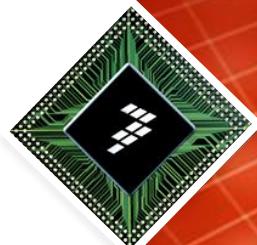




TRK-MPC5606B Fast Start kit Training



October 2013

Freescale, the Freescale logo, AlliVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetic, mobileG1, PEO, PowerQUICC, Processor Expert, QorIQ, Coriwa, SafeAssure, the SafeAssure logo, StarCore, Symphony and VirtIQ are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airstar, BeeKit, BeeStack, CoreNet, Flexie, Layerscape, MagniV, MXC, Platform in a Package, QorIQ Converge, QUICC Engine, Ready Play, SMARTMOS, Tower, TurboLink, Vybrid and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2013 Freescale Semiconductor, Inc.





TRK-MPC5606B Fast Start Kit hardware contents

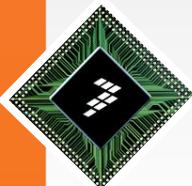
- StarterTRAK TRK-MPC5606B Evaluation Board
- USB cable
- Installation USB stick
- Instruction pamphlet



TRK-MPC5606B Fast Start Kit software contents

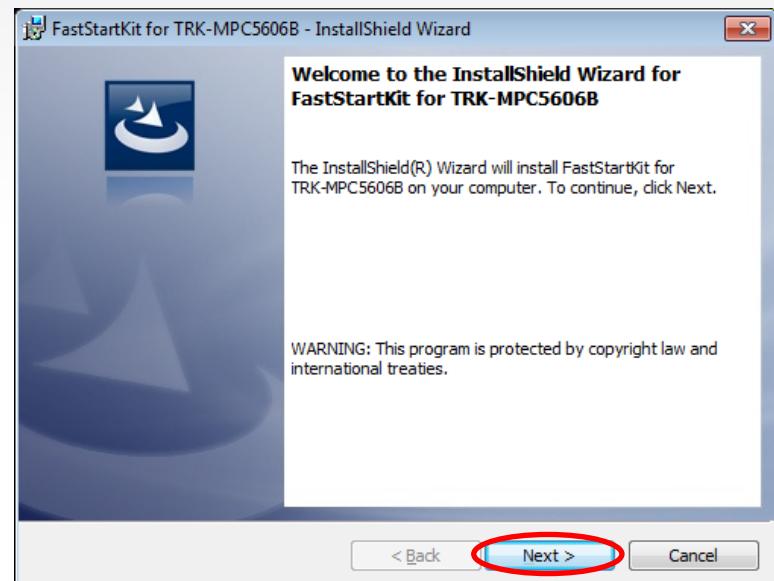
- The TRK-MPC5606B Fast Start Kit contains Single installer that installs all software tools, documents and example projects:
 1. RAppID init and RAppID pin wizard with permanent license
 2. Utility to add RAppID generated code to CodeWarrior 10.5 Project
 3. CodeWarrior for MPC56xx 10.5 SE
 4. Low Level Drivers (LLDs) for MPC5606
 5. High Level Drivers (HLDs) for StarterTRAK TRK-MPC5606B
 6. RAppID Boot loader utility
 7. FreeMASTER utility
 8. Simple LED application examples that demonstrates the usage of software tools, LLDs and HLDs.

Installing Software tools

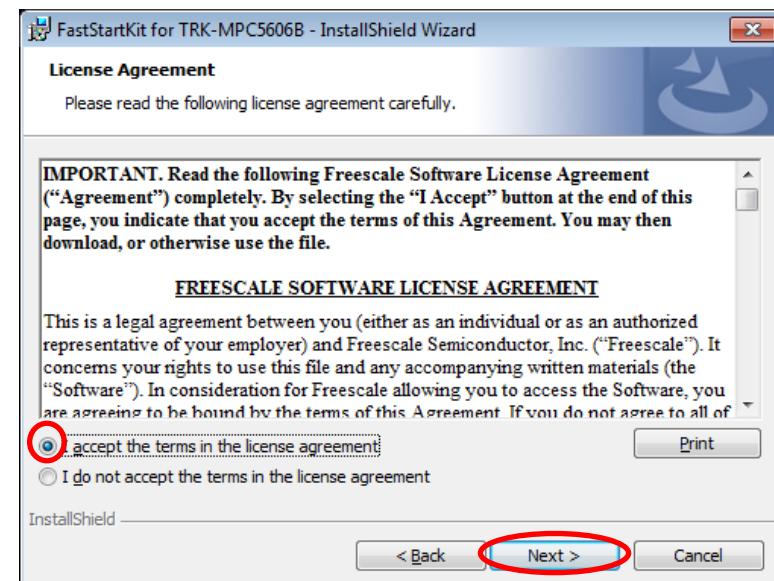
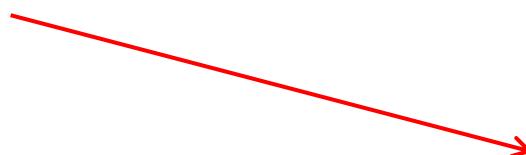


Start installation process by executing **setup.exe** located in the installation media.

Click on *Next*



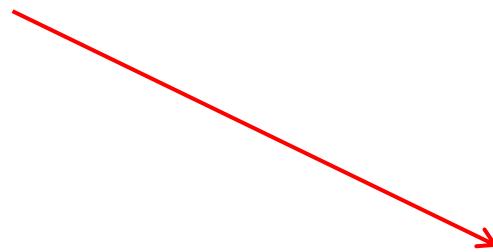
Accept the License agreement and click on *Next*



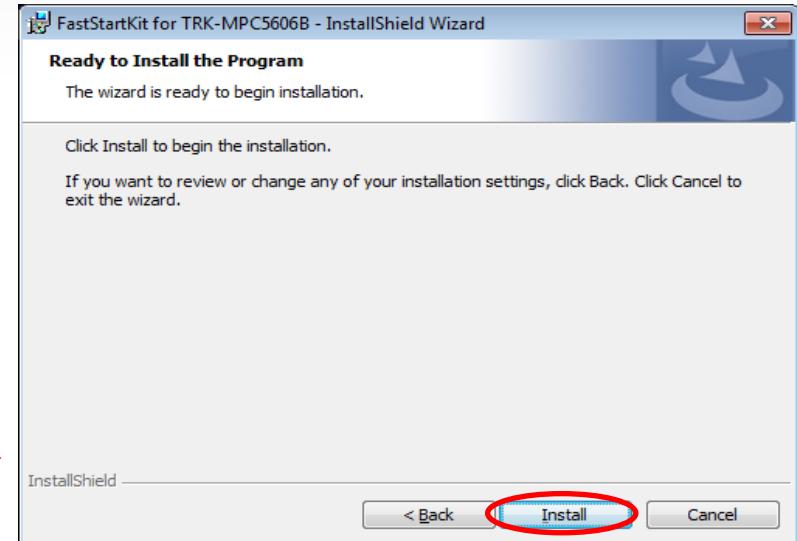
Installing Software tools



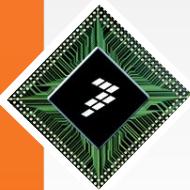
Select *Install* to begin installation of *Fast Start Kit tools*.



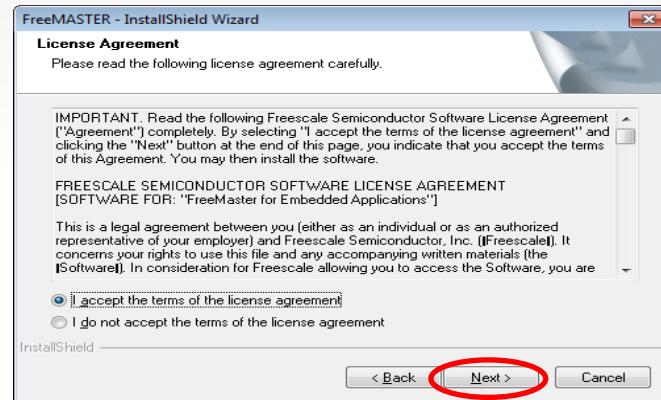
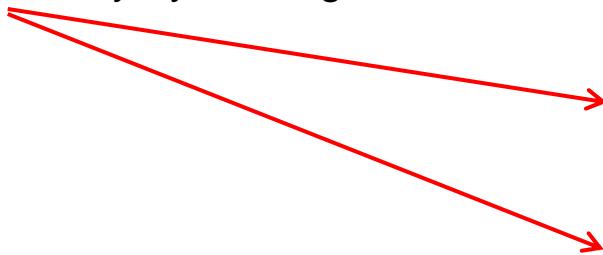
This will launch *FreeMASTER installer*.



