edge of PWM in B1: 64000

Enter ALTA channel data: 0

Disable Transfers from A2 -> A1,B2 -> B1

Figure 29. eMIOS Channel configuration III

This completes the configuration of peripherals.

4. Select *Next* to configure PIT interrupt.

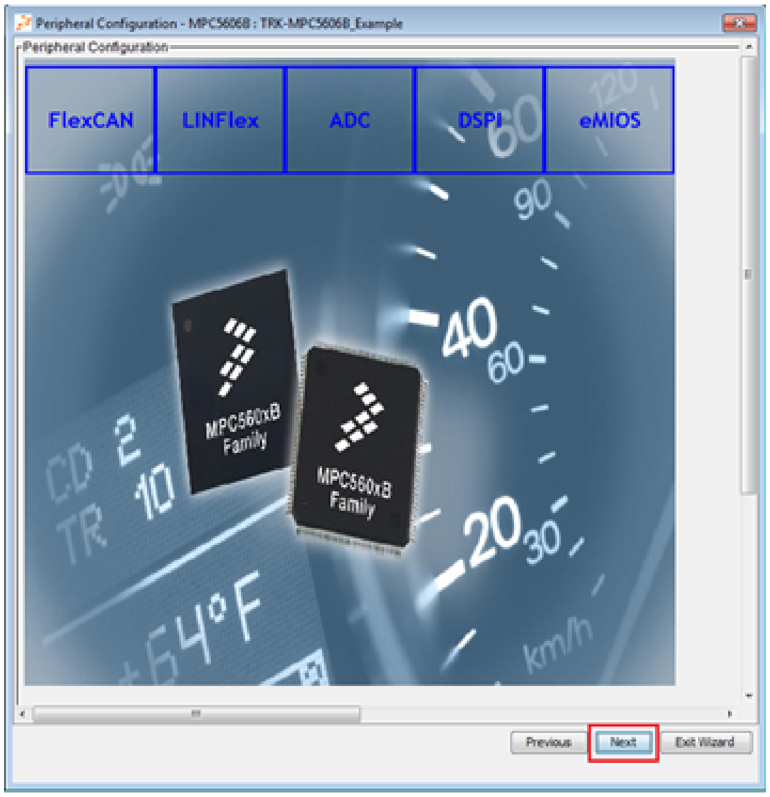
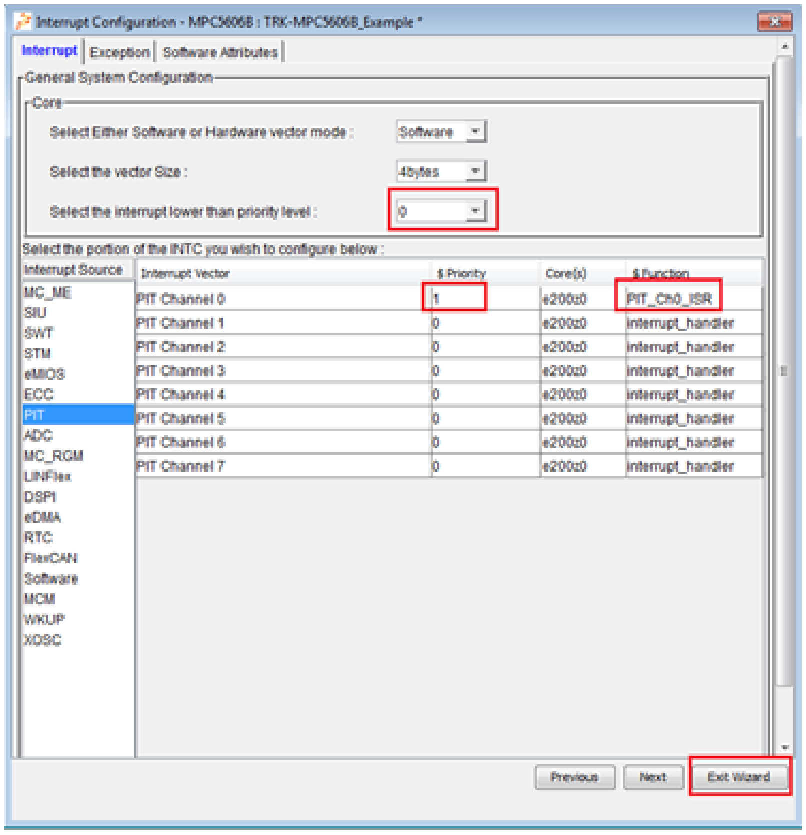


Figure 30. Completing peripheral configuration

3.19 Interrupt configuration

In this example, PIT Channel 0 interrupt is used to process input switches S3 and S4 to increment or decrement duty cycle of LED4 PWM. Configure PIT Ch0 interrupt as shown. RAppID is configured to generate a skeleton ISR function named PIT_Ch0_ISR. The contents to process input switches and changing PWM values will be explained later. This completes the microcontroller configuration. Select *Exit Wizard* to exit to main RAppID window.



General System Configuration

Core

Select Either Software or Hardware vector mode : Software

Select the vector Size : 4bytes

Select the interrupt lower than priority level : 0

Select the portion of the INTC you wish to configure below :

Interrupt Source: MC_ME, SIU, SWT, STM, eMIOS, ECC, PIT, ADC, MC_RGM, LINFlex, DSPI, eDMA, RTC, FlexCAN, Software, MCM, WKUP, XOSC

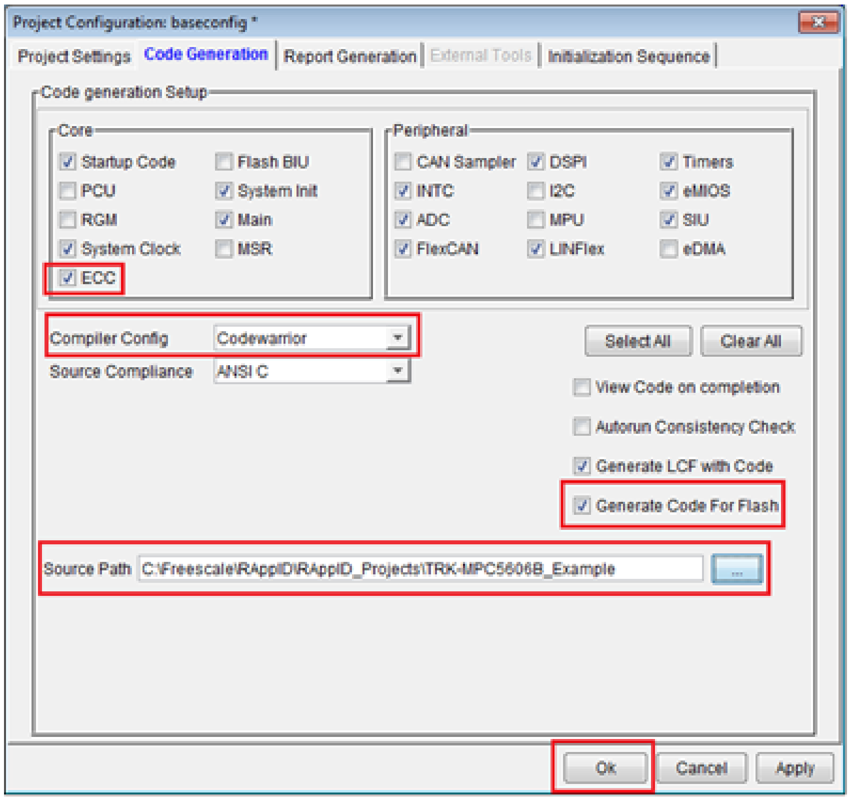
Interrupt Vector	$Priority	Core(s)	$Function
PIT Channel 0	1	e200z0	PIT_Ch0_ISR
PIT Channel 1	0	e200z0	interrupt_handler
PIT Channel 2	0	e200z0	interrupt_handler
PIT Channel 3	0	e200z0	interrupt_handler
PIT Channel 4	0	e200z0	interrupt_handler
PIT Channel 5	0	e200z0	interrupt_handler
PIT Channel 6	0	e200z0	interrupt_handler
PIT Channel 7	0	e200z0	interrupt_handler

Figure 31. PIT Interrupt configuration

3.20 Code configuration

Once you have completed all pin and peripheral configurations required for this example project, you are ready to generate the code.

1. From main RAppID window, select menu *Configuration* > *Code Generation*. This will pop up Code Generation window shown below. By default, RAppID generates code for all peripherals. In this example, you need to select only the peripherals that are used in the example for code generation.
2. Deselect the peripherals that are not required (Flash BIU, PCU, RGM, MSR, CAN Sampler, I2C, MPU, and eDMA).
3. Select code generation for ECC to initialize SRAM and ECC registers.
4. Select CodeWarrior compiler option.
5. By default, RAppID generates code for RAM. In this example you generate code for Flash. Select *Generate Code For Flash* option.
6. Enter the path where code should be generated and select OK to complete Code Configuration options.


