

Document Number:AN4850 Rev 1, Mar 2014

Getting Started with Qorivva Fast Start Kit for TRK-MPC5606B

by: Sudhakar Srinivasa

1 Introduction

This application note provides an overview of software tools provided with the Fast Start Kit for the <u>TRK-MPC5606B</u>. 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,

© 2014 Freescale Semiconductor, Inc.

Contents

2 Overview of Qorivva Fast Start Kit for TRK-	
MPC5606B	1
3 Using RAppID tool to configure MPC5606B and generate code	3
4 Build code using CodeWarrior	28
5 Flash code	37
6 Test code	41
7 Conclusion	45





overview of Qorivva Fast Start Kit for TRK-MPC5606B

- 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.

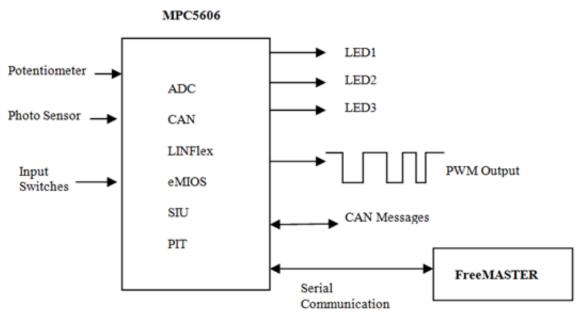


Figure 1. Block diagram of example application

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.



Using RAppID tool to configure MPC5606B and generate code

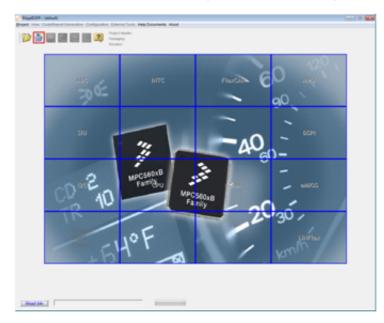


Figure 3. RAppID main window

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.

art Selection-					
Part	Core	Max. Speed (MHz)	Flash (KB)	RAM (KB)	
MPC5601D	z0	48	128	12	
MPC5602D	20	48	256	16	
MPC56058	z0h	64	768	64	
@ MPC56068	z0h	64	1024	80	
MPC56078	z0h	64	1536	96	
Load existing pro	ject				
			1		

Figure 4. Part selection

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



using RAppID tool to configure MPC5606B and generate code

Package Selection - MPC5606B : baseconfig *		×
Package Selection 208 BGA * 176 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: MPC560x8/C products in 208 MAPBGA packages.		
[Previous Next Exit Wizard	כ

Figure 5. Package selection

3.3 Configure ADC pins

In TRK-MPC5606B, the potentiometer is connected to input ANPO (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.

Pin Allocation - MPC56068 : baseco	nlig*			
Pin Allocation Wzard-				
			TOX PAD IHIB Allocated as Input THIS RAU THIB Allocated as Input TOX PAD CR Allocated as Drut TOX PAD CR Allocated as Drut PAS PAD PAB Allocated as Drut PAS PAD PAB Allocated as Drut ALC0_CC_1 MP 1 PAD PBS Allocated as Drut ADC_0_RAUC_1 MP 1 PAD PBS Allocated as Drut	
Functions	Input	Output	User Assigned Signal Name	
ADC_0_ADC_1 ANP 0	2.704		AVPO_Potentiometer_input	
ADC_0_ADC_1 ANP 1	2 P65		AVP1_PhotoSensor_input	
ADC_0_ADC_1 ANP 2	766		WY LOTIONOPHIC INPUT	
ADC_0_ADC_1 ANP 3	E #67			
ADC 0 ADC 1ANP 4	P P00			
1	=			
			Previous Next Exit Waard	Ven

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.

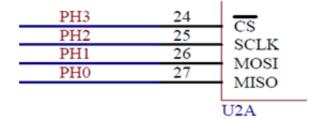


Figure 7. Connection between SBC and DSPI 1

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

N817050216	<u></u>		94 000000000000000000000000000000000000	
2 2 2 2 2 2 2 2	END COLUMPER MISC CANOS			formal Name
ADC 0591 FlexC4V CORD	Input	Output	User Assigned	
	Input E PA6	04put	User Assigned	
Functions		E PA6 E PO14 E PF6	User Assgred	
Punctions SPI_1 Chip Select 1 SPI_1 Chip Select 0	E PAS	E PA6 E PO14 E PF6		put
Functions SEPI_1 Chip Select 1	PA6 PA4 EPC3 EPE5 EP013 EPH3	PA6 P014 PF6	2 PH3_DSPI1_CS0_Outp	put put
Punctions SPI_1 Chip Select 1 SPI_1 Chip Select 0 SPI_1 Chip Select 0 SPI_1 Chick	PA6 PA4 EPC3 EPE5 EP013 EPH3	PA6 P034 PP6 PA4 PC3 PE5 P013 PE4 PC2 PE2	PH3_DSP11_CS0_OVp PH2_DSP11_CLK_OVp	put put put

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.