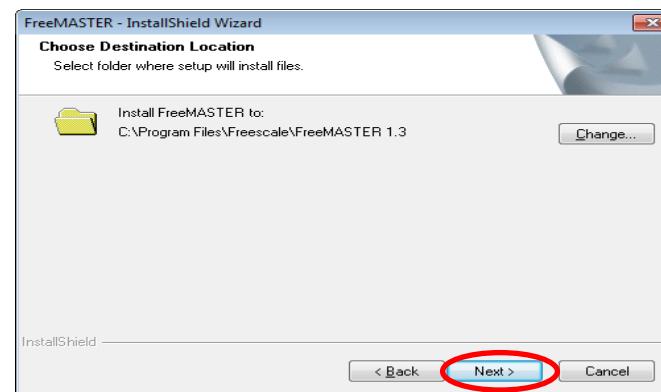
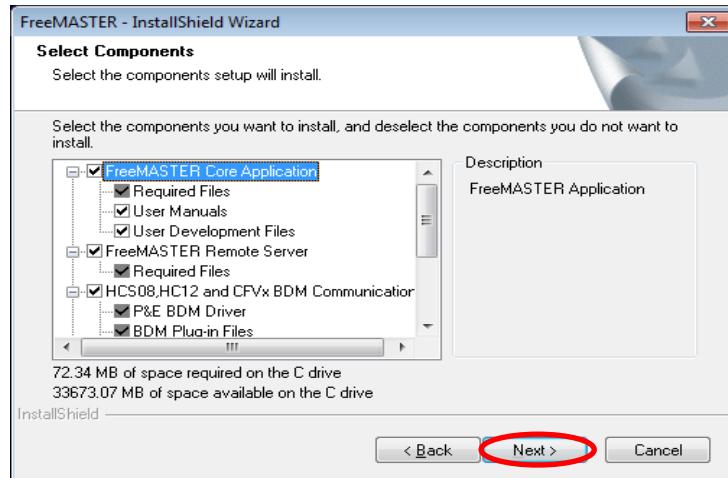
Installing Software tools – FreeMASTER



Accept license agreement and default installation directory by clicking on *Next*.



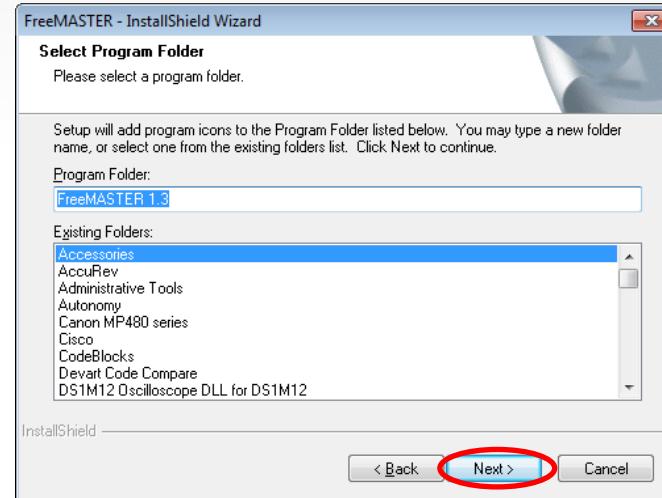
Accept default options in the “Select Components” window by selecting *Next*.



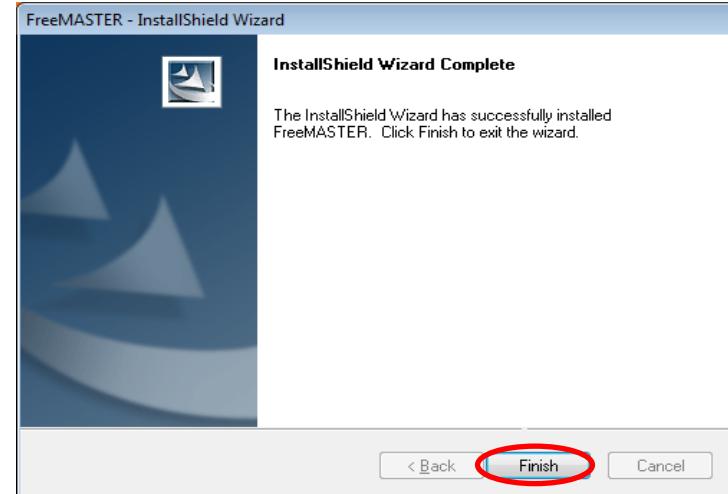
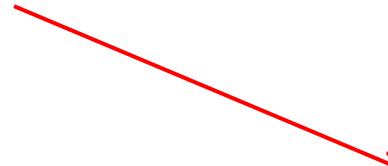
Installing Software tools – FreeMASTER



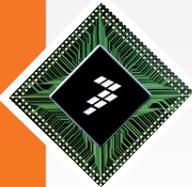
Accept default Program Folder by clicking on *Next*.



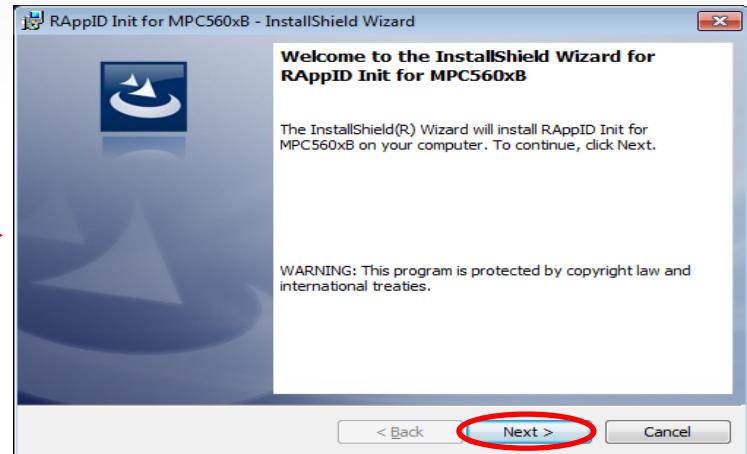
Select *Finish* to complete FreeMASTER installation. This will launch RAppID Init installer.



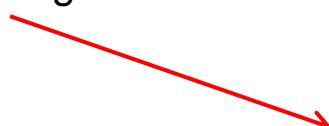
Installing Software tools – RAppID Init



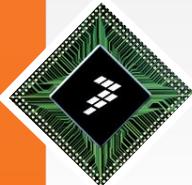
Start RAppID Init installation by clicking on *Next* button



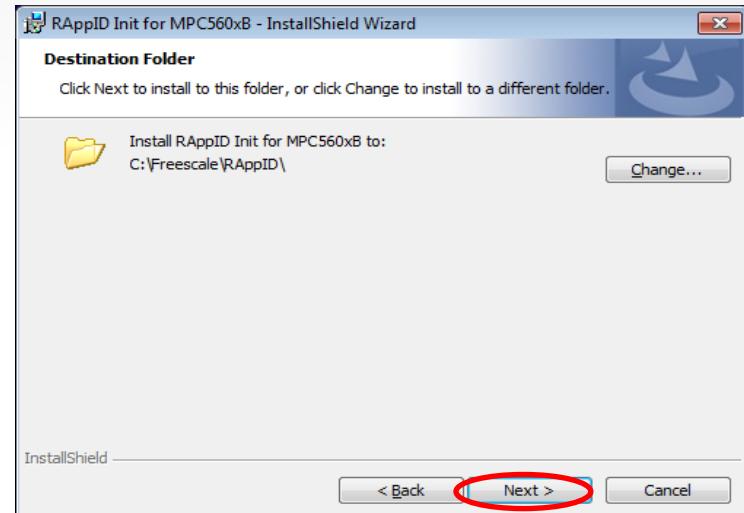
Accept license agreement and default directory location by clicking on *Next*



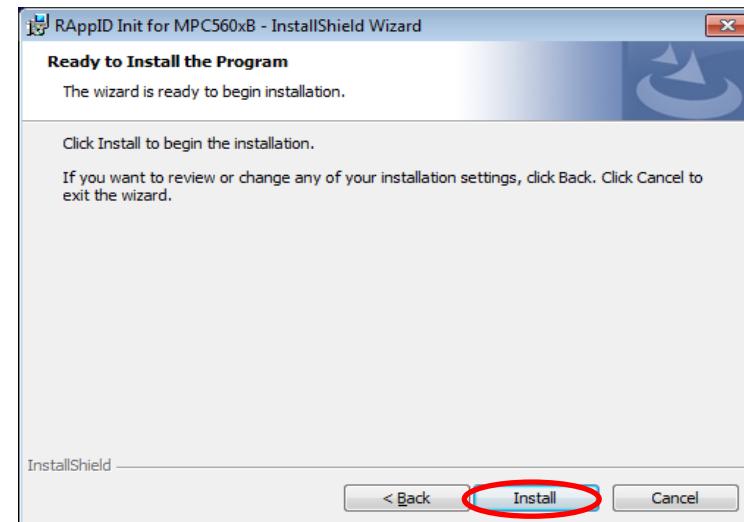
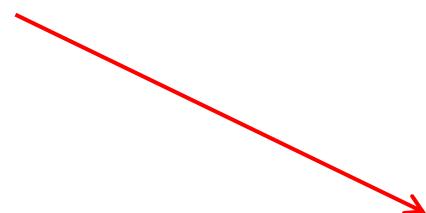
Installing Software tools – RAppID Init



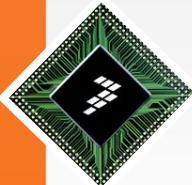
Accept default location for destination folder by clicking on *Next*