Project Configuration: baseconfig *

Project Settings | Code Generation | Report Generation | External Tools | Initialization Sequence

Code generation Setup

Core

- [✓] Startup Code
- [ ] PCU
- [ ] RGM
- [✓] System Clock
- [✓] ECC
- [ ] Flash BIU
- [✓] System Init
- [✓] Main
- [ ] MSR

Peripheral

- [ ] CAN Sampler
- [✓] INTC
- [✓] ADC
- [✓] FlexCAN
- [✓] DSPI
- [ ] I2C
- [ ] MPU
- [✓] LINFlex
- [✓] Timers
- [✓] eMIOS
- [✓] SIU
- [ ] eDMA

Compiler Config: Codewarrior

Source Compliance: ANSI C

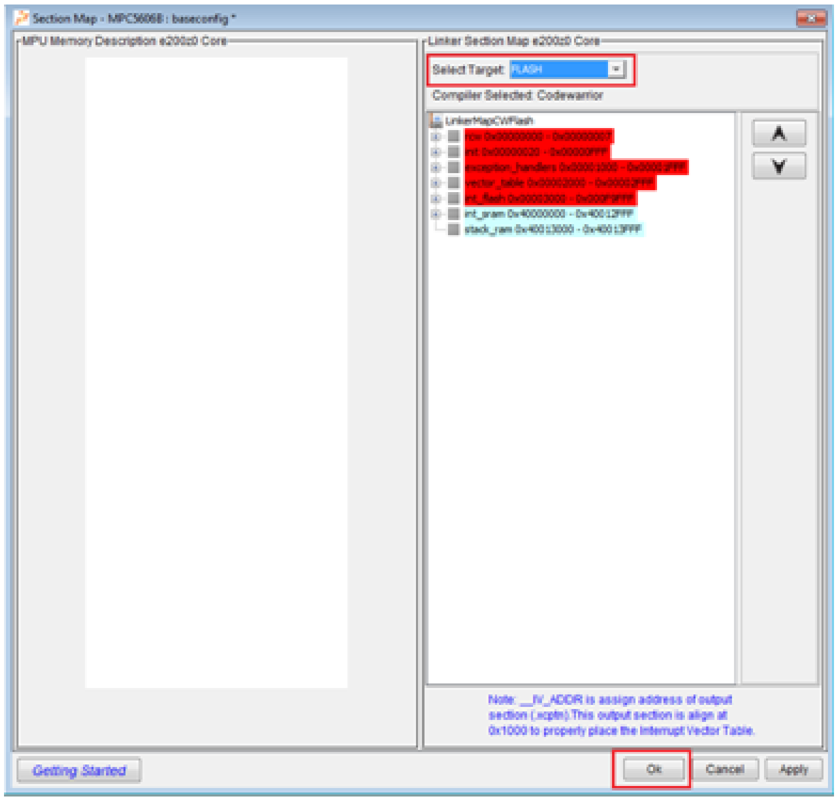
- [ ] View Code on completion
- [ ] Autorun Consistency Check
- [✓] Generate LCF with Code
- [✓] Generate Code For Flash

Source Path: C:\Freescale\RAppID\RAppID_Projects\TRK-MPC5606B_Example

Figure 32. Code configuration

Since you are generating code for Flash, you need to generate linker file for Flash as well.

7. From main RAppID window, select menu *View> View Section Map*.
8. Change Target to *Flash*.
9. Select *Ok*.


Section Map - MPC5606B : baseconfig *
MPU Memory Description e200z0 Core
Linker Section Map e200z0 Core
Getting Started
Ok
Cancel
Apply

Select Target: FLASH

Compiler Selected: Codewarrior

```
LinkerMapCWFlash
  rcw 0x00000000 - 0x00000007
  init 0x00000020 - 0x00000FFF
  exception_handlers 0x00001000 - 0x00001FFF
  vector_table 0x00002000 - 0x00002FFF
  int_flash 0x00003000 - 0x000F9FFF
  int_sram 0x40000000 - 0x40012FFF
  stack_ram 0x40013000 - 0x40013FFF
```

Note: __IV_ADDR is assign address of output section (.xcptn).This output section is align at 0x1000 to properly place the Interrupt Vector Table.

Figure 33. Section map

3.21 Generate code

Select Generate Code icon to generate code. Save the project when prompted. You are now ready to build the project using CodeWarrior.

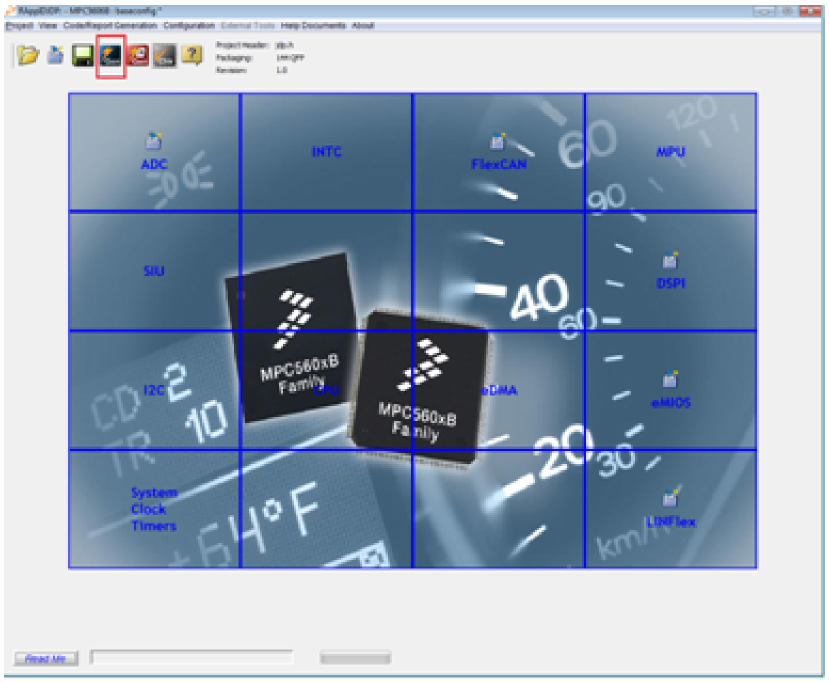


Figure 34. Generate code

In addition to RAppID generated code, you will use the driver code supplied with Fast Start Kit and FreeMASTER code in this example project. During the installation, driver code should be installed at the folder *C:\Freescale\FastStartKit\TRK-MPC5606B\drivercode*. Copy all the driver code to the folder where RAppID code is generated.

The FreeMASTER code should be installed in the folder *C:\Freescale\FreeMASTER Serial Communication V1.6*. Copy all the code from sub-folders *src_common* and *src_platforms\MPC56xx* to the location where RAppID code is generated.

FreeMASTER can be used in polling or interrupt mode via CAN or SCI. In this example, you will use FreeMASTER in poll mode via SCI.

To use FreeMASTER in polling or interrupt mode you will have to make changes to FreeMASTER configuration header file.

1. Rename *freemaster_cfg.h.example* file to *freemaster_cfg.h*. This file contains all the macro definitions available for the FreeMASTER configuration.
2. Select poll driven SCI communication and disable TSA by making following changes to *freemaster_cfg.h*:

```
#define FMSTR_SHORT_INTR        0
#define FMSTR_POLL_DRIVEN       1
#define FMSTR_USE_TSA           0
```

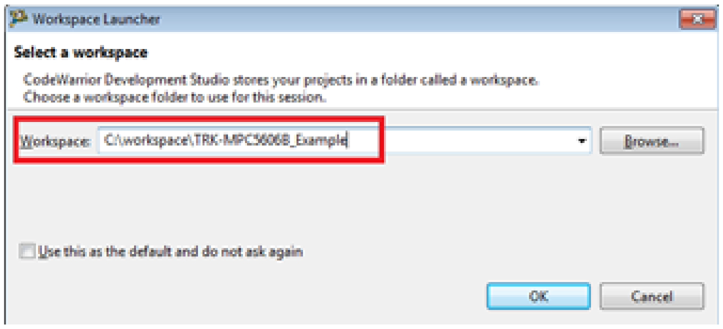
4 Build code using CodeWarrior

To compile and build RAppID generated source code, create an empty CodeWarrior project and use *cwpjmaker* utility to add the RAppID generated source code to the project.