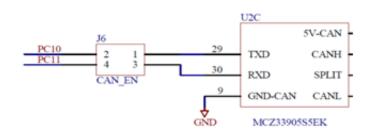


Figure 9. CAN TX and CAN RX pins of SBC connected to pins PC10 and PC11

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

	ordig "		TCCFAD PH9 Allocated as Input TCCFAD PH9 Allocated as Input TCF PAD PC0 Allocated as Input TCF PAD PC0 Allocated as Input PAD PAD PAB Allocated as Input PAD PAD PAB Allocated as Input ACC_0_ACC_1 APP 0 PAD PH4 Allocated as Input ACC_0_ACC_1 APP 0 PAD PH4 Allocated as Input DSPL 1 Obj Select 0 PAD PH4 Allocated as Input DSPL 1, 1 Obj Select 0 PAD PH4 Allocated as Output DSPL 1, 1 Obj Select 0 PAD PH4 Allocated as Output DSPL 1, 1 Obj Select 0 PAD PH4 Allocated as Output DSPL 1, 1 Obj Select 0 PAD PH4 Allocated as Input OSPL 1, 1 Obj Select 0 PAD PH4 Allocated as Input OSPL 1, 1 Obj Select 0 PAD PH4 Allocated as Input CAN_1 The PAD PH2 Allocated as Input CAN_1 The PAD PC11 Allocated as Input	
NOC DSP FlexCAN Case 12 Functions	Input		User Assigned Signal Name	
RUE DEPENDENT CAN CONS		Ovput		
0 0	Input		PC10_CAN1_Tx	
RUE DEPENDENT CAN CONS	Input	Ovput	PC10_CAN1_Tr	

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.



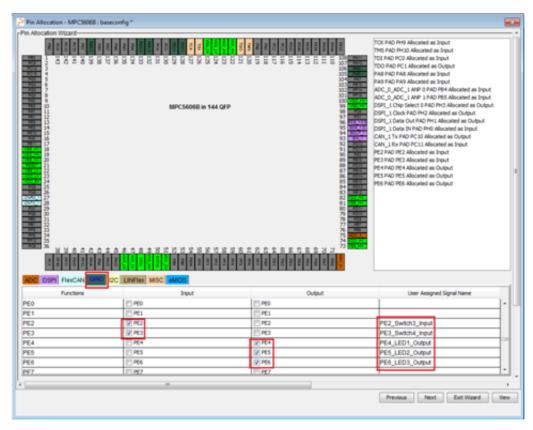


Figure 11. Configure GPIO pins

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.

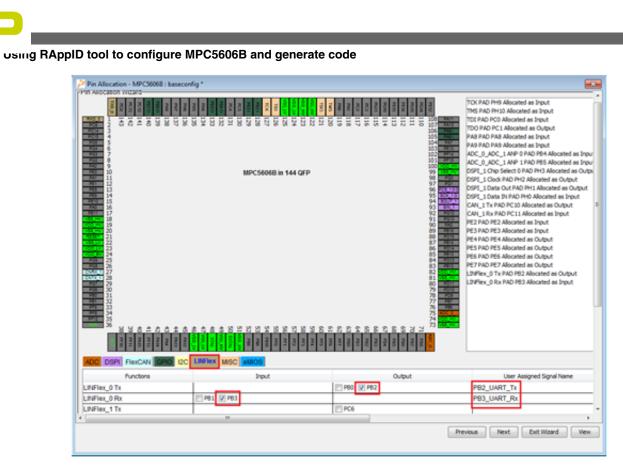


Figure 12. Configure LINFlex (UART) pins

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.



Using RAppID tool to configure MPC5606B and generate code

	1 8 9 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1	200 200 200 200 200 200 200 200 200 200	TO THO PC3 MICRIME as Curput PAI PIAD PIAB Allocated as Tipust PAI PIAD PIAB Allocated as Tipust ACC_B_ADCL_1 ANP 9 TAID PIB4 Allocated as Tipust ACC_B_ADCL_1 ANP 9 TAID PIB4 Allocated as Tipust DSPL_1 Disk Delete 1PAD PIB4 Allocated as Curput DSPL_1 Disk DVAD PI12 Allocated as Curput DSPL_1 Disk DVAD PI12 Allocated as Curput CAL_1 TI: PIAD PC23 Allocated as Curput CAL_1 TI: PIAD PC23 Allocated as Tipust PC3 PIAD PI22 Allocated as Tipust PC3 PIAD PI22 Allocated as Tipust PC3 PIAD PI22 Allocated as Tipust PC3 PIAD PC3 Allocated as Tipust PC3 PIAD PC4 Allocated as Topust PC3 PIAD PC4 Allocated as Topust PC4 PIAD PC4 Allocated as Topust PC4 PIAD PC4 Allocated as Topust PC5 PIAD PC5 Allocated as Topust PC5 PIAD PC6 Allocated as Topust	
AL NO.				
OC DSPI Flex Oarnel				5
Cite Control C	Punctions Default Mode Default Mode	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	Output User Assigned Signal Riame PES PEB PEB PFS	
CC DSPI Flax Channel MIDS_0 C+C21 MIDS_0 C+C22 MIDS_0 C+C22	Punctions Default Mode Default Mode Output Pulse Width and Prequency Modulation Buffered	Pps PPS -	Output User Assigned Signal Name PDS PDS PDS PDS PDS PDS PDS PDS	
	Punctions Default Mode Default Mode	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	Output User Assigned Signal Riame PES PEB PEB PFS	

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.