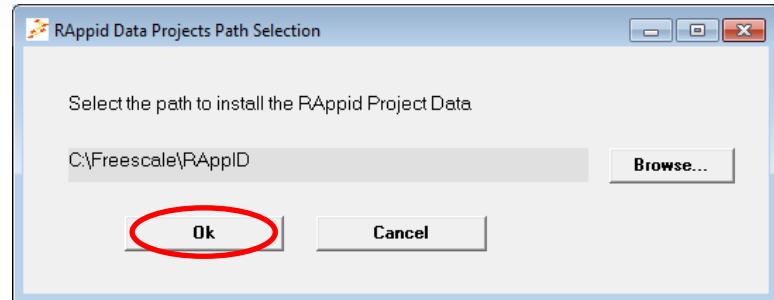
Click On *Install* button



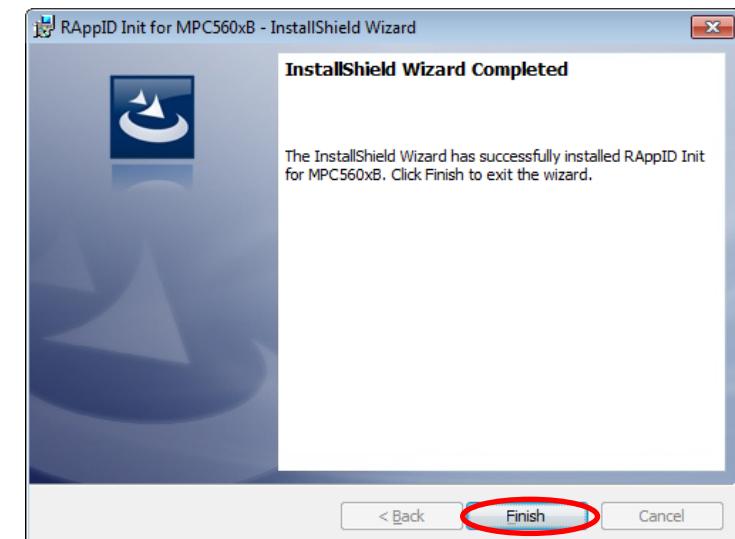
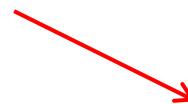
Installing Software tools – RAppID Init



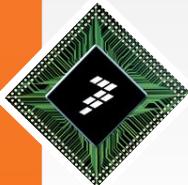
Accept default location for RAppID project data folder by clicking on *Ok*



Click On *Finish* button to complete RAppID Init installation. This will launch RAppID Boot loader installer.



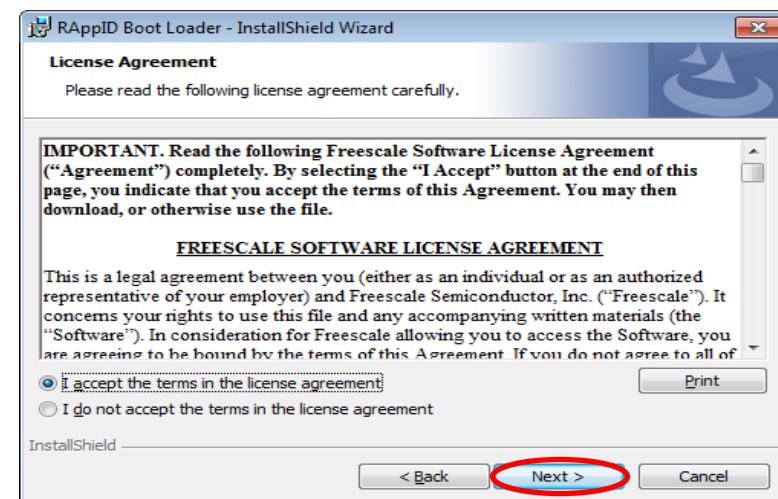
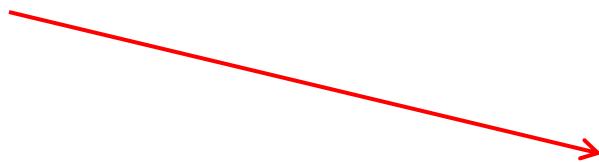
Installing Software tools – RAppID Boot loader



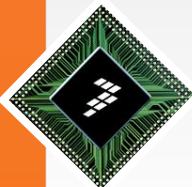
Click on *Next* to start RAppID boot loader installation.



Accept license agreement and click on *Next*



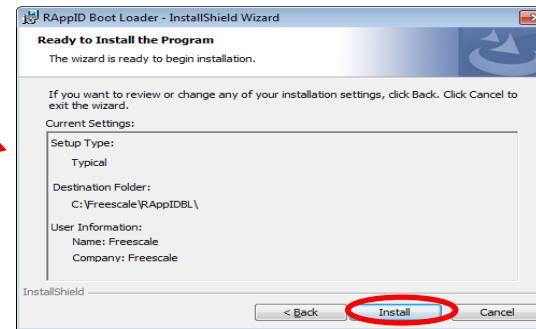
Installing Software tools – RAppID Boot loader



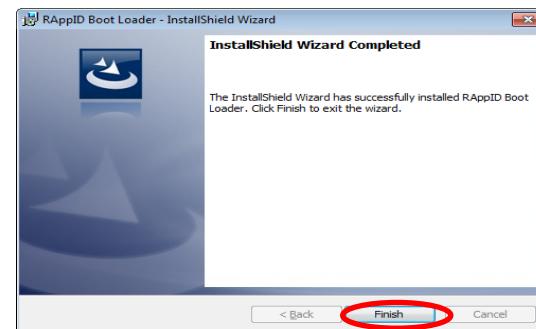
Accept default destination folder by accepting by clicking on *Next*.



Start installation by selecting Install.



After installation is complete, select Finish to complete installation.



This will start CodeWarrior installation.

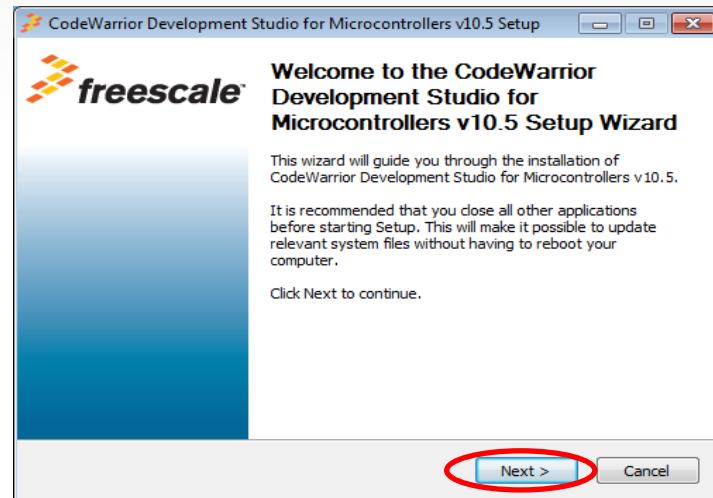
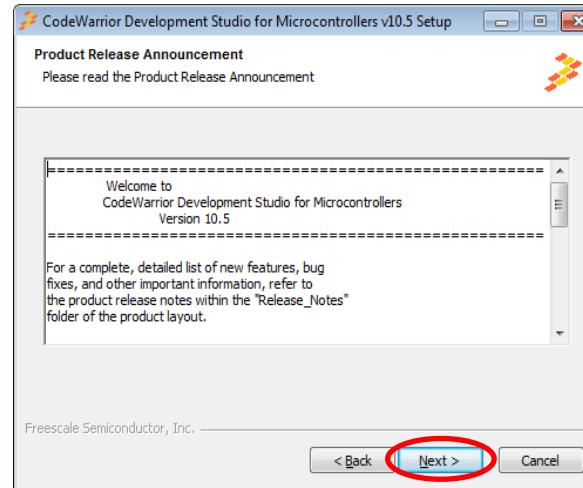
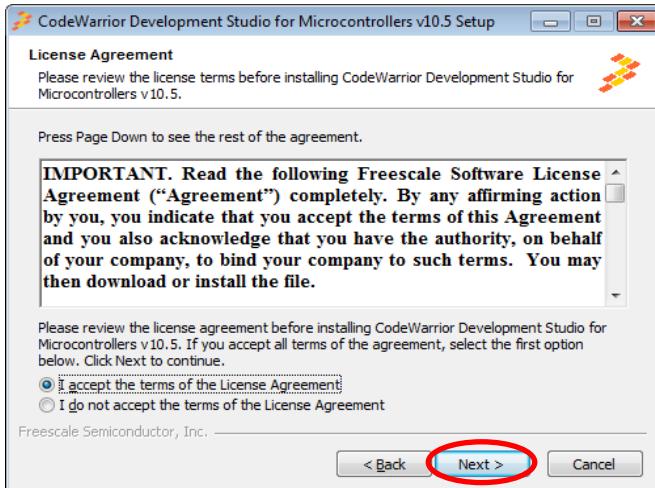
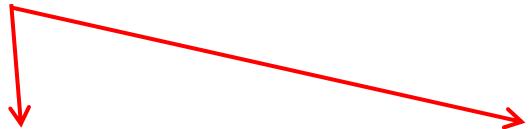
Installing Software tools - CodeWarrior



Start CodeWarrior installation by clicking on **Next**



Accept License agreement and Product Release announcement by clicking on **Next**