4.1 Create CodeWarrior project

To create a CodeWarrior project:

1. Launch CodeWarrior 10.5 from Windows Start menu and provide a workspace name.
2. Select *OK*.


Workspace Launcher

Select a workspace

CodeWarrior Development Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\workspace\TRK-MPC5606B_Example

Figure 35. CodeWarrior Workspace Launcher

To create an empty bare board project:

1. Select *New MCU project* from Commander window.

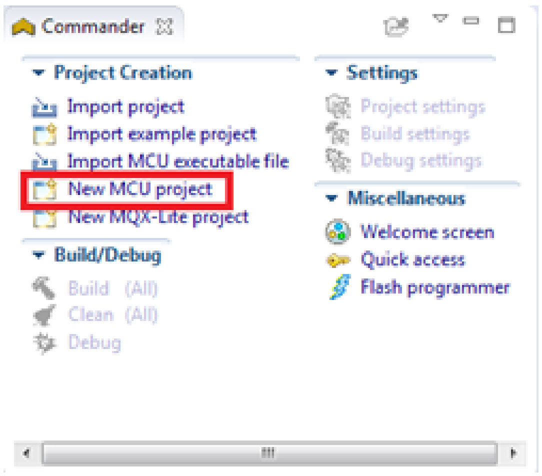
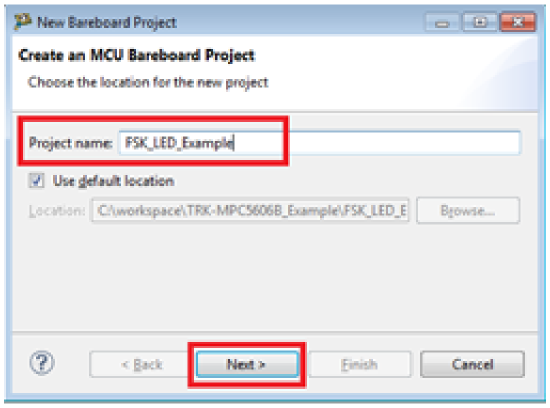


Figure 36. Start New CodeWarrior project

2. Provide a name for the project and select *Next*.


New Bareboard Project

Create an MCU Bareboard Project

Choose the location for the new project

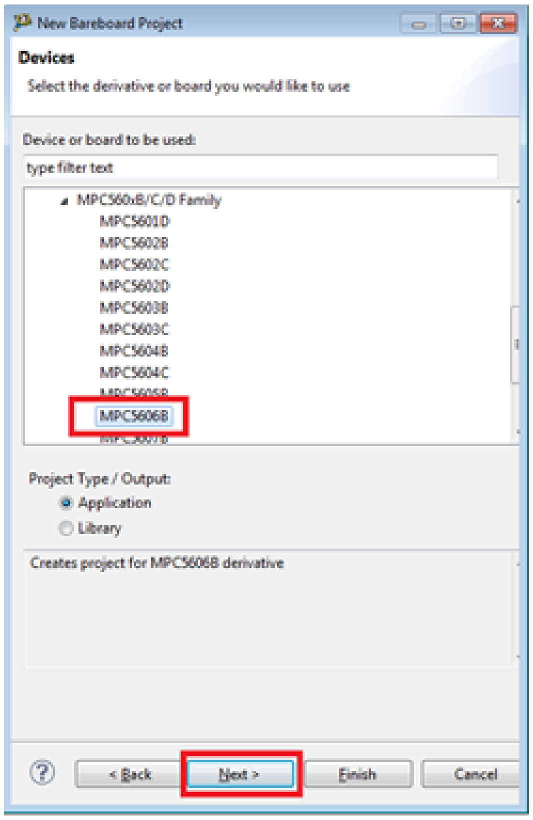
Project name: FSK_LED_Example

Use default location

Location: C:\workspace\TRK-MPC5606B_Example\FSK_LED_E

Figure 37. Provide a project name

3. Select *5606B device* and select *Next*.


New Bareboard Project

Devices

Select the derivative or board you would like to use

Device or board to be used:

type filter text

- MPC560xB/C/D Family
  - MPC5601D
  - MPC5602B
  - MPC5602C
  - MPC5602D
  - MPC5603B
  - MPC5603C
  - MPC5604B
  - MPC5604C
  - MPC5605B
  - MPC5606B
  - MPC5607B

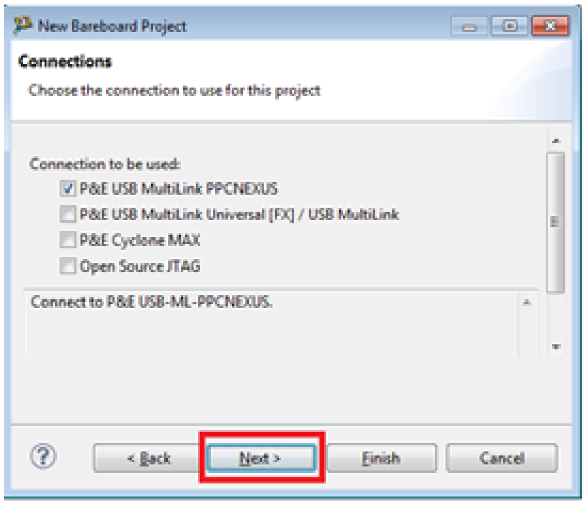
Project Type / Output:

- (●) Application
- ( ) Library

Creates project for MPC5606B derivative

Figure 38. Select devices

4. Click *Next* and *Finish* to accept default options for Connections and Language options.


New Bareboard Project

Connections

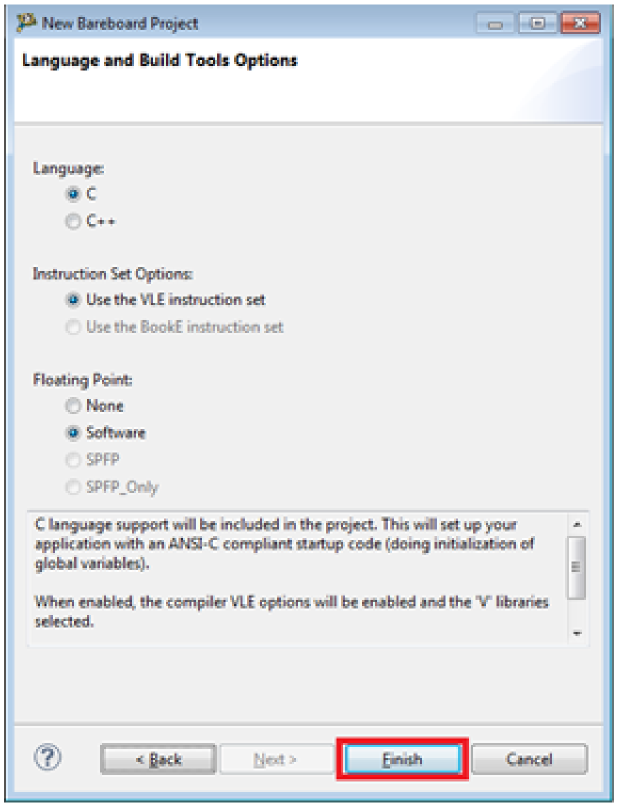
Choose the connection to use for this project

Connection to be used:

- [✓] P&E USB MultiLink PPCNEXUS
- [ ] P&E USB MultiLink Universal [FX] / USB MultiLink
- [ ] P&E Cyclone MAX
- [ ] Open Source JTAG

Connect to P&E USB-ML-PPCNEXUS.

Figure 39. Select connections


New Bareboard Project

Language and Build Tools Options

Language:
- [✓ C]
- C++

Instruction Set Options:
- [✓ Use the VLE instruction set]
- Use the BookE instruction set

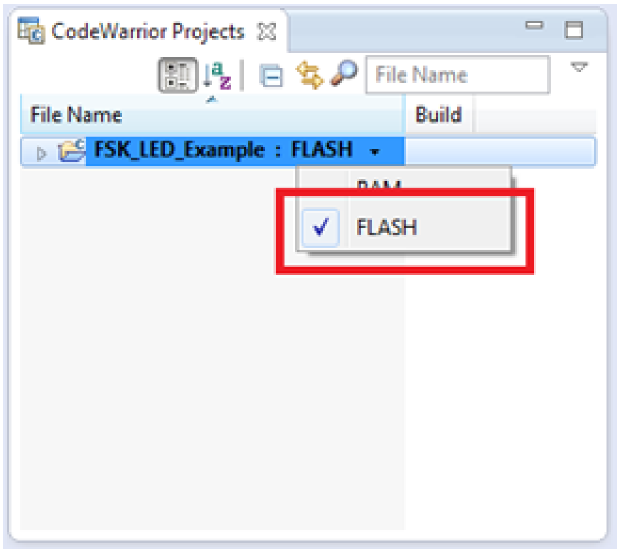
Floating Point:
- None
- [✓ Software]
- SPFP
- SPFP_Only

C language support will be included in the project. This will set up your application with an ANSI-C compliant startup code (doing initialization of global variables).

When enabled, the compiler VLE options will be enabled and the 'V' libraries selected.

Figure 40. Select language and build tools options

By default, CodeWarrior sets the option to create build for RAM. Since your example project is for Flash, change the CodeWarrior build option to FLASH as shown below.


CodeWarrior Projects

File Name	Build
FSK_LED_Example : FLASH	

Figure 41. Select FLASH build option

4.2 Add RAppID code to CodeWarrior project

CodeWarrior creates a new bare board project with CodeWarrior generated code and linker file included in the project. Since the code in the example project is generated using RAppID, you need to remove all the CodeWarrior generated code. Next, you need to add the RAppID generated code, the linker file along with the FreeMASTER, and the driver code to the CodeWarrior project.