Mode Node Node Node Node Node Node Node N			TEST Mo DRUN M RUN1 M RUN3 M	ode ode			
Mode Mode Mode			RUN1 M	:de	0		
Mode Mode							
Mode Mode							
Mode	а а		RUN3 Mo	de	01		
	04						
EVO Mode			STOP0 M	lode	01		
	01						
	a						
puration							
SPDO SM/R	SDFLACE	\$ CFLA ON	\$ PLL0 ON	\$ XDSC0 ON	\$ 16MHz_IRC ON	\$ SYSCLK	
- Cl	Normal	Normal	20	00	- 19	15PHz internal RC escillator	
- CE	Normal	Normal	00	00	10	16HHz internal RC oscillator	
	Normal	Normal	01	00		1944 internal RC escillator	
DI DI	Carlos Lucio		100		101	System PLL	
01 (1) (1)	Normal	Normal	D1	D1		Shandari Let	
04 E		Normal		01		1044-U wterna RC oscillator	
24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Normal		N 51	01			
22 24 24	Normal Normal	Normal	08 08 08	01	69	10PPU internal RC oscillator	
01 01 01 01 01 01	Normal Normal Normal	Normal	04 05	CH DE	851 851	1999 La internal RC oscillator 1999 te internal RC oscillator	
04 04 04 04 04 04 04 04 04 04	Normal Normal Normal Normal	Normal Normal Normal	04 06 08	00 00 00	10 10 10	1979 La internal RC osciliator 1979 ti internal RC osciliator 1979 ti internal RC osciliator	
04 0 08 0 04 0 04 0 04 0 04 0 04 0	Normal Normal Normal Normal Normal	Normal Normal Normal	64 64 64 64	01 05 05 01	0 15 15 10 10	1974 U internal RC oscilator 1974 ti internal RC oscilator 1974 ti internal RC oscilator 1974 ti internal RC oscilator	
I	SPDO SM/RC	SPDO SIMURION SDFLAION	SPDO SIMRON SDFLAON SCFLAON	SPDO SIMPON SDFLAON SCFLAON SPLLOON	SPDO SM/R ON SDFLA ON SCFLA ON SPLL0 ON SX05C0 ON 52 Vormal Normal 53 53	SPDO SMARON SDFLAON SCFLAON SPLLOON SXDSCOON S10MHL_RCON	SPDO SM/R ON S DFLA ON S PLLO ON S X05C0 ON S 10MHL_IRC ON S 31'SCLK Cit Vormal Tormal Cit Tormal 10PHz internal IC socilator

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.

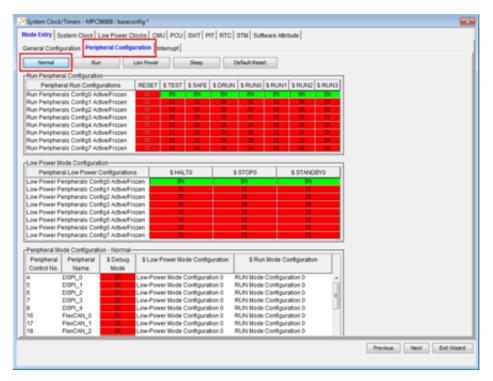


Figure 15. Mode Entry peripheral configuration



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/Timers - MPC56068 : baseconfig *			
Node Entry System Clock Low Power Clocks CN	AU POU SWT PIT RTC STM Software Attribu-	ute	
System Clock and Peripheral Clock Setup			
System Clock Setup			
DRUN Mode Clock System RL	XOSC Frequency 8.00 HHz	System Clock Frequence	y 64.000 MH2
Clock Source Configuration		Bystem Clock Frequen	ly Formula
Clock S Bypass S Divider	S End of Count S Interrupt B S TRM	System Clock Frequence	
2050 DK 1 12		Final Reference Fre	
15kBet IRC]1			
PLL Setup		Clock Output Setup	
Progressive Clock Switching	PLL Frequency Modulation Setup	Clock Output	
Input Delsion Factor 2 -	Frequency Modulation	Clock Output Select	Orystal Osofietor
Output Division Factor	Spread Type Center Spread _ v	Clock Divider	1 *
Loop Division Factor 84 *	Modulation Period D 0	Clock Output Frequency	8.000 MHz
Final Raterence Frequency Multiplier 8	Increment Step D 0	Output Clock Pin	16 (PAD) - Not Assigned
		Peripheral Clock Selup-	
PLL OP Frequency 64 SHz		Peripheral Clock Set	SDwider SDwider Final Clock Status (MHz)
VCO Frequency 256 MHz		Set1 (LINFIer & I2C)	Di 64.000
Final Reference Prequency Butgler + Loop Dyson Factor PLL O/P Prequency + XDSC Prequency * Final Reference F		Set2 (FlexCAN & DSPI) Set3 (eMIOS & ADC/CTU)	Di 1 54.000 Di 1 54.000
VCD Frequency + XDSC * (Loop Division Factor / Input Div	(alon Factor)	(HHOS & ADD(10)	
			Previous Next Extillated

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.



using RAppID tool to configure MPC5606B and generate code

System Clock/Times - MPC56068 : baseconfig *	_			
Mode Entry System Clock Low Power Clocks CMU	PCL SWT PIT RTC :	STM Software Attribute		
(Watchdog Control				
Enable Watchdog Timer	Clock Source	IRC 128 KHz		
Reset on Invalid Access Di	Window Mode	24		
Interrupt then Reset	Hard Lock	24		
SoftLock 26	Stop Mode Control	21		
Debug Mode Control Bi	Window Value	Do		
Watchdog Timeout Value D 1200				- U
rWatchdog Timeout	1 r Watchdog Window Star	f Pariod		1
Clock Source Frequency: 128 KHz		low Time in Clock Cycles: 0		
Watchdog Timeout Range: 10 ms to 33554.43 sec	Service Sequence Wind			
Watchdog Resolution: 0.008 ms	Window Time as Percer	ntage of Timeout Period: 0%		
Watchdog Timeout in Clock Cycles: 1280				
Watchdog Timeout. 90 ms				
×[•
			Previous Next	Exit Waard

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.