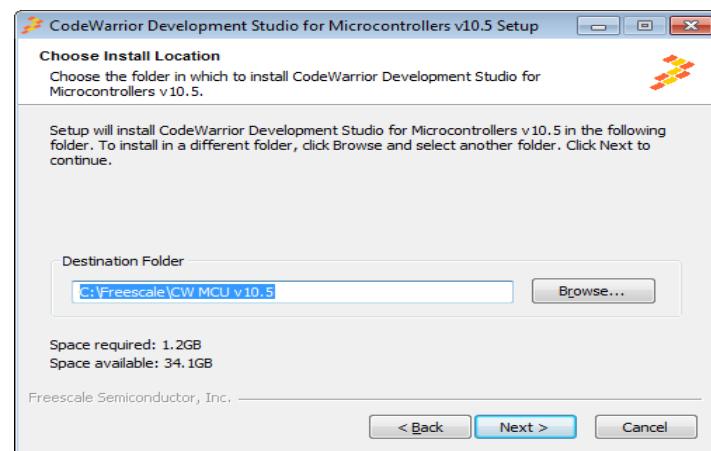
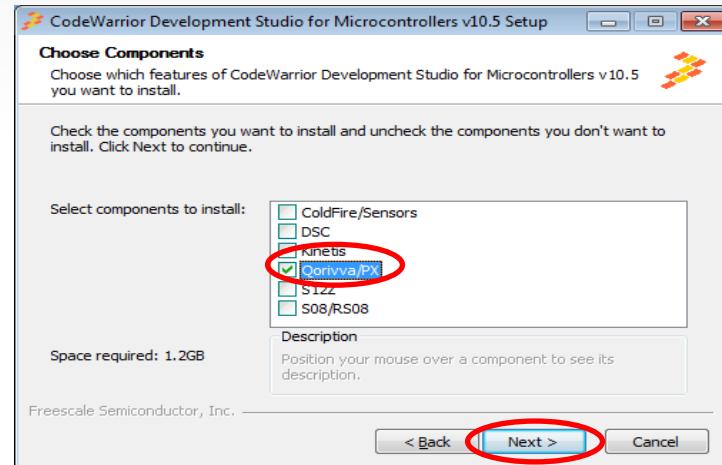
Installing Software tools - CodeWarrior



Choose Qorriva component and click on *Next*



Accept the default install location by selecting *Next*



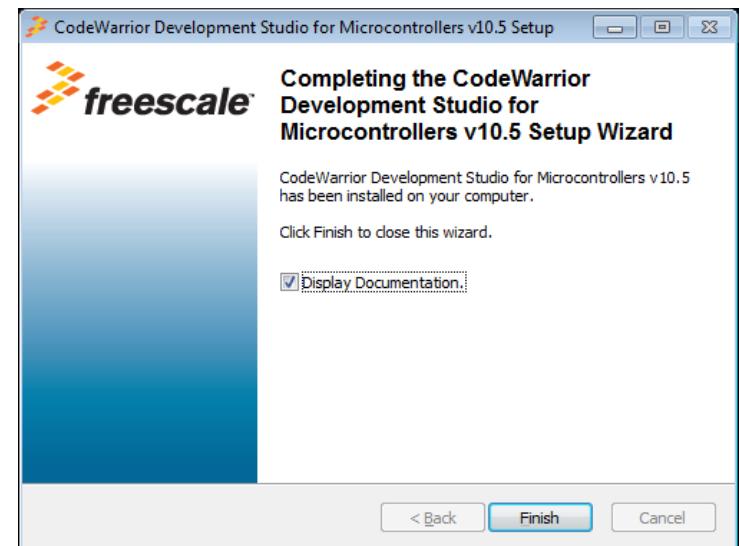
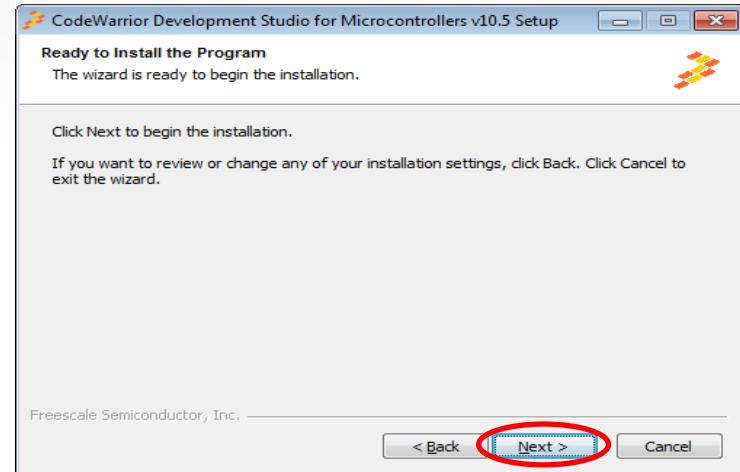
Installing Software tools - CodeWarrior

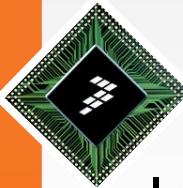


Select Next to begin CodeWarrior installation.



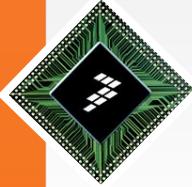
After installation is complete, select Finish to complete installation.





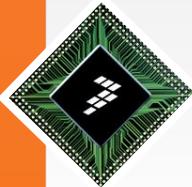
RApID Init Overview

- 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



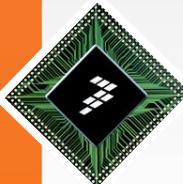
RApID Init Overview

- 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



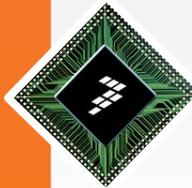
RAppID Generates What?

Driver Utilities	Low Functionality Drivers
Interrupt/Exception Infrastructure	Interrupt Vector Table, Handler, ISR functions
Device Initialization	Device and Peripheral Initialization Code
Main Function – System Init Function	Example Main, Init Sequence Function...
Low Level Setup Code	From Reset to Main, crt0, Stack,...
Reset Vector Code	RCHW, Section Map...



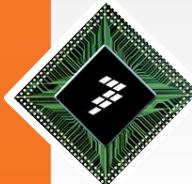
CodeWarrior project maker utility

- The Fast Start Kit for MPC5606B provides a utility to assist in adding RAppID generated code to a CodeWarrior project
- After creating a empty CodeWarrior project for the required microcontroller, the user can invoke the utility `cwpjmaker_0.1.exe` which will add RAppID generated code and sets up the CodeWarrior project by adding all the CodeWarrior setup variables to enable clean build.



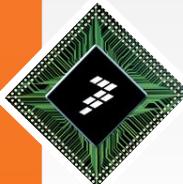
Low Level Driver code

- In the installation disk, the following low level driver code is provided
 - GPIO
 - ADC
 - UART
 - CAN



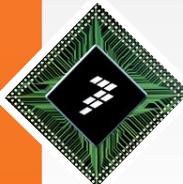
GPIO Low Level Driver code

- **uint8_t GPIO_GetState (uint16_t ch)**
 - This function returns the state of requested GPIO pin
- **void GPIO_SetState (uint16_t ch, uint8_t value)**
 - This function sets the GPIO pin to the specified state



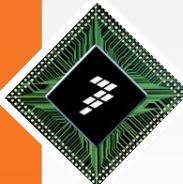
ADC Low Level Driver code

- **uint16_t A2D_GetSingleCh_12bit(uint32_t ch)**
 - This function sets up, starts, and returns a conversion for a single 12-bit ADC1 channel
- **uint16_t A2D_GetSingleCh_10bit (uint32_t ch)**
 - This function Sets up, starts, and returns a conversion for a single 10-bit ADC0 channel
- **uint16_t A2D_GetChResult_12bit (uint32_t ch)**
 - This function returns the result for a single 12-bit ADC1 channel
- **uint16_t A2D_GetChResult_10bit (uint32_t ch)**
 - This function returns the result for a single 10-bit ADC0 channel
- **void A2D_SetupCh_12bit (uint32_t ch)**
 - This function sets up channel for the ADC1 converter
- **void A2D_SetupCh_10bit (uint32_t ch)**
 - This function sets up channel for the ADC0 converter



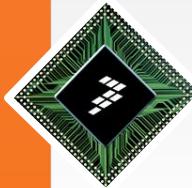
UART Low Level Driver code

- **void UartTxMsg(uint8_t *u8TxData, uint32_t u32Size)**
 - This function transmits a message in buffer *u8TxData* of size *u32Size*
- **uint8_t UartRxDataByte(void)**
 - This function returns data from UART buffer
- **uint32_t UartRxNewDataSize(void)**
 - This function checks how much new data there is in UART buffer
- **uint8_t UartRxBufEmpty(void)**
 - This function checks if the UART buffer is empty
- **void UartBufInit(void)**
 - This function initializes UART Buffer
- **void UartRxFillBuf(void)**
 - This function fills the UART Buffer from the UART RX peripheral



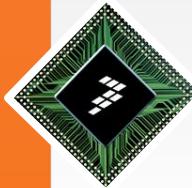
CAN Low Level Driver code

- **void SetCanRxFilter(uint32_t id, uint8_t mb, uint8_t ext)**
 - This function sets up the mailboxes on the specified CAN channel and works for standard and extended IDs.
- **void CanTxMsg (uint32_t id, uint8_t mb, uint8_t dlc, uint8_t data[], uint8_t ext)**
 - This function transmits a CAN message.
- **can_msg_struct CanRxMsg (uint8_t mb)**
 - This function receives a CAN message.
- **uint8_t CanRxMbFull (uint8_t mb)**
 - This function checks if CAN Mail box is full.
- **uint8_t CanTxMbEmpty (uint8_t mb)**
 - This function checks if CAN Mail box is empty.