You can accomplish this using the CodeWarrior project maker utility – *cwpjmaker*.

1. Select *Project* > *Close Project* to close the FSK_LED_Example project.
2. Run the *cwpjmaker* utility. This can be done by selecting the executable from the directory where this utility is installed (*C:\Freescale\cwpjmaker\cwpjmaker_01.exe*) or by using RAppID menu *External Tools->CWInterface*.

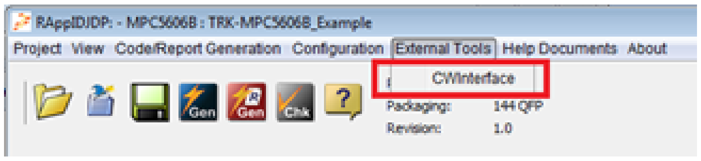

RAppIDJDP: - MPC5606B : TRK-MPC5606B_Example
Project View Code/Report Generation Configuration External Tools Help Documents About
CWInterface
Packaging: 144 QFP
Revision: 1.0

Figure 42. Select language and build tools options

The Code Warrior Project Maker utility expects three inputs:

- The path where RAppID generated source code resides. All the code that exists in this folder will be copied to the CodeWarrior project.
- The path where CodeWarrior project resides
- The path of RAppID generated linker file to be included in the CodeWarrior project.

3. Enter the required values as shown below and select *OK*.

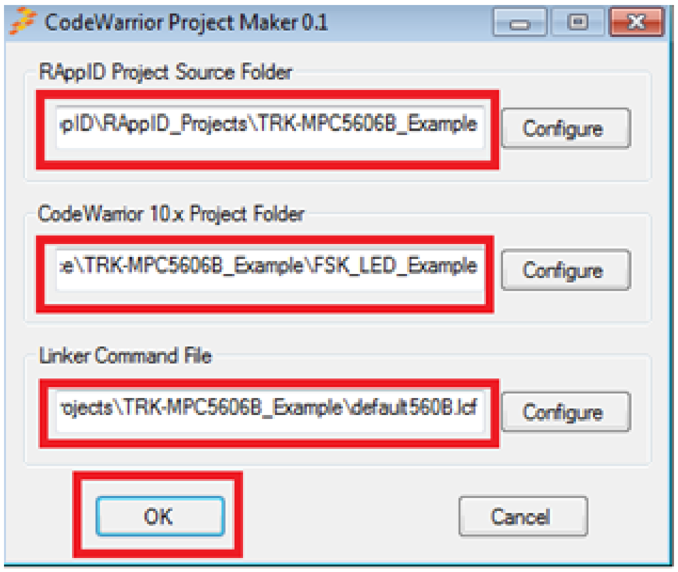
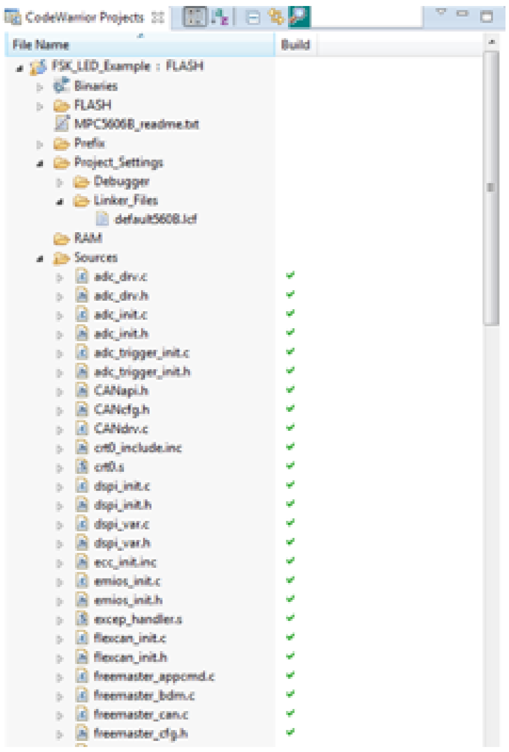

CodeWarrior Project Maker 0.1
RAppID Project Source Folder
pID\RAppID_Projects\TRK-MPC5606B_Example
Configure
CodeWarrior 10.x Project Folder
e\TRK-MPC5606B_Example\FSK_LED_Example
Configure
Linker Command File
ojects\TRK-MPC5606B_Example\default560B.lcf
Configure
OK
Cancel

Figure 43. CodeWarrior Project Maker

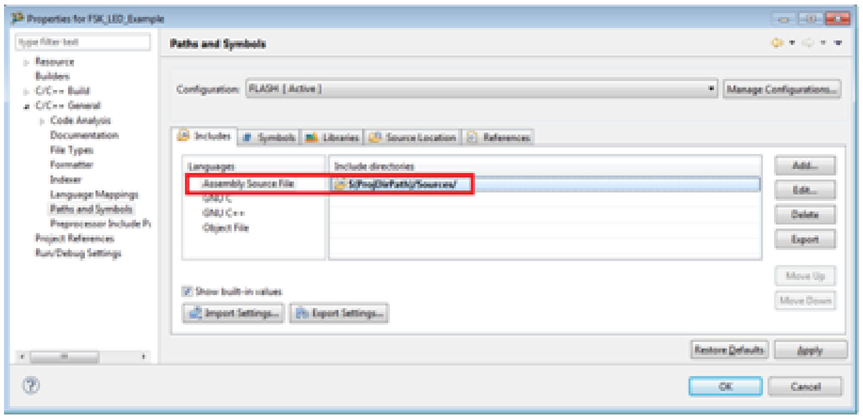
4. Once the tool is executed, all the RAppID generated source files, driver code and FreeMASTER code is added to the CodeWarrior project.
5. Open the CodeWarrior project that was previously closed and verify that the project now includes all the required source files as shown below.



```
CodeWarrior Projects
File Name                               Build
FSK_LED_Example : FLASH
  Binaries
  FLASH
  MPC5606B_readme.txt
  Prefix
  Project_Settings
    Debugger
    Linker_Files
      default5608.lcf
  RAM
  Sources
    adc_drv.c                           ✓
    adc_drv.h                           ✓
    adc_init.c                          ✓
    adc_init.h                          ✓
    adc_trigger_init.c                  ✓
    adc_trigger_init.h                  ✓
    CANapi.h                            ✓
    CANcfg.h                            ✓
    CANdrv.c                            ✓
    crt0_include.inc                    ✓
    crt0.s                              ✓
    dspi_init.c                         ✓
    dspi_init.h                         ✓
    dspi_var.c                          ✓
    dspi_var.h                          ✓
    ecc_init.inc                        ✓
    emios_init.c                        ✓
    emios_init.h                        ✓
    excep_handler.s                     ✓
    flexcan_init.c                      ✓
    flexcan_init.h                      ✓
    freemaster_appcmd.c                 ✓
    freemaster_bdm.c                    ✓
    freemaster_can.c                    ✓
    freemaster_cfg.h                    ✓
```

Figure 44. CodeWarrior Projects view

6. Add path of RAppID generated assembly source files to CodeWarrior path.
   - Select *Build Settings* menu from CodeWarrior Commander window. This will pop up Properties window shown below.
   - Select *C/C++General->Paths and Symbols* option and add entry “${ProjDirPath}/Sources/” to Assembly Source File as shown below.



Properties for FSK_LED_Example

Paths and Symbols

Configuration: FLASH [Active]

Languages	Include directories
Assembly Source File	${ProjDirPath}/Sources/
GNU C	
GNU C++	
Object File	

Figure 45. Adding Path for assembly source

4.3 Add application code to auto generated code

4.3.1 Adding code to main.c

The main.c generated by RAppID initializes peripherals and includes an empty while loop. To add code to main.c:

1. Initialize SBC to enable CAN
2. Process ADC – Turn on/off LED1 and LED2 based on Potentiometer and Photo sensor ADC inputs
3. Process CAN messages: Turn on/off LED3 based on CAN message received and transmit appropriate response
4. Add code to support FreeMASTER utility

The complete c-code listing for main.c is shown below.