Using RAppID tool to configure MPC5606B and generate code

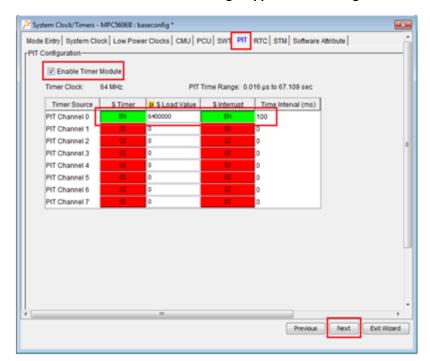


Figure 18. PIT configuration

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.

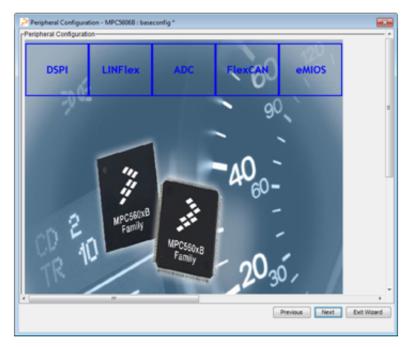


Figure 19. Select DSPI



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

PLO DSPL1	DSPL2 DSPL3 DSPL4 Software Attributes		
eneral Configura	tion Clock and Transfer Attributes SPI Commands	DMAInterrupts	
SPI General Co	nfiguration		1
	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 v	
	Peripheral Chip Select Line 4 inactive state	Disable v Disable v Disable v Low v Low v Low v Low v	
	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 👻 Disable 👻	
	Continuous Serial Communication Clock	Disable 💌	
	Halt Mode	Disable 💌	
	Module	Enable 💌	
	Transfer Count On Reset	DØ	

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.



Using RAppID tool to configure MPC5606B and generate code

Local Interconnect Network - N			-
	2 UNRECT UNRECT UNRECT	Software Attribute	
Ceneral Configuration Interrup	pts,DMA & Filter Timeout UART		
Master Mode		LINFlex System Clock (MHz)	64 MHz
	_	Loopback Mode	0
Receiver Buffer Locked Mode		Self Test Mode	
LIN Master Break Length	10 bits -	Slave Mode Break Detection Threshold	11545 *
Bypass Filter	EN .		1100 -
Auto Wakeup	DI	LIN Auto Sync	-
Checksum Field	EN .	Idle on Bit Error	EN .
Programmed Checksum		Checksum Calculation	0
Integer Baud Rate Factor	D 34	Idle on Identifier Parity Error	EN
Baudrate(Symbols/Sec) =	115107.91	Fractional Baud Rate Factor	12/16
		Lueur-	
		Baudrate (Symbols/Sec) + Fcpu / (16 * LFD UFDIV + Integer factor(Mantissa) + Fraction	
Getting Started		Ok Car	cel Apply

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.

Cocal Interconnect Network -			
	C2 UNFIRC3 UNFIRC4 UNFIRC5 pts.CMA & Filter Timeout UART	Software Attribute	
	pts,DMA & Pitter Timeour UNKT		
UART Configuration-	E:	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	D 4095		
Transmitter BuffenFIFO Size (Bytes)	1	Receiver Buffer/FIFO Size (Bytes)	1
Transmit FIFO	CL	Receive FIFO	DI
Transmit Buffer Size:	D 1	Receive Buffer Size:	D 1
Note: To Enable Automatic DHA	TCD DHATX[0]/ DHARX[0] should be enabled	from Interrupts,OHA & Filter Tab.	
•			
Getting Started		Ok	Cancel Apply

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.

Ż	ADC CTU - MPC56068 : baseconf	fig *			
1	ADC_0 ADC_1 ADC CTU Som	ware Attributes			<u>^</u>
Г	Device Setup Sampling Control	Channel Setup Analog W	/atchdog Setup Presampling Setup Interrupt	DMA Configuration	
٦	General Configuration				
	ADC System Clock (MHz)	64 MHz	Number of ADC Threshold Analog Wate	thdogs : 6 (ADC_0)	
	ADC Resolution(bits)	: 10 (ADC_0)	Conversion Mode	One Shot 👻	
	Overwrite	DL	Alignment	Right 💌	
	Auto Clock Off	DI	Decode Signal Delay (ADC Clock Cycles)	D 0	
	Power Down Delay (ADC Clock Cycles)	D 0	Power Down Enable	DI	
	ADC Clock Select	System dock / 2 💌			
	Trigger Mechanism				
	Injected Mode Trigger-				
	Triggered injection	DI	Trigger Injection edge	Faling v	
	Note: To enable PIT channel 2 tr	iggering for ADC 0 conversi	on, Trigger Injection button should be enabled.		
	CTU Trigger Configuration-				1
	ADC/Cross Triggering Unit (CT	U) DI			
L					-
[Getting Started			Ok Cancel	Apply

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.