High Level Driver code

- In the installation disk, the following high level driver code is provided
 - Potentiometer
 - Photo Sensor
 - SBC



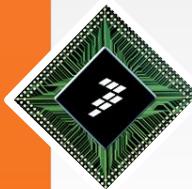
Potentiometer High Level Driver code

- **uint16_t Pot_Get_Value(void)**
 - This function sets up, starts, and returns conversion value for Potentiometer channel (ANP0 of TRK-MPC5606B board).



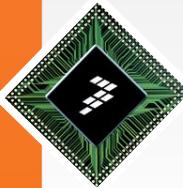
Photo Sensor High Level Driver code

- **uint16_t Photo_Sensor_Get_Value (void)**
 - This function sets up, starts, and returns conversion value for Photo sensor channel (ANP1 of TRK-MPC5606B board).



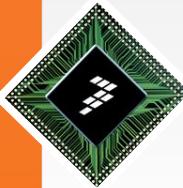
SBC High Level Driver code

- **void SBC_Init_DBG(void)**
 - Sets up SBC in TRK-MPC5606B board to enable CAN. It is assumed that SBC is in Debug mode and watchdog refresh is required only on initialization.



RAppID Bootloader utility

- The RAppID Boot Loader is a tool developed by Freescale to help with the development of software for Freescale MCUs by allowing the customer a method to update software of a MCU through a serial link using CCP.
- The RAppID Boot Loader works with the built in Boot Assist Module (BAM) included in the Freescale Qorivva & PX series 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.).



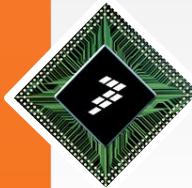
FreeMASTER utility

- FreeMASTER allows users to debug applications in true real-time through its ability to watch and modify variables.
- Remote control capability allows it to be used as a diagnostic tool for debugging customer applications remotely across a network.
- It is an outstanding tool for demonstrating algorithm or application execution and variable outputs.
- It provides monitoring/visualization of application variables in the same manner as a classical oscilloscope with a CRT.
- Simple RS232 native connection and other options possible on selected platforms (BDM, JTAG, CAN,...)
- Built-in support for standard variable types (integer, floating point, bit fields)



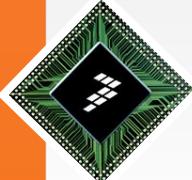
LED Example Using RAppID Init

- The next few slides will demonstrate an example project that describes steps to configure MPC5606B, generate, build, flash and test the code using various tools provided with TRK-MPC5606B Fast Start Kit
- In this example we will use RAppID Init tool to configure and generate initialization code for MPC5606B
- We will use the GPIO, UART, ADC and CAN driver code supplied with the installation
- We will use CodeWarrior 10.5 to build the code
- We will use RAppID Boot loader utility to flash the code to the target
- The example turns on/off LEDs based on Switch input, Potentiometer input, UART and CAN commands.



LED Example overview

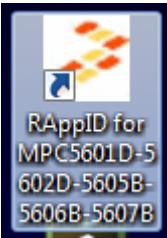
- LED1 turns on when switch S1 is in pressed state and turned off when S1 is in un-pressed state
- LED2 is turned On/Off based on serial command input
- LED3 is turned On/Off based on CAN command input
- LED4 is turned On/Off based on potentiometer input



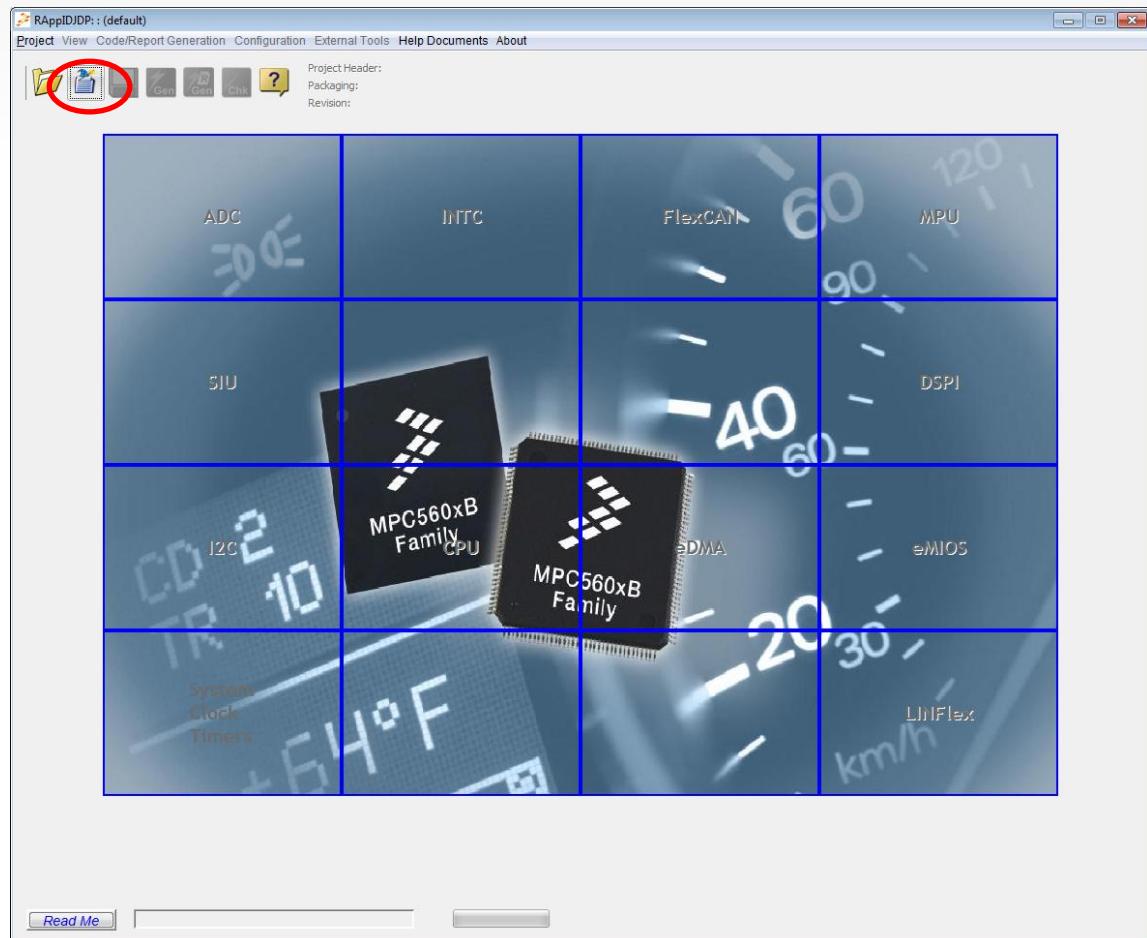
Creating LED example using RAppID init tool

- The next few slides will demonstrate how to create RAppID project to configure all the pins and peripherals required for this example and generate code for CodeWarrior compiler.