Example 1. Listing of main.c

```
/********************  Dependent Include files here *********************/

#include "rappid_ref.h"
#include "rappid_utils.h"
#include "sys_init.h"
#include "CANapi.h"
#include "sbc_hld.h"
#include "gpio_drv.h"
#include "pot_hld.h"
#include "photo_sensor_hld.h"
#include "freemaster.h"

/**********************  Function Prototype here ************************/

void main(void);
void ProcessCAN(void);
void ProcessADC(void);

/**********************  Global variables here ************************/
/* CAN messages to transmit */
unsigned char msgOKCAN[8]  = {1,1,0,0,0,0,0,0};
unsigned char msgErrorCAN[8]  = {1,0xFF,0,0,0,0,0,0};

uint16_t potValue;/* Potentiometer ADC input value*/
uint16_t photoSensorValue;/* Photo sensor ADC input value */

/*********************  Initialization Function(s) ***********************/

void main(void)
{

/* ----------------------------------------------------------- */
/*             System Initialization Function                  */
/* ----------------------------------------------------------- */
   sys_init_fnc();

   /* Initialize SBC */
   SBC_Init_DBG();

   /* FreeMASTER internal variables initialization */
FMSTR_Init();

/********* Enable External Interrupt *********/
   EnableExternalInterrupts();

   /* Turn off LEDs */
   GPIO_SetState(68, 1);
   GPIO_SetState(69, 1);
   GPIO_SetState(70, 1);
   GPIO_SetState(71, 1);

   /* Initialize CAN filter */
   SetCanRxFilter(1, 0, 0);

   while(1)
   {
FMSTR_Poll();
```

```
  ProcessCAN();
  ProcessADC();
   }

}

/*******************************************************************************
*   Function: ProcessCAN
*
*   Description: Process CAN messages
*
*******************************************************************************/
void ProcessCAN(void)
{
can_msg_struct msgCanRX;

if (CanRxMbFull(0) == 1)/* Check if CAN message received */
{
    msgCanRX = CanRxMsg(0);
    if (msgCanRX.data[0] == 0)/* If first data byte is 0, turn off LED3 and send positive
response */
    {
    GPIO_SetState(70, 1);
    CanTxMsg (2, 1, 8, (uint8_t *)msgOKCAN, 0);
    }
    else if (msgCanRX.data[0] == 1)/* If first data byte is 1, turn on LED3 and send
positive response */
    {
    GPIO_SetState(70, 0);
    CanTxMsg (2, 1, 8, (uint8_t *)msgOKCAN, 0);
    }
    else/* If first data byte is not 0 or 1, send a negative response */
    {
    CanTxMsg (2, 1, 8, (uint8_t *)msgErrorCAN, 0);
    }
}
}


/*******************************************************************************
*   Function: ProcessADC
*
*   Description: Processes Potentiometer and Photo sensor ADC inputs
*
*******************************************************************************/

void ProcessADC(void)
{
potValue = Pot_Get_Value();
if(potValue <= 500) /* If Potentiometer input is <= 500 turn on LED1, otherwise turn off
LED1 */
{
GPIO_SetState(68, 0);
}
else
{
GPIO_SetState(68, 1);
}

photoSensorValue = Photo_Sensor_Get_Value();
if(photoSensorValue <= 500) /* If Photo sensor input is <= 500 turn on LED2, otherwise turn
off LED2 */
{
GPIO_SetState(69, 0);
}
else
{
GPIO_SetState(69, 1);
}
}
```

4.3.2 Adding code to PIT ISR function

The skeleton PIT ISR code generated by RAppID clears the timer interrupt flag. You need to add code to process the S3 and S4 switch inputs and based on the states of these switches, increment or decrement the PWM duty cycle values for LED4 output.

The complete c-code listing for intc_pit.c is shown below.

```
Example 2. Listing of intc_pit.c
/********************  Dependent Include files here **********************/

#include "intc_pit.h"
#include "gpio_drv.h"

#define DC_STEP6400
#define MIN_DC6400
#define MAX_DC64000

/************************* INTERRUPT HANDLERS ************************/
void PIT_Ch0_ISR (void)
{
    uint32_t tmp;

    PIT.CH[0].TFLG.R = 0x00000001;
/* If switch S3 is pressed and S4 is not pressed */
    if (GPIO_GetState(66) && !GPIO_GetState(67))
    {
        /* Increase PWM  Duty Cycle */
        tmp = EMIOS_0.CH[23].CADR.R;
        if(tmp < MAX_DC)
        {
        tmp = tmp + DC_STEP;
        }
        EMIOS_0.CH[23].CADR.R = tmp;
    }
    /* If switch S4 is pressed and S3 is not pressed */
    else if(GPIO_GetState(67) && !GPIO_GetState(66))
    {
        /* Decrease PWM Duty Cycle */
        tmp = EMIOS_0.CH[23].CADR.R;
        if(tmp > MIN_DC)
        {
        tmp = tmp - DC_STEP;
        }
        EMIOS_0.CH[23].CADR.R = tmp;
    }
}
```

4.4 Build code

After completing the code modifications described in the previous section, you are ready to build the example project.

To build the project, select *Build* menu in the Commander window. This will compile and build the project resulting in *.elf* file that contains debug symbols and *.mot* file which is s-record file.

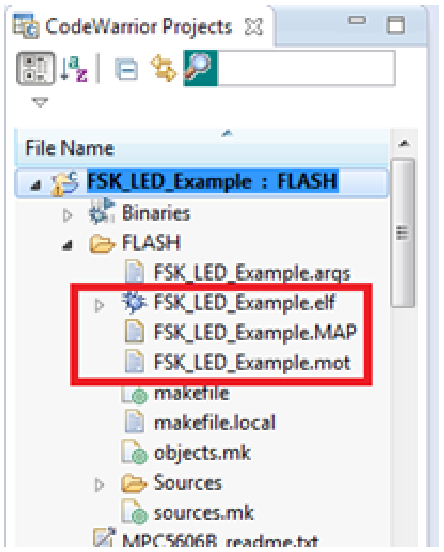


Figure 46. CodeWarrior build files

5 Flash code

Next you need to Flash the code using the RAppID Boot loader utility on to the target board. However, before flashing the code, ensure that the board is powered correctly and the jumpers are in the correct position.

5.1 Jumper settings and power connections

The TRK-MPC5606B board can be powered by external 12 V supply or using USB connector. When the board is powered by USB connector, SBC is not powered. Whereas, the SBC is powered when the board is powered using the external 12 V supply. Since, you are using CAN in this example, you need to power up SBC. Connect the J1 jumper across SBC_5V as shown below and connect external power to JP1. Using the USB cable provided with Fast Start Kit, connect the board to your computer. This provides a virtual serial port connection. You can confirm this by checking the available COM port using the Windows Device manager.

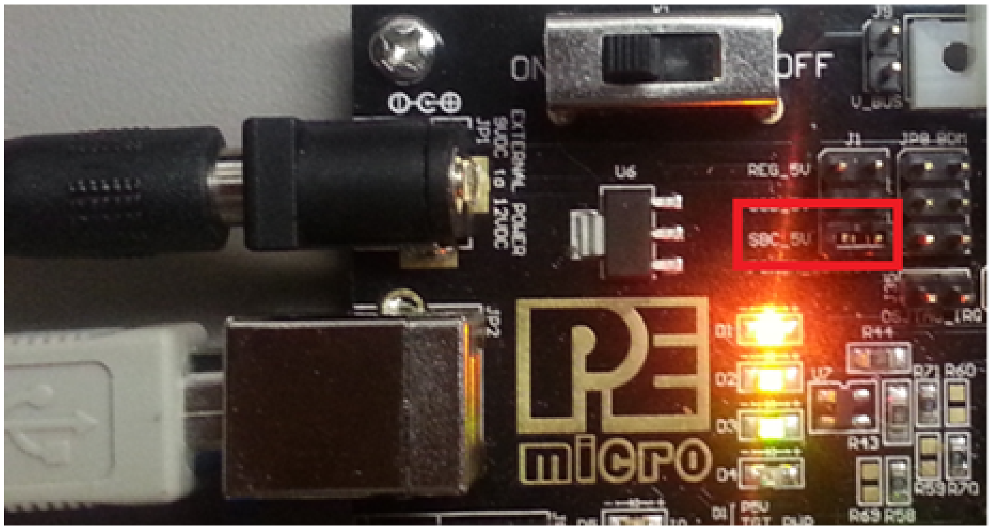


Figure 47. Connect the J1 jumper across SBC_5V

To enable the UART communication, connect jumper J7 (TXD_EN) and J8 (RXD_EN) to position 1-2 as shown below.

To enable CAN communication, connect J6_A (CAN_TXD) and J6_B (CAN_RXD) to position 1-2.

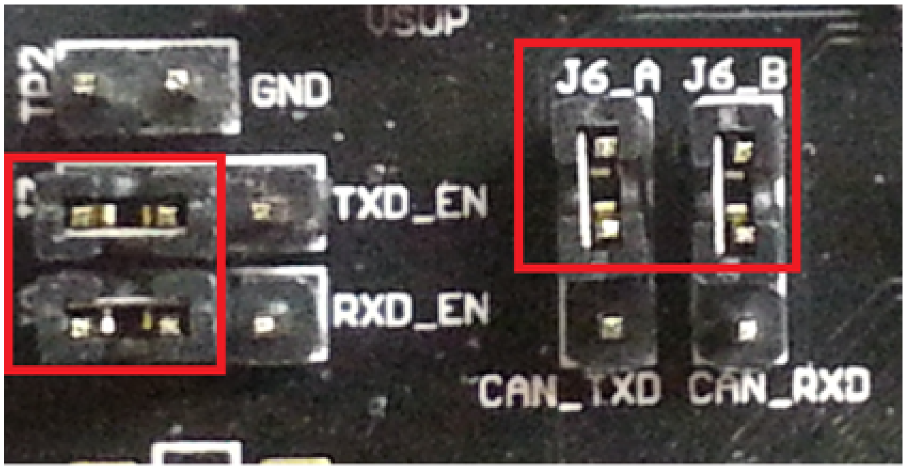


Figure 48. Connect UART and CAN jumpers

When the SBC is powered, the CAN_5v LED should be lit as shown below

To enable DSPI, connect all four jumpers in J38 as shown below.