Using RAppID tool to configure MPC5606B and generate code

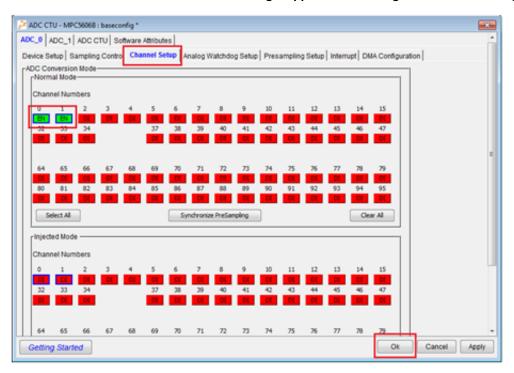


Figure 24. ADC channels configuration

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.



AN Settings Interrupt Mask Rec	eive Buffer Mask				
PRICAN	_	_		_	_
llodule:	Enable	-	Max Message Buffer:	D 16	
Soft Reset	Disable	*	Bus Off Recovery:	Enable	٣
Freeze Enable:	Disable	×	Lowest Buffer Transfer First	Disable	٠
Halt FlexCAN:	Disable	*	Loop Back Mode:	Disable	Ŧ
Clock Source:	System	*	Listen Only mode:	Disable	Ŧ
TexCAN System Clock (MHz):	64		Timer Synch Mode:	Disable	* * * * * * * * * * * * * * *
CAN Speed (k bits/s):	500	*	Supervisor Mode:	Supervisor mode	٣
CAN Bit Timing			Self Reception:	Enable	Ŧ
Prescaler Division Factor:	D 8		Backwards Compatibility Configuration(BCC):	Enable	٣
Phase Segment 1:	4	*	Abort Enable:	Disable	Ŧ
Phase Segment 2:	4	*	Local Priority Enable:	Disable	Ψ
Propagation Segment	7	Ŧ	ID Acceptance Mode:	A	٣
Resynchronization Jump Width:	1	Ŧ	FIFO Enable:	Disable	٠
CAN Bit Sampling Mode:	1 Sample	*	Warning Interrupt Enable:	Disable	٣
Sample Point (%):	75				
Serial Clock Frequency (MHz):	8				
Enable CAN2.08 Checks					
Note While setting Phase Segment 1, P Propagation Segment following the 1) Phase Segment 2 ≥ Phase Seg ≥ 8 3)Propagation Segment + Phase S	ee condition shoul gment 1 ment 2 + Propagati	d be considere	FlexCAN System Clock		

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.



Using RAppID tool to configure MPC5606B and generate code

2	Enhanced M	odular Input Ou	rtput System - N	MPC5606B : bas	econfig *			
2	MIOS_0 eMI	OS_1 Softwa	re Attributes					
ŀ	Adule Config	uration Char	nel Configurat	ion				
٢	General Modu	le Configuratio	n					
	Module	Enable 💌						
		ug Freeze Con						
	Contract Con		are set to FRE	EZE their regis	ter when DEB	JG mode is se	lected	
	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 En	able						
	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	V CH31
							Clear Al	Select All
	🗹 Global T	'ime base Ena	ble					
	And Address of the Owner of the	Global Prescal bal Prescaler D		D 1	64 MHz F	Prescaler Frequ	Jency	
	Getting Sta	arted					Ok Ca	ncel Apply

Figure 26. eMIOS configuration

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



using RAppID tool to configure MPC5606B and generate code

fied Channel Setup-	annel Configuration
Channel 00	Channel Mode: when entering an output mode (except for GPIO mode) the output flip flop is set to the complement of the Edge polarity.
Channel 01	Select Channel Mode
Channel 02 Channel 03	Output Pulse Width and Frequency Modulation Buffered
Channel 04 Channel 05 Channel 05 Channel 07 Channel 08 Channel 09 Channel 10	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
Channel 11 Channel 12 Channel 13	Config 1 Config 2 Config 3
Channel 14 Channel 15 Channel 16 Channel 17	Logic Level on Output Pin Set on Notch A Set
Channel 18 Channel 19 Channel 20 Channel 21	Enable Interrupt on FLAG Flag Generation Assigned to Interrupt DMA/CTU
Channel 22 Channel 23 (PIN-140) Channel 24	Automatic DMA/TCD Setup I/O Buffer Size D 0
Channel 25	
channel 26	 DMA Channel for Transmit TCD Setup:

Figure 27. eMIOS Channel configuration I

fied Channel Setup		
Channel 00 Channel 01 Channel 02 Channel 03 Channel 05 Channel 05 Channel 07 Channel 08		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. Config2 - match value for A1 must be less than match value for B1
Channel 09 Channel 10 Channel 11 Channel 12 Channel 13 Channel 14 Channel 15 Channel 15 Channel 16 Channel 18 Channel 19 Channel 20 Channel 21 Channel 22 Channel 22	0)	Config 1 Config 2 Config 3
Channel 24 Channel 25 Channel 26		

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).

Enhanced Modular Input Output	t System - MPC56068 : baseconfig *	
Module Configuration Channel	Configuration	
-Unified Channel Setup		_
Channel 00 Channel 01 Channel 01 Channel 02 Channel 03 Channel 04 Channel 05 Channel 05 Channel 05 Channel 07 Channel 07 Channel 00 Channel 10 Channel 11 Channel 11 Channel 15 Channel 15 Channel 15 Channel 16 Channel 19	Config 1 Config 2 Config 3 Bus Select Internal counter Force match on Comparator A Enter time of leading edge of PWM in A1	
Channel 20	Force match on Comparator B	
Channel 21	Enter time of trailing edge of PWM in B1 D 64000	
Channel 22		- 11
Channel 23 (PIN-140)	Enter ALTA channel data D 0	
Channel 24 Channel 25		
	Disable Transfers from A2 -> A1,B2 -> B1	- 11
CAMPACITY AND		
Getting Started	Ok Cancel A	pipily]

Figure 29. eMIOS Channel configuration III

This completes the configuration of peripherals.