Create a New RAppID Project

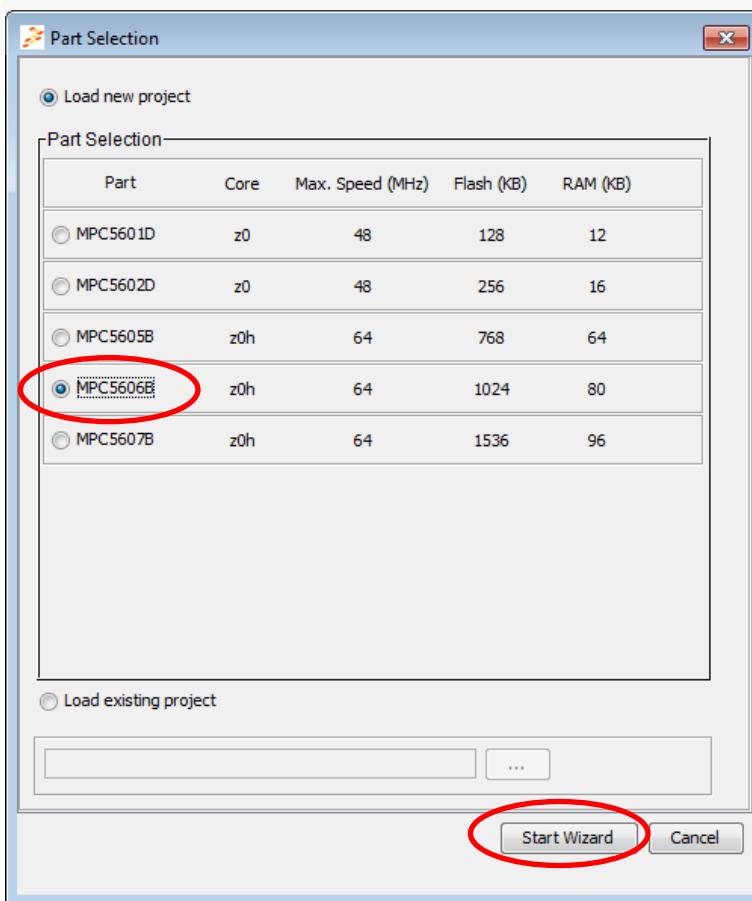


Launch RAppID init by double clicking on RAppID init desktop icon

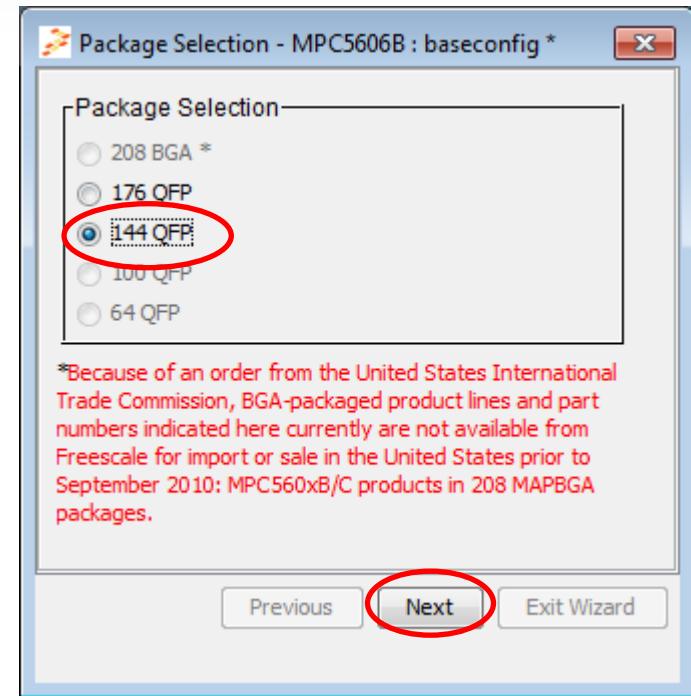


Start a new project by clicking on “New Project Wizard” button

Select Part and Package

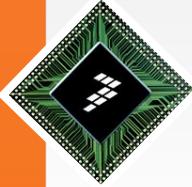


Select a Part MPC5606B and select Start Wizard



Select a Package 144 QFP and select Next

Configure ADC input



Pin Allocation - MPC5606B : baseconfig *

Pin Allocation Wizard

MPC5606B in 144 QFP

PA3	1	PE2	108
PC9	2	PA11	109
PC14	3	PA10	106
PC15	4	PA9	105
PC5	5	PA8	104
PC4	6	PA7	103
RG3	7	PA6	102
PG2	8	PA5	101
PA2	9	PA4	100
PE0	10	PA3	99
PA1	11	PA2	98
PE1	12	PA1	97
PE8	13	PA0	96
PE9	14	TCK	95
PE10	15	TDO	94
PA0	16	TDI	93
PE11	17	TMS	92
VSS_HV	18	TDI	91
VDD_HV	19	TDI	90
VSS_AV	20	TDI	89
NESEB1	21	TDI	88
VSS_AV	22	TDI	87
EGND	23	TDI	86
VDD_AV	24	TDI	85
PG9	25	TDI	84
PG8	26	TDI	83
PC11	27	TDI	82
PC10	28	TDI	81
PG7	29	TDI	80
PG6	30	TDI	79
PG5	31	TDI	78
PG1	32	TDI	77
PF9	33	TDI	76
PF8	34	TDI	75
PF12	35	TDI	74
PG6	36	TDI	73

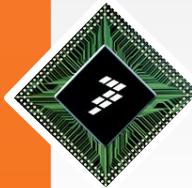
ADC DSPI FlexCAN GPIO I2C LINFlex MISC eMIOS

Functions	Input	Output	User Assigned Signal Name
ADC_0_ADC_1 ANP 0	<input checked="" type="checkbox"/> PB4		ANP0_Pot_Input
ADC_0_ADC_1 ANP 1	<input type="checkbox"/> PB5		
ADC_0_ADC_1 ANP 2	<input type="checkbox"/> PB6		
ADC_0_ADC_1 ANP 3	<input type="checkbox"/> PB7		

Previous Next Exit Wizard View

Potentiometer is connected to ANP0.

In ADC tab, configure PB4 as ADC input and add appropriate name in User Assigned Signal Name edit box.

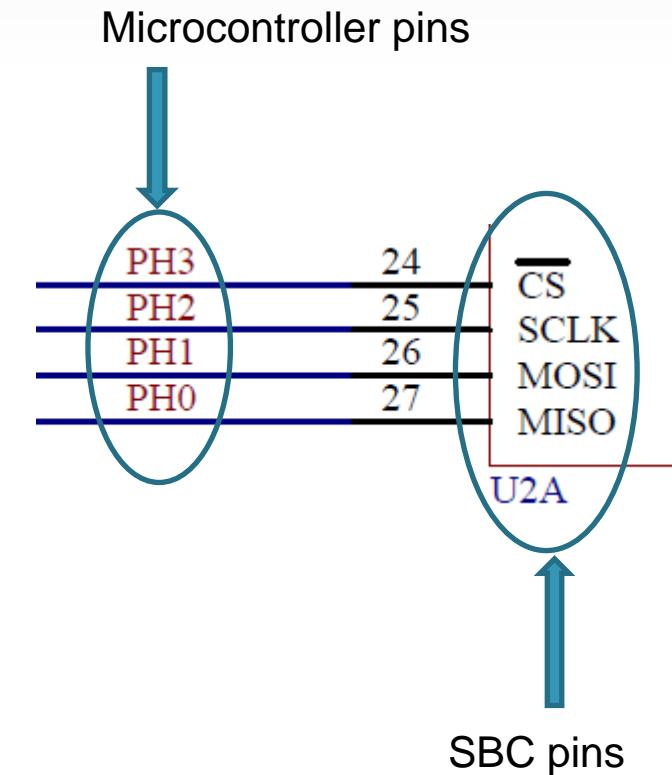


Configure DSPI pins

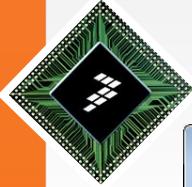
- TRK-MPC5606B contains MCZ3390S5EK system basis chip (SBC) with integrated CAN transceiver and LIN 2.0 interface.
- DSPI 1 is connected to SBC. We need to configure DSPI 1 as master and SBC as slave so that SBC can be configured to enable CAN by sending appropriate commands via DSPI 1.

Configure DSPI pins

- The connection between DSPI 1 of microcontroller and SBC is shown in this picture
- Configure DSPI 1 pins as follows to enable communication with SBC.
 - PH3 is connected to CS pin of SBC. Configure PH3 as DSPI_1 Chip select 0 output
 - PH2 is connected to Clock input pin of SBC. Configure PH2 as DSPI_1 Clock output
 - PH1 is connected to MOSI pin of SBC. Configure PH1 as DSPI_1 Data output
 - PH0 is connected to MISO pin of SBC. Configure PH0 as DSPI_1 Data input



Configure DSPI pins



Pin Allocation - MPC5606B : baseconfig *

Pin Allocation Wizard

MPC5606B in 144 QFP

PG3	1	PE12	108	PA11
PC9	2	PE14	110	PA10
PC14	3	PE15	111	PA9
PC15	4	PE16	113	PA8
PG5	5	PE10	114	PA7
PG4	6	PE21	115	PA6
PG3	7	PC3	116	PA5
PG2	8	PC2	117	PA4
PA2	9	PE6	118	PA3
PE0	10	PA6	119	PA2
PA1	11	TMS	120	PA1
PE1	12	TDO	121	PA0
PE8	13	VSS_HV	122	PG1
PE9	14	VDD_HV	123	PG0
PE10	15	PD10	124	PG15
PA0	16	PD11	125	PG14
PE11	17	PD12	126	PG13
VSS_HV	18	PD13	127	PG12
VDD_HV	19	PD14	128	PG11
VSS_HV	20	PD15	129	PG10
RES3	21	PD16	130	PG9
VSS_HV	22	PD17	131	PG8
VDD_HV	23	PD18	132	PG7
VDD_HV	24	PD19	133	PG6
PG9	25	PD20	134	PG5
PG8	26	PD21	135	PG4
PC11	27	PD22	136	PG3
PC10	28	PD23	137	PG2
P7	29	PD24	138	PG1
PG6	30	PD25	139	PG0
P80	31	PD26	140	PF7
P81	32	PD27	141	PF6
PF9	33	PD28	142	PF5
PF8	34	PD29	143	PF4
PF12	35	PD30	144	PF3
PC6	36	PD31	145	PF2
PC7	37	PD32	146	PF1

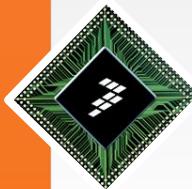
ADC DSPI FlexCAN GPIO I2C LINFlex MISC eMIOS