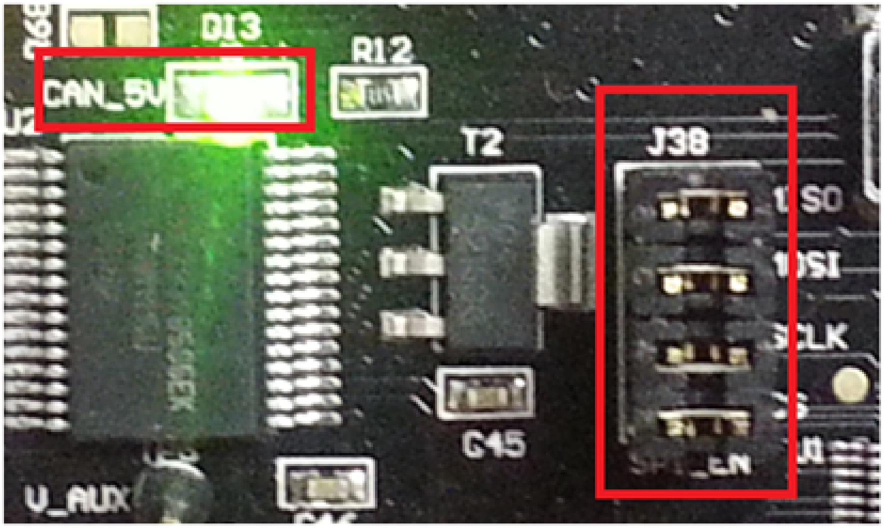


Figure 49. Connect DSPI jumpers

To enable LEDs, connect all four jumpers in J27 as shown below.

To enable input switches, connect all four jumpers in J26 as shown below.

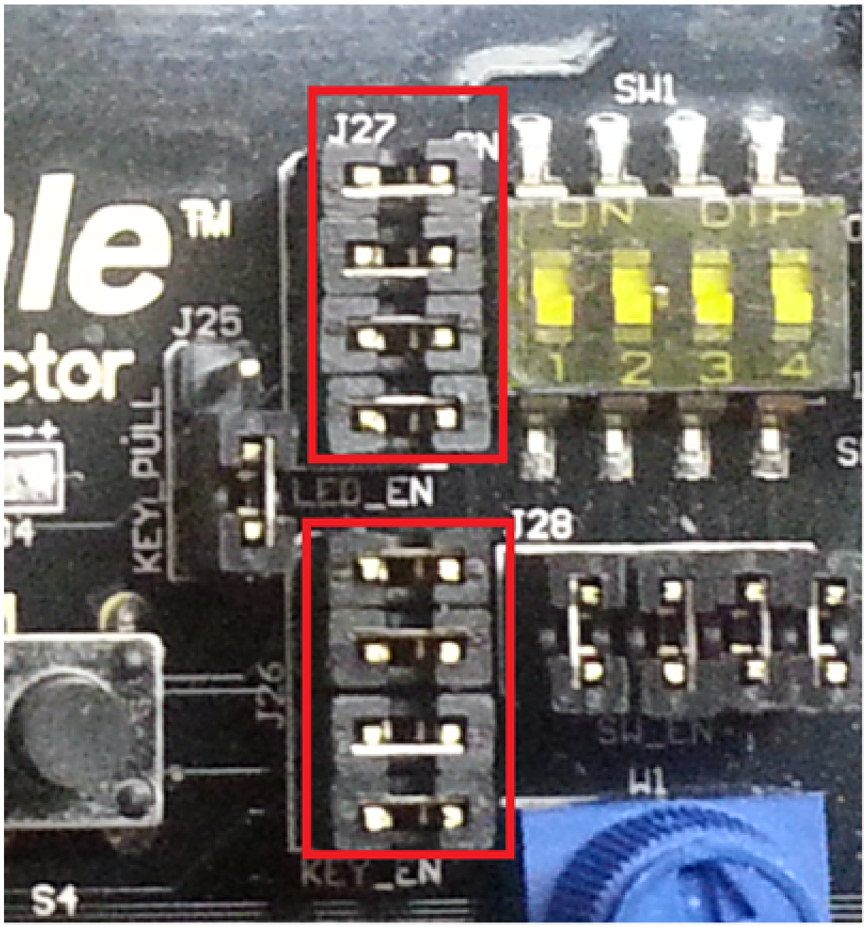


Figure 50. Connect LED and switch jumpers

5.2 Flash code using Boot Loader utility

To flash code:

1. Set the jumper of J17 to position 1-2 which pulls FAB high and jumper J18 to position 2-3 to set ABS low as shown below.

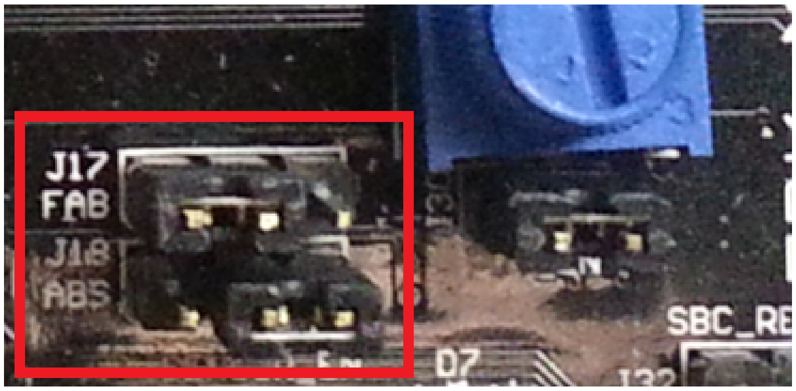
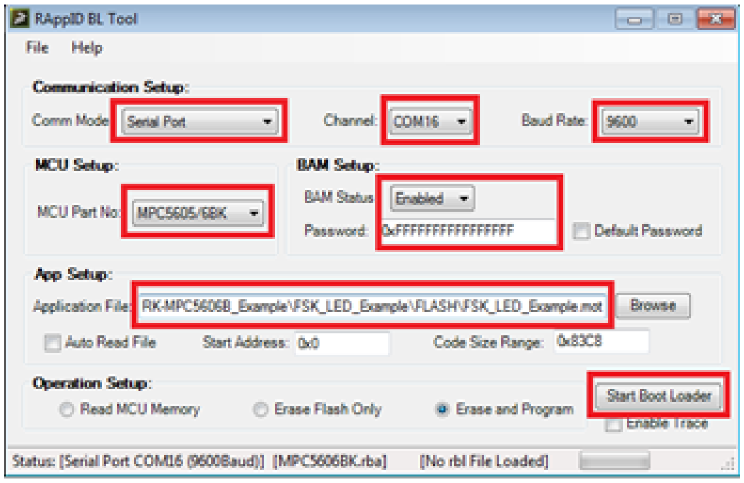


Figure 51. Jumper setting for Flash

2. Use the RAppID Boot loader utility to flash S-record file FSK_LED_Example.mot using serial port.
3. Launch RAppID Boot loader utility from Windows Start menu.
4. Set the values as shown in the RAppID Boot loader utility application window below.



Communication Setup:
Comm Mode: Serial Port
Channel: COM16
Baud Rate: 9600

MCU Setup:
MCU Part No: MPC5605/6BK

BAM Setup:
BAM Status: Enabled
Password: 0xFFFFFFFFFFFFFFFF
Default Password

App Setup:
Application File: RK-MPC5606B_Example\FSK_LED_Example\FLASH\FSK_LED_Example.mot
Auto Read File
Start Address: 0x0
Code Size Range: 0x83C8

Operation Setup:
Read MCU Memory
Erase Flash Only
[✓ Erase and Program]
Enable Trace

Status: [Serial Port COM16 (9600Baud)] [MPC5606BK.rba] [No rbl File Loaded]

Figure 52. RAppID Boot loader application window

5. Click the Start Boot Loader button to start the flash process. When asked to cycle power to MCU, press the reset button on the board. Flashing process should start.
6. After Flash is complete, the code should begin executing.
7. To have the code execute after a power cycle, move the jumper J17 to position 2-3 to pull FAB low as shown in the figure below.
8. Turn the power off to the module and re-apply the power. The code should be running now.