4. Select *Next* to configure PIT interrupt.



using RAppID tool to configure MPC5606B and generate code

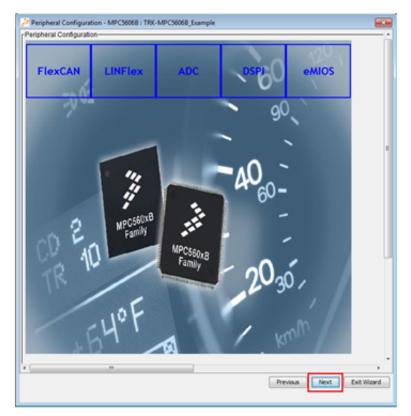


Figure 30. Completing peripheral configuration

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.



Using RAppID tool to configure MPC5606B and generate code

🄁 Interrupt Confi	guration - MPC5606B : TRK-MPC5606B_E	ample *			*
Interrupt Excep	tion Software Attributes				-
General System	Configuration	Software *			
Select the ve		4bytes ×			
	of the INTC you wish to configure below	V :			
Interrupt Source	Interrupt Vector	\$ Priority	Core(s)	SFunction	
MC_ME SIU	PIT Channel 0	1	e200z0	PIT_Ch0_ISR	
SWT	PIT Channel 1	0	e200z0	interrupt_handler	_
STM	PIT Channel 2	0	e200z0	interrupt_handler	
eMIOS	PIT Channel 3	0	e200z0	interrupt_handler	1
ECC	PIT Channel 4	0	e200z0	interrupt_handler	
PIT	PIT Channel 5	0	e200z0	interrupt_handler	
ADC MC_RGM	PIT Channel 6	0	e200z0	interrupt_handler	
LINFlex	PIT Channel 7	0	e200z0	interrupt_handler	
DSPI eDMA RTC FlexCAN Software MCM WKUP XOSC					
			Previous	Next Ext Wa	ard

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.



using RAppID tool to configure MPC5606B and generate code

Project Configuration: base	config *				×
Project Settings Code Ge	meration Report Ger	neration External Too	Initialization S	equence	
Code generation Setur					_
Core Startup Code	Flash BIU	Peripheral CAN Sampler	DSPI	Timers MIOS	1
RGM System Clock	V System ink	V ADC V FlexCAN	MPU	SIU eDMA	
Compiler Config	Codewarrior	-	Selec	t All Clear All	
Source Compliance	ANSI C	*	View C	ode on completion	
			C Autorur	n Consistency Chec	k
			Generation	ate LCF with Code	
			🔽 Genera	ate Code For Flash	
Source Path C:Frees	cale\RAppID\RAppID_	Projects\TRK-MPC560	068_Example		
			Ok	Cancel	pply

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.



Using RAppID tool to configure MPC5606B and generate code

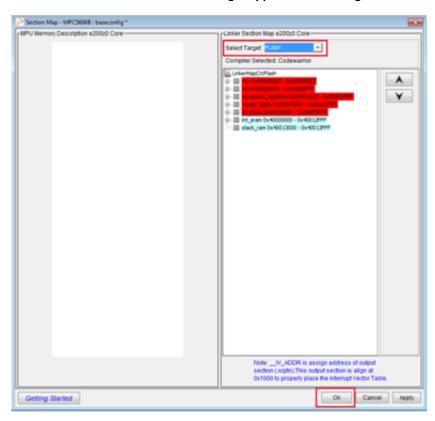


Figure 33. Section map

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.



Dund code using CodeWarrior

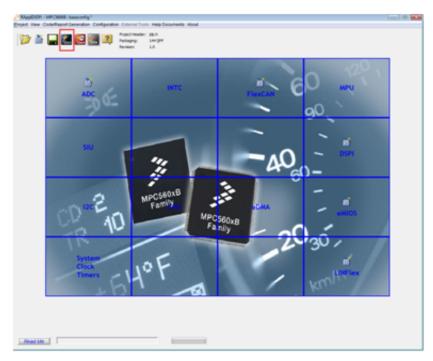


Figure 34. Generate code

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.



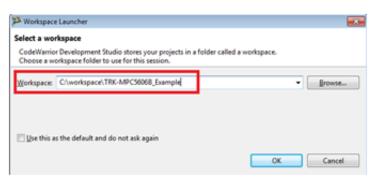


Figure 35. CodeWarrior Workspace Launcher

To create an empty bare board project:

1. Select New MCU project from Commander window.

 Project Creation 	▼ Settings
Import project	Project settings
Import example project	Build settings
Import MCU executable file	E Debug settings
19 New MCU project	▼ Miscellaneous
New MQX-Lite project	Welcome screen
 Build/Debug 	Quick access
🗞 Build (All)	🖇 Flash programmer
Clean (All)	
📴 Debug	

Figure 36. Start New CodeWarrior project

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

PNew Bareboard Project	
Create an MCU Bareboard Project	
Choose the location for the new project	
Project name: FSK_LED_Example	
Vse gefault location	
Location: C/workspace\TRK-MPC5606B_Example\FSK_LED_E	Browse
(?) < Back Next > Einish	Cancel