Functions	Input	Output	User Assigned Signal Name
DSPI_1 Chip Select 1	<input type="checkbox"/> PA6	<input type="checkbox"/> PA6 <input type="checkbox"/> PD14 <input type="checkbox"/> PF6	
DSPI_1 Chip Select 0	<input type="checkbox"/> PA4 <input type="checkbox"/> PC3 <input type="checkbox"/> PE5 <input type="checkbox"/> PD13 <input type="checkbox"/> PH3	<input type="checkbox"/> PA4 <input type="checkbox"/> PC3 <input type="checkbox"/> PE5 <input type="checkbox"/> PD14 <input checked="" type="checkbox"/> PH3	PH3_DSPI1_CS0_OUT
DSPI_1 Clock	<input type="checkbox"/> PE4 <input type="checkbox"/> PC2 <input type="checkbox"/> PH2	<input type="checkbox"/> PE4 <input type="checkbox"/> PC1 <input checked="" type="checkbox"/> PH2	PH2_DSPI1_CLK_OUT
DSPI_1 Data Out		<input type="checkbox"/> PE3 <input type="checkbox"/> PC0 <input checked="" type="checkbox"/> PH1	PH1_DSPI1_DATA_OUT
DSPI_1 Data IN	<input type="checkbox"/> PE2 <input type="checkbox"/> PC4 <input checked="" type="checkbox"/> PH0		PH1_DSPI1_DATA_IN
DSPI_2 Chip Select 3		<input type="checkbox"/> PF5 <input type="checkbox"/> PH7	

Previous Next Exit Wizard View

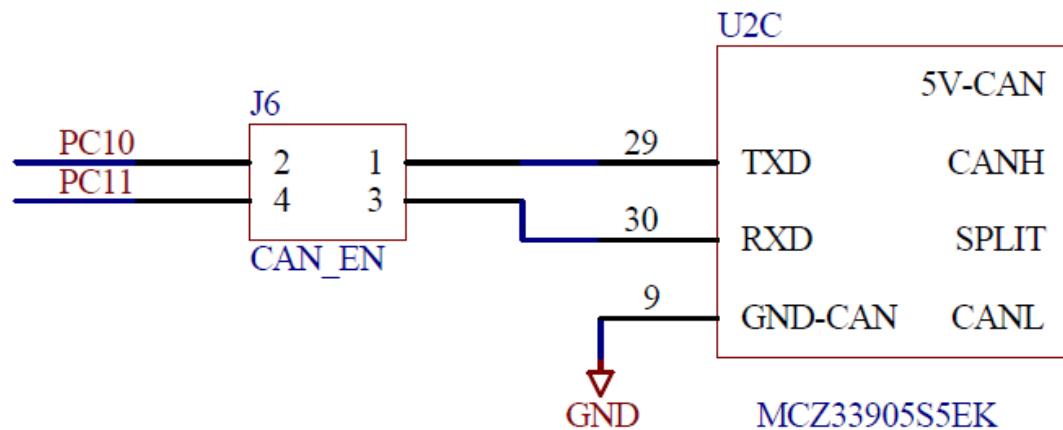
TCK PAD PH9 Allocated as Input
TMS PAD PH10 Allocated as Input
TDI PAD PC0 Allocated as Input
TDO PAD PC1 Allocated as Output
PA8 PAD PA8 Allocated as Input
PA9 PAD PA9 Allocated as Input
ADC_0_ADC_1 ANP 0 PAD PB4 Allocated as Input
DSPI_1 Chip Select 0 PAD PH3 Allocated as Output
DSPI_1 Clock PAD PH2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



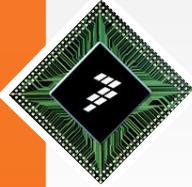


Configure FlexCAN pins

- The CAN TX and CAN RX pins of SBC are connected to the pins PC10 and PC11 of CAN 1 peripheral of the microcontroller.
- We need to configure PC10 as CAN 1 TX pin and PC11 as CAN 1 RX pin.



Configure FlexCAN pins

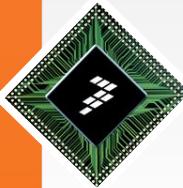


Pin Allocation - MPC5606B : baseconfig *

Pin Allocation Wizard

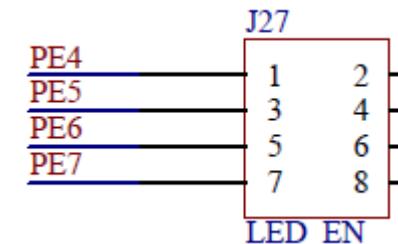
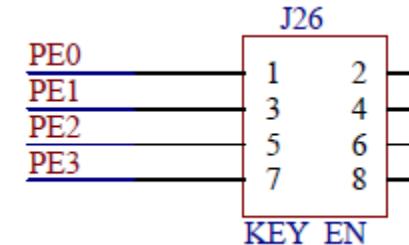
MPC5606B in 144 QFP