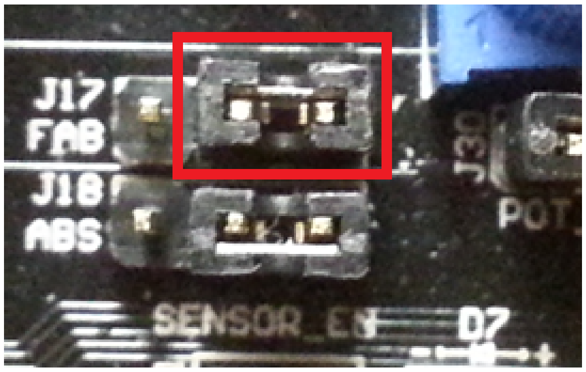


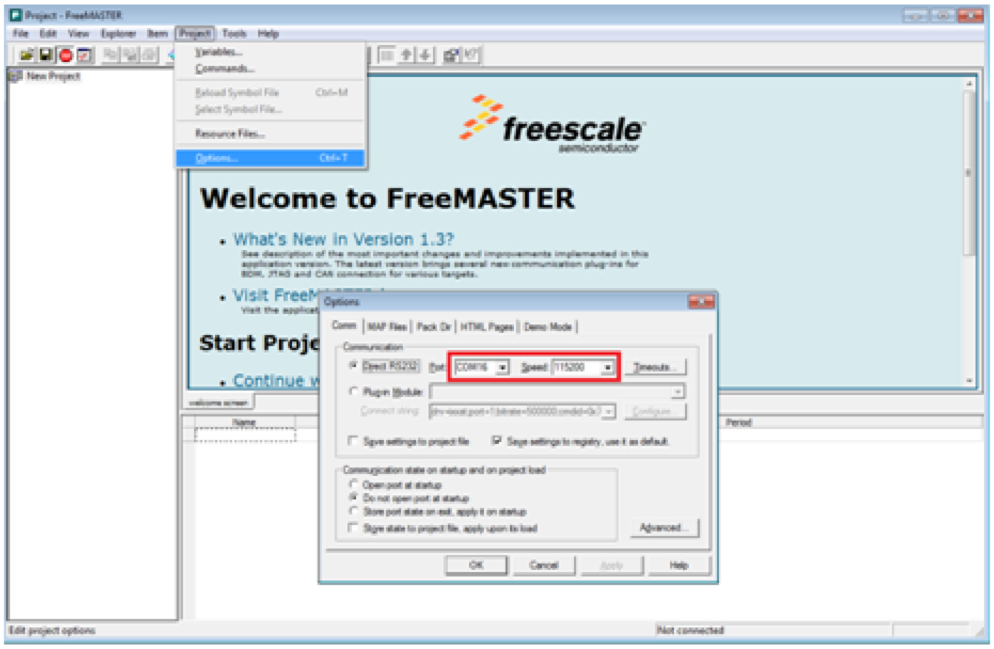
Figure 53. Jumper setting after Flash

6 Test code

6.1 Monitoring variables using FreeMASTER

Use FreeMASTER to monitor global variables used in this example project.

1. Launch FreeMASTER utility from Windows Start menu.
2. After the application is started, select *Project* > *Options* > *Communication* from menu and set the communication port number and baud rate of 115200 as shown below.



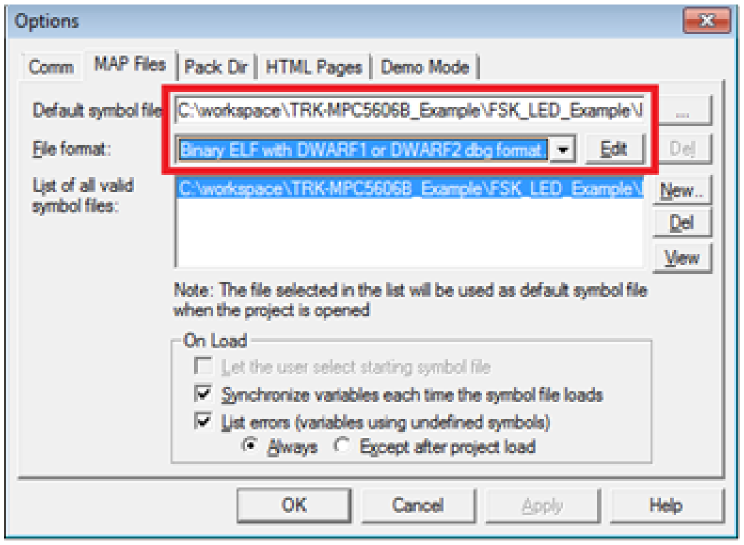
Direct RS232 Port: COM16 Speed: 115200

Save settings to project file
Save settings to registry, use it as default

Communication state on startup and on project load
Open port at startup
Do not open port at startup
Store port state on exit, apply it on startup
Store state to project file, apply upon its load

Figure 54. FreeMASTER communication options

3. Select the example application MAP file. In this example, the MAP file is the *.elf* file generated during the build process. FreeMASTER uses the information about the variables, their names, types, and addresses contained in the *.elf* file.
4. From the MAP tab, select the MAP file as shown below.



Default symbol file: C:\workspace\TRK-MPC5606B_Example\FSK_LED_Example\

File format: Binary ELF with DWARF1 or DWARF2 dbg format.

List of all valid symbol files: C:\workspace\TRK-MPC5606B_Example\FSK_LED_Example\

Note: The file selected in the list will be used as default symbol file when the project is opened

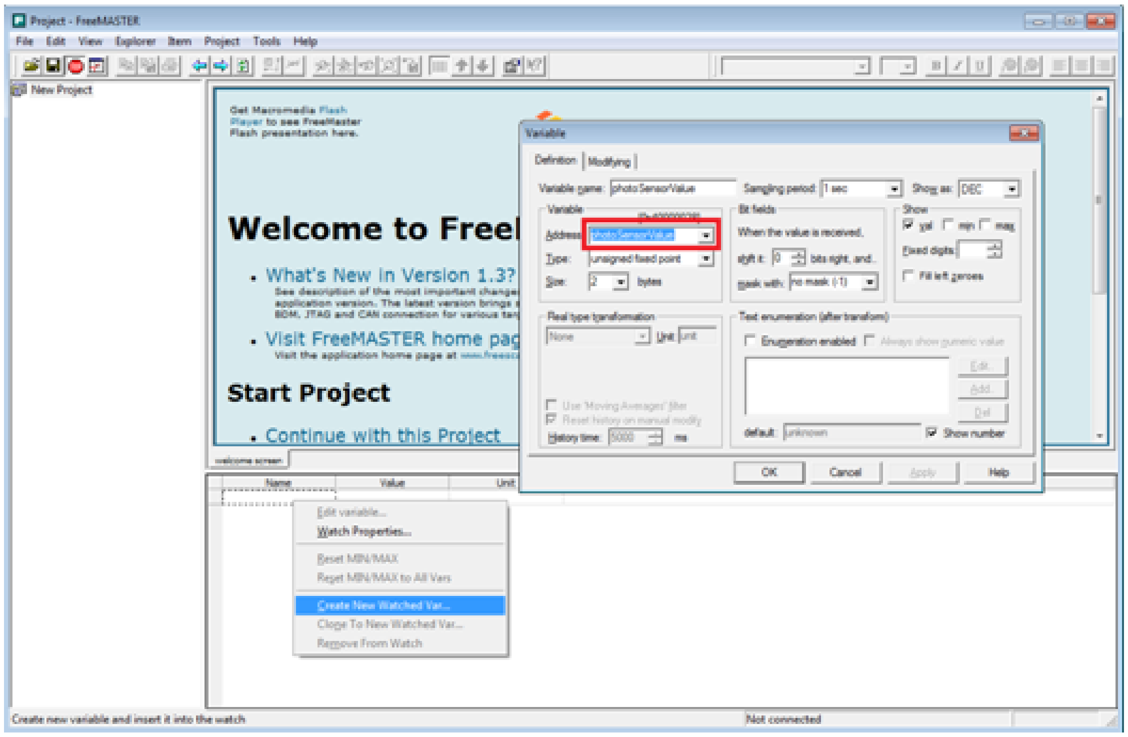
On Load

- [ ] Let the user select starting symbol file
- [x] Synchronize variables each time the symbol file loads
- [x] List errors (variables using undefined symbols)

[✓ Always] Except after project load

Figure 55. Select MAP file

5. Add the two global variables in this example project to monitor in the watch window – *photoSensorValue* and *potValue*.
6. To select variable to watch:
   - Right-click on the variable grid.
   - Select *Create New Watched Var*from the menu. This will pop up a variable selection window.
   - Select *photoSensorValue* from the drop down as shown below and select OK. This will add the variable *photoSensorValue* to watch window.
   - Using similar steps, add *potValue* to watch window.