Figure 37. Provide a project name

3. Select 5606B device and select Next.



Dund code using CodeWarrior

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	
MPC56028	
MPC5602C	
MPC5602D	
MPC56038	
MPC5603C	
MPC5604B	
MPC5604C	
MPC5605B	
MP C20070	
Project Type / Output:	
Application	
Clibrary	
Creates project for MPC5606B derivative	
(?) < Back Next > Einish	Cancel

Figure 38. Select devices

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

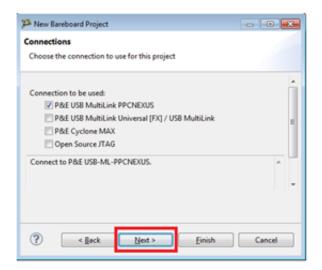


Figure 39. Select connections





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 🛛			
📰 Jªz E	i 🔩 🔎 🕅	e Name	~
File Name		Build	
FSK_LED_Example	: FLASH 👻		
	DAM		
	✓ FLAS	н	
			,

Figure 41. Select FLASH build option



ound code using CodeWarrior

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_O1.exe*) or by using RAppID menu *External Tools->CWInterface*.



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
xe\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.

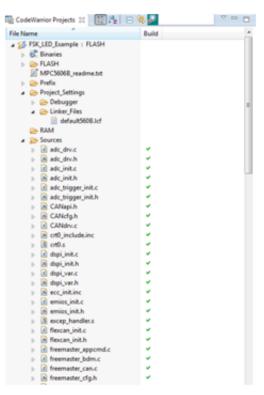


Figure 44. CodeWarrior Projects view

- 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_Examp	k		0 0
type filter text	Paths and Symbols		🌼 • 🔅 •
Resource Builders C/C== Build C/C== Build C/C== General Code Analysis Documentation	Configuration RASH (Activ	e) և Libraries 📣 Seurce Location 📄 References	• Manage Configurations.
File Types Formatter	Languages	Include directories	Add_
Indexer Language Mappings	Assembly Source File GNU C	S(ProjDirFath)/Sources/	Edt.
Paths and Symbols Preprocessor Include Pr	GNU C++		Delete
Project References Rury/Debug Settings	Object File		Export
	🖉 Show built-in values 👷 Import Settings 👔	Export Settings	Move Up Move Down
· ·			Restore Defaults doply
۲			OK Cancel

Figure 45. Adding Path for assembly source

4.3 Add application code to auto generated code



Dund code using CodeWarrior

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
#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);
/* 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 */
void main(void)
ł
/* ___
     /*
       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();
```

Build code using CodeWarrior