Pin Number	Pin Name	Function
1	PB3	VSS_HV
2	PC8	VDD_HV
3	PC14	VSS_HV
4	PG15	VDD_HV
5	PG5	VSS_HV
6	PG4	VDD_HV
7	PG3	VSS_HV
8	PA2	VDD_HV
9	PE0	VSS_HV
10	PA1	VDD_HV
11	PE1	VSS_HV
12	PE8	VDD_HV
13	PE9	VSS_HV
14	PE10	VDD_HV
15	PA0	VSS_HV
16	PE11	VDD_HV
17	PE12	VSS_HV
18	PG9	VDD_HV
19	PG8	VSS_HV
20	RESET	VDD_HV
21	VSS_LV	VSS_LV
22	VDD_LV	VDD_LV
23	VDD_LV	VDD_LV
24	VDD_BV	VDD_BV
25	PG9	VDD_BV
26	PG8	VSS_BV
27	CNTRX_1	CNTRX_1
28	CNTX_1	CNTX_1
29	PG7	CNTRX_1
30	PG6	CNTX_1
31	PB0	CNTRX_1
32	PB1	CNTX_1
33	PF9	CNTRX_1
34	PF8	CNTX_1
35	PF12	CNTRX_1
36	PC5	CNTX_1
37	PE7	VSS_HV
38	PE6	VDD_HV
39	PE5	VSS_HV
40	PE4	VDD_HV
41	PE3	VSS_HV
42	PE2	VDD_HV
43	PE1	VSS_HV
44	PE0	VDD_HV
45	PF12	VSS_HV
46	PF11	VDD_HV
47	PF10	VSS_HV
48	PF9	VDD_HV
49	PF8	VSS_HV
50	PF7	VDD_HV
51	PF6	VSS_HV
52	PF5	VDD_HV
53	PF4	VSS_HV
54	PF3	VDD_HV
55	PF2	VSS_HV
56	PF1	VDD_HV
57	PF0	VSS_HV
58	PE10	VDD_HV
59	PE9	VSS_HV
60	PE8	VDD_HV
61	PE7	VSS_HV
62	PE6	VDD_HV
63	PE5	VSS_HV
64	PE4	VDD_HV
65	PE3	VSS_HV
66	PE2	VDD_HV
67	PE1	VSS_HV
68	PE0	VDD_HV
69	PF6	VSS_HV
70	PF5	VDD_HV
71	PF4	VSS_HV
72	PF3	VDD_HV
73	PF2	VSS_HV
74	PF1	VDD_HV
75	PF0	VSS_HV
76	PE6	VDD_HV
77	PE5	VSS_HV
78	PE4	VDD_HV
79	PE3	VSS_HV
80	PE2	VDD_HV
81	PE1	VSS_HV
82	PE0	VDD_HV
83	PD15	VSS_HV
84	PD14	VDD_HV
85	PD13	VSS_HV
86	PD12	VDD_HV
87	PD11	VSS_HV
88	PD10	VDD_HV
89	PD9	VSS_HV
90	PD8	VDD_HV
91	PD7	VSS_HV
92	PD6	VDD_HV
93	PD5	VSS_HV
94	PD4	VDD_HV
95	PD3	VSS_HV
96	PD2	VDD_HV
97	PD1	VSS_HV
98	PD0	VDD_HV
99	PD9	VSS_HV
100	PD8	VDD_HV
101	PD7	VSS_HV
102	PD6	VDD_HV
103	PD5	VSS_HV
104	PD4	VDD_HV
105	PD3	VSS_HV
106	PD2	VDD_HV
107	PD1	VSS_HV
108	PD0	VDD_HV
109	PA11	VSS_HV
110	PA10	VDD_HV
111	PA9	VSS_HV
112	PA8	VDD_HV
113	PA7	VSS_HV
114	PA6	VDD_HV
115	PA5	VSS_HV
116	PA4	VDD_HV
117	PA3	VSS_HV
118	PA2	VDD_HV
119	PA1	VSS_HV
120	PA0	VDD_HV
121	TDO	VSS_HV
122	TDI	VDD_HV
123	TMS	VSS_HV
124	TDI	VDD_HV
125	TMS	VSS_HV
126	TCK	VDD_HV
127	TDO	VSS_HV
128	TDI	VDD_HV
129	TMS	VSS_HV
130	TCK	VDD_HV
131	TDO	VSS_HV
132	TDI	VDD_HV
133	TMS	VSS_HV
134	TCK	VDD_HV
135	TDO	VSS_HV
136	TDI	VDD_HV
137	TMS	VSS_HV
138	TCK	VDD_HV
139	TDO	VSS_HV
140	TDI	VDD_HV
141	TMS	VSS_HV
142	TCK	VDD_HV
143	TDO	VSS_HV
144	TDI	VDD_HV
145	TMS	VSS_HV
146	TCK	VDD_HV
147	TDO	VSS_HV
148	TDI	VDD_HV
149	TMS	VSS_HV
150	TCK	VDD_HV
151	TDO	VSS_HV
152	TDI	VDD_HV
153	TMS	VSS_HV
154	TCK	VDD_HV
155	TDO	VSS_HV
156	TDI	VDD_HV
157	TMS	VSS_HV
158	TCK	VDD_HV
159	TDO	VSS_HV
160	TDI	VDD_HV
161	TMS	VSS_HV
162	TCK	VDD_HV
163	TDO	VSS_HV
164	TDI	VDD_HV
165	TMS	VSS_HV
166	TCK	VDD_HV
167	TDO	VSS_HV
168	TDI	VDD_HV
169	TMS	VSS_HV
170	TCK	VDD_HV
171	TDO	VSS_HV
172	TDI	VDD_HV
173	TMS	VSS_HV
174	TCK	VDD_HV
175	TDO	VSS_HV
176	TDI	VDD_HV
177	TMS	VSS_HV
178	TCK	VDD_HV
179	TDO	VSS_HV
180	TDI	VDD_HV
181	TMS	VSS_HV
182	TCK	VDD_HV
183	TDO	VSS_HV
184	TDI	VDD_HV
185	TMS	VSS_HV
186	TCK	VDD_HV
187	TDO	VSS_HV
188	TDI	VDD_HV
189	TMS	VSS_HV
190	TCK	VDD_HV
191	TDO	VSS_HV
192	TDI	VDD_HV
193	TMS	VSS_HV
194	TCK	VDD_HV
195	TDO	VSS_HV
196	TDI	VDD_HV
197	TMS	VSS_HV
198	TCK	VDD_HV
199	TDO	VSS_HV
200	TDI	VDD_HV
201	TMS	VSS_HV
202	TCK	VDD_HV
203	TDO	VSS_HV
204	TDI	VDD_HV
205	TMS	VSS_HV
206	TCK	VDD_HV
207	TDO	VSS_HV
208	TDI	VDD_HV
209	TMS	VSS_HV
210	TCK	VDD_HV
211	TDO	VSS_HV
212	TDI	VDD_HV
213	TMS	VSS_HV
214	TCK	VDD_HV
215	TDO	VSS_HV
216	TDI	VDD_HV
217	TMS	VSS_HV
218	TCK	VDD_HV
219	TDO	VSS_HV
220	TDI	VDD_HV
221	TMS	VSS_HV
222	TCK	VDD_HV
223	TDO	VSS_HV
224	TDI	VDD_HV
225	TMS	VSS_HV
226	TCK	VDD_HV
227	TDO	VSS_HV
228	TDI	VDD_HV
229	TMS	VSS_HV
230	TCK	VDD_HV
231	TDO	VSS_HV
232	TDI	VDD_HV
233	TMS	VSS_HV
234	TCK	VDD_HV
235	TDO	VSS_HV
236	TDI	VDD_HV
237	TMS	VSS_HV
238	TCK	VDD_HV
239	TDO	VSS_HV
240	TDI	VDD_HV
241	TMS	VSS_HV
242	TCK	VDD_HV
243	TDO	VSS_HV
244	TDI	VDD_HV
245	TMS	VSS_HV
246	TCK	VDD_HV
247	TDO	VSS_HV
248	TDI	VDD_HV
249	TMS	VSS_HV
250	TCK	VDD_HV
251	TDO	VSS_HV
252	TDI	VDD_HV
253	TMS	VSS_HV
254	TCK	VDD_HV
255	TDO	VSS_HV
256	TDI	VDD_HV
257	TMS	VSS_HV
258	TCK	VDD_HV
259	TDO	VSS_HV
260	TDI	VDD_HV
261	TMS	VSS_HV
262	TCK	VDD_HV
263	TDO	VSS_HV
264	TDI	VDD_HV
265	TMS	VSS_HV
266	TCK	VDD_HV
267	TDO	VSS_HV
268	TDI	VDD_HV
269	TMS	VSS_HV
270	TCK	VDD_HV
271	TDO	VSS_HV
272	TDI	VDD_HV
273	TMS	VSS_HV
274	TCK	VDD_HV
275	TDO	VSS_HV
276	TDI	VDD_HV
277	TMS	VSS_HV
278	TCK	VDD_HV
279	TDO	VSS_HV
280	TDI	VDD_HV
281	TMS	VSS_HV
282	TCK	VDD_HV
283	TDO	VSS_HV
284	TDI	VDD_HV
285	TMS	VSS_HV
286	TCK	VDD_HV
287	TDO	VSS_HV
288	TDI	VDD_HV
289	TMS	VSS_HV
290	TCK	VDD_HV
291	TDO	VSS_HV
292	TDI	VDD_HV
293	TMS	VSS_HV
294	TCK	VDD_HV
295	TDO	VSS_HV
296	TDI	VDD_HV
297	TMS	VSS_HV
298	TCK	VDD_HV
299	TDO	VSS_HV
300	TDI	VDD_HV
301	TMS	VSS_HV
302	TCK	VDD_HV
303	TDO	VSS_HV
304	TDI	VDD_HV
305	TMS	VSS_HV
306	TCK	VDD_HV
307	TDO	VSS_HV
308	TDI	VDD_HV
309	TMS	VSS_HV
310	TCK	VDD_HV
311	TDO	VSS_HV
312	TDI	VDD_HV
313	TMS	VSS_HV
314	TCK	VDD_HV
315	TDO	VSS_HV
316	TDI	VDD_HV
317	TMS	VSS_HV
318	TCK	VDD_HV
319	TDO	VSS_HV
320	TDI	VDD_HV
321	TMS	VSS_HV
322	TCK	VDD_HV
323	TDO	VSS_HV
324	TDI	VDD_HV
325	TMS	VSS_HV
326	TCK	VDD_HV
327	TDO	VSS_HV
328	TDI	VDD_HV
329	TMS	VSS_HV
330	TCK	VDD_HV
331	TDO	VSS_HV
332	TDI	VDD_HV
333	TMS	VSS_HV
334	TCK	VDD_HV
335	TDO	VSS_HV
336	TDI	VDD_HV
337	TMS	VSS_HV
338	TCK	VDD_HV
339	TDO	VSS_HV
340	TDI	VDD_HV
341	TMS	VSS_HV
342	TCK	VDD_HV
343	TDO	VSS_HV
344	TDI	VDD_HV
345	TMS	VSS_HV
346	TCK	VDD_HV
347	TDO	VSS_HV
348	TDI	VDD_HV
349	TMS	VSS_HV
350	TCK	VDD_HV
351	TDO	VSS_HV
352	TDI	VDD_HV
353	TMS	VSS_HV
354	TCK	VDD_HV
355	TDO	VSS_HV
356	TDI	VDD_HV
357	TMS	VSS_HV
358	TCK	VDD_HV
359	TDO	VSS_HV
360	TDI	VDD_HV
361	TMS	VSS_HV
362	TCK	VDD_HV
363	TDO	VSS_HV
364	TDI	VDD_HV
365	TMS	VSS_HV
366	TCK	VDD_HV
367	TDO	VSS_HV
368	TDI	VDD_HV
369	TMS	VSS_HV
370	TCK	VDD_HV
371	TDO	VSS_HV
372	TDI	VDD_HV
373	TMS	VSS_HV
374	TCK	VDD_HV
375	TDO	VSS_HV
376	TDI	VDD_HV
377	TMS	VSS_HV
378	TCK	VDD_HV
379	TDO	VSS_HV
380	TDI	VDD_HV
381	TMS	VSS_HV
382	TCK	VDD_HV
383	TDO	VSS_HV
384	TDI	VDD_HV
385	TMS	VSS_HV
386	TCK	VDD_HV
387	TDO	VSS_HV
388	TDI	VDD_HV
389	TMS	VSS_HV
390	TCK	VDD_HV
391	TDO	VSS_HV
392	TDI	VDD_HV
393	TMS	VSS_HV
394	TCK	VDD_HV
395	TDO	VSS_HV
396	TDI	VDD_HV
397	TMS	VSS_HV
398	TCK	VDD_HV
399	TDO	VSS_HV
400	TDI	VDD_HV
401	TMS	VSS_HV
402	TCK	VDD_HV
403	TDO	VSS_HV
404	TDI	VDD_HV
405	TMS	VSS_HV
406	TCK	VDD_HV
407	TDO	VSS_HV
408	TDI	VDD_HV
409	TMS	VSS_HV
410	TCK	VDD_HV
411	TDO	VSS_HV
412	TDI	VDD_HV
413	TMS	VSS_HV
414	TCK	VDD_HV
415	TDO	VSS_HV
416	TDI	VDD_HV
417	TMS	VSS_HV
418	TCK	VDD_HV
419	TDO	VSS_HV
420	TDI	VDD_HV
421	TMS	VSS_HV
422	TCK	VDD_HV
423	TDO	VSS_HV
424	TDI	VDD_HV
425	TMS	VSS_HV
426	TCK	VDD_HV
427	TDO	VSS_HV
428	TDI	VDD_HV
429	TMS	VSS_HV
430	TCK	VDD_HV
431	TDO	VSS_HV
432	TDI