Variable name: photoSensorValue

Sampling period: 1 sec

Show as: DEC

Address: photoSensorValue

Type: unsigned fixed point

Size: 2 bytes

When the value is received, shift it: 0 bits right, and mask with: no mask (-1)

Real type transformation: None

History time: 5000 ms

default: unknown

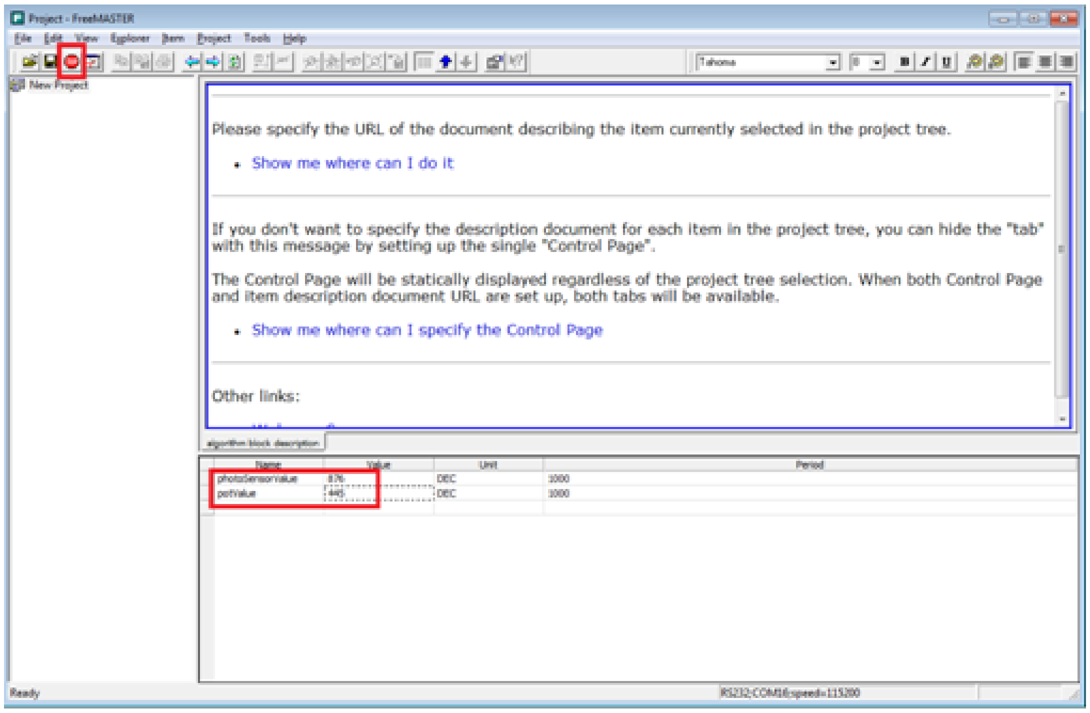
Figure 56. Add variable to monitor

7. Select the icon *Start/Stop communication* to start communication and observe the two watch variables getting updated.

6.2 Testing ADC inputs

When you rotate the potentiometer, the watch window should update. LED1 should turn on when the *potValue* is below 500 counts and it should turn off when the count exceeds 500.

The *photoSensorValue* should change when the sensor on the board is exposed to varying light source. LED2 should turn on when the *photoSensorValue* is below 500 counts and it should turn off when the count exceeds 500.



Please specify the URL of the document describing the item currently selected in the project tree.

- Show me where can I do it

If you don't want to specify the description document for each item in the project tree, you can hide the "tab" with this message by setting up the single "Control Page".

The Control Page will be statically displayed regardless of the project tree selection. When both Control Page and item description document URL are set up, both tabs will be available.

- Show me where can I specify the Control Page

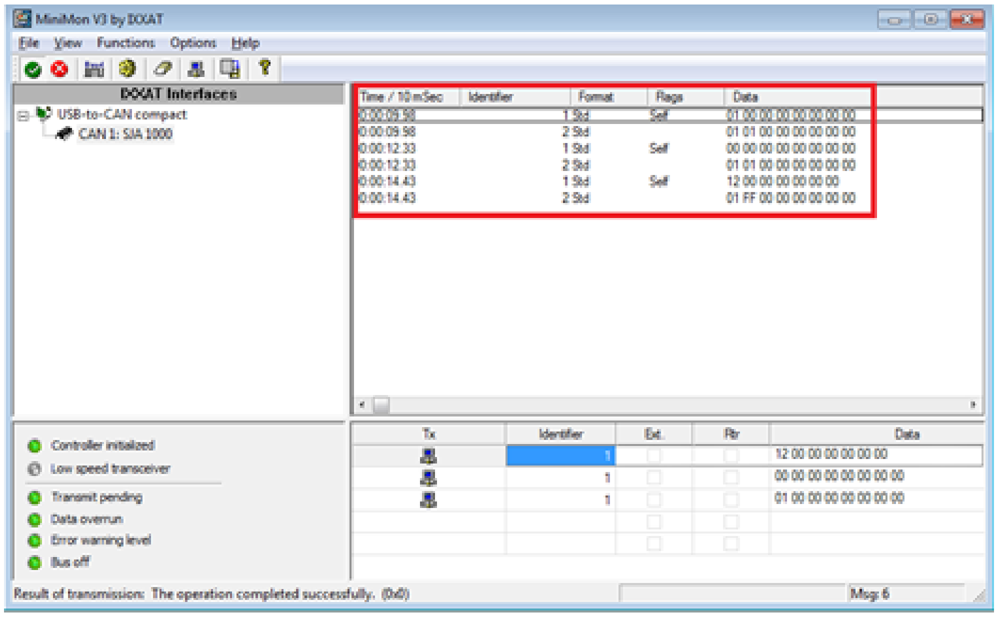
Other links:

Name	Value	Unit	Period
photoSensorValue	876	DEC	1000
potValue	445	DEC	1000

Figure 57. FreeMASTER main window

6.3 Testing CAN communication

You can use a CAN communication tool like CANalyzer or IXXAT MiniMon for sending and receiving CAN messages. Send a CAN message using with value of 1 in first byte to turn on LED3 and a value of 0 to turn off LED3. In both cases, you should see a positive CAN message transmitted by the microcontroller. When you send a CAN message with first byte other than 0 or 1, you should see a negative message being transmitted as shown in the figure below.



Time / 10 mSec	Identifier	Format	Flags	Data
0:00:09.98	1	Std	Self	01 00 00 00 00 00 00 00
0:00:09.98	2	Std		01 01 00 00 00 00 00 00
0:00:12.33	1	Std	Self	00 00 00 00 00 00 00 00
0:00:12.33	2	Std		01 01 00 00 00 00 00 00
0:00:14.43	1	Std	Self	12 00 00 00 00 00 00
0:00:14.43	2	Std		01 FF 00 00 00 00 00 00

● Controller initialized
● Low speed transceiver
● Transmit pending
● Data overrun
● Error warning level
● Bus off

Tx	Identifier	Ext.	Rtr	Data
	1	☐	☐	12 00 00 00 00 00 00
	1	☐	☐	00 00 00 00 00 00 00 00
	1	☐	☐	01 00 00 00 00 00 00 00
		☐	☐	
		☐	☐	

Result of transmission: The operation completed successfully. (0x0)   Msg: 6

Figure 58. Testing CAN communication

6.4 Testing PWM output

By default, LED4 output should be driven by PWM signal at 50% duty cycle. Connect PE7 pin on the TRK-5606B board to a scope to verify the LED4 output. The figure below shows the scope trace of the LED4 PWM output at 1 ms period and 50% duty cycle. Press S3 and S4 switches to confirm duty cycle increases or decreases.

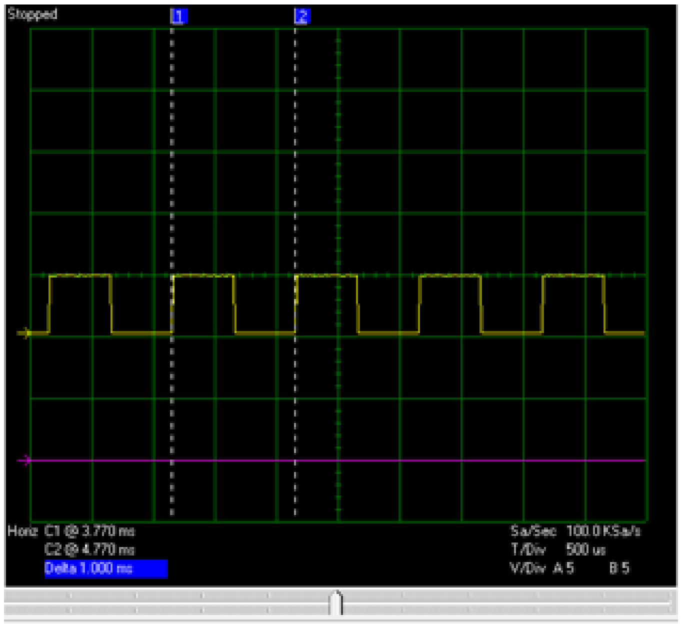


Figure 59. LED4 PWM output

7 Conclusion

The Qorivva Fast Start Kit for TRK-MPC5606B contains all the hardware and software tools with comprehensive documentation and examples to help developers get quickly started on application development for the Freescale MPC5606B microcontroller. This application note provides an overview of all the tools provided with the Fast Start Kit and an example providing step by step instructions which will help users better understand the process of application development using Freescale microcontrollers and development tools.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN4850
Revision 1, Mar 2014