```
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 */
   CanTxMsq (2, 1, 8, (uint8 t *)msqErrorCAN, 0);
Ì
      *
   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);
ł
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);
```

Getting Started with Qorivva Fast Start Kit for TRK-MPC5606B, Rev 1, Mar 2014

ProcessCAN();



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)
ł
    uint32 t tmp;
    PIT.CH[0].TFLG.R = 0 \times 00000001;
/* 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))
    ł
        /* 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.



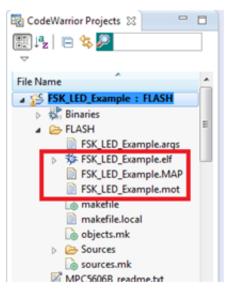


Figure 46. CodeWarrior build files

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.



riasi) code

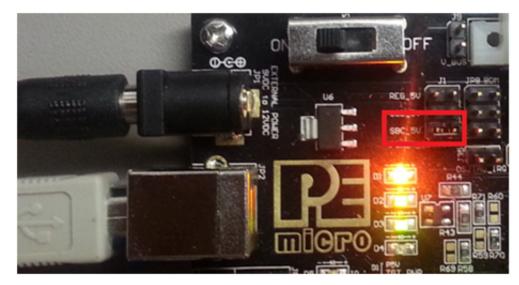


Figure 47. Connect the J1 jumper across SBC_5V

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.

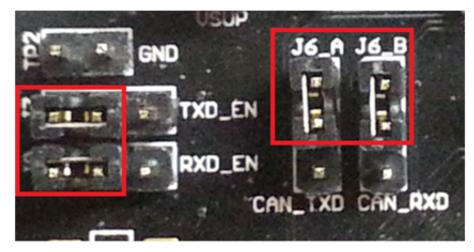


Figure 48. Connect UART and CAN jumpers

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.



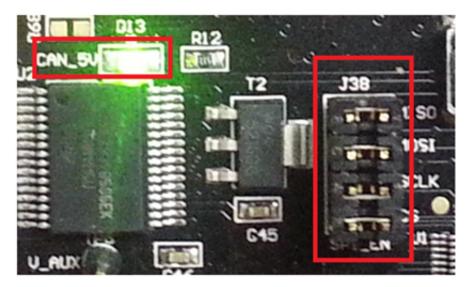


Figure 49. Connect DSPI jumpers

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.

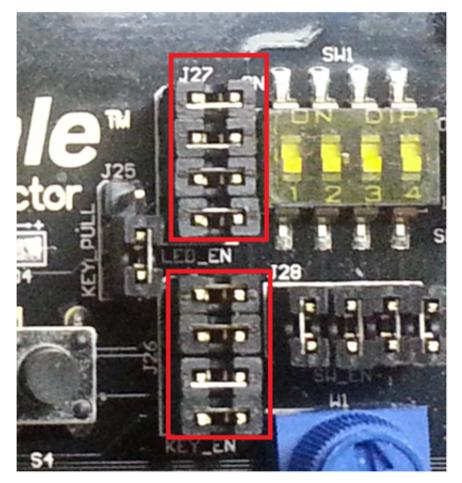


Figure 50. Connect LED and switch jumpers



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.

RAppID BL Tool			
File Help			
Communication Setup:			
Comm Mode Serial Port	Channel: CO	M16 🔻 8	Saud Rate: 9600 🔹
MCU Setup:	BAM Setup:		_
MCU Part No: MPC5605/68X	-	abled •	Default Password
App Setup: Application File: RK-MPC5606	B_Example\FSK_LED_Example\	FLASHVFSK_LED_	Example mot Browse
Auto Read File Sta	rt Address: 0x0	Code Size Ran;	pe: 0x83C8
Operation Setup:			Start Boot Loader
Read MCU Memory	Erase Flash Only	Erase and P	Program Enable Trace
atus: [Serial Port COM16 (9600	Saud)] [MPC5606BK.rba]	[No rbl File Load	ed]

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.



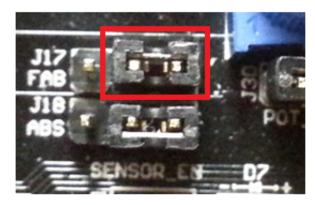


Figure 53. Jumper setting after Flash

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.

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.



iesi code

Options		x
Comm MAP Files	Pack Dir HTML Pages Demo Mode	
Default symbol file	C:\workspace\TRK-MPC5606B_Example\FSK_LED_Example\I	
Ele format:	Binary ELF with DWARF1 or DWARF2 dbg format.	
List of all valid symbol files:	C:\workspace\TRK-MPC56068_Example\FSK_LED_Example\I	v
-,	De	-1
	Note: The file selected in the list will be used as default symbol file	<u>~</u>
	when the project is opened	
	On Load	
	Synchronize variables each time the symbol file loads	
	List errors (variables using undefined symbols) C Stream of Stream interventions	
	 Aways C Except after project load 	
	OK Cancel Apply Help	

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 Varfrom 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.

Project - FreeMASTER	
File Edit View Explorer Item Project Tools Help	
- #	B Z U (9,9) E E E
E New Project Get Marromadia Flash Flager to see FreeMaster Flash presentation here.	Variable Definition Mudiying
Welcome to Free • What's New in Version 1.3? See dearying and the meat impartent thongs selicities wather. The lattert wanten brings 1004. JTAB and CAN connection for various tex • Visit FreeMASTER home page Visit the application home page at membrane Start Project • Continue with this Project	Type: Unsigned faced part style: Descriptions Each data Each data Stre: 2 bytes gask with: To mask (1) Fill let: Fill let: Fill let: gardes Real type transformation Text enumeration (after transform) Text enumeration (after transform) Text enumeration analytic To make (1) Text enumeration
Name Value Unit	OK Cancel (soly Heb
Edit variable <u>Workh Properties</u> <u>Beset MPA/MAX</u> Reget MPA/MAX to All Vars <u>Close To New Watched Var</u> Regiove From Watche	
Create new variable and insert it into the watch	Not connected

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.

Project - FreeMASTER						
Ele Edit View Explorer Jam	Broject Tools Help					
I I I I I I I I I I I I I I I I I I I 	 · · · · · · · · · · · · · · · · · · ·	2011) + +	2 K?	Tahona	• 0 • <u>0 / U</u>	
gil New Project	Please specify the • Show me who If you don't want to with this message The Control Page to and item description	URL of the docume ere can I do it to specify the descr by setting up the statically dis	int describing the l iption document fo single "Control Pag splayed regardless re set up, both tal	r each item in the p	ted in the project tree. project tree, you can hide selection. When both Co	e the "tab"
	Name	Value UN			Period	
Ready	potrationsorvalue 8% potratie 446	080	1000 1000	RIELCOM	16:speed=115200	

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.

Elle Yiew Functions Options Help					
😋 😂 📷 🦃 🧷 🎩 🖫 🖇 👘					
DOGAT Interfaces	Time / 10 mSec Iden	tifier Format	Rags	Data	
USB-to-CAN compact	0.00.09.98	1.9d	Set		000000
CAN 1: SIA 1000	0.00.09.98	2 9td			00 00 00 00
	0.00:12.33	1 Std	Self		00 00 00 00
	0:00:12:33	2 Std		01 01 00 00	
	0:00:14.43	1 Std	Self	12 00 00 00	
	0.00:14.43	2 Std		01 FF 00 00	00 00 00 00 00
	* 🖂				
	Tx	Identifier	Be.	Rr	
Controller initialized		Identifier	E4.		00 00 00 00 00 00 00
Controller initialized O Low speed transceiver	Tx	Identifier	Ext.	12	
-	Tx 是 是	Identifier 1 1	Ex.	12	00 00 00 00 00 00 00
Low speed transceiver	Tx 🗸	Identifier 1 1 1	Est.	12	00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 58. Testing CAN communication

6.4 Testing PWM output

rest code

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.

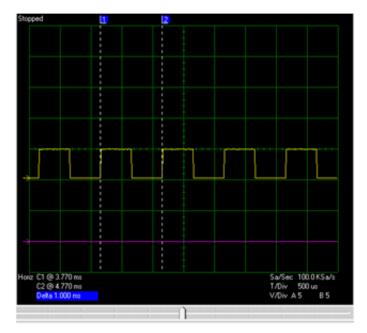


Figure 59. LED4 PWM output





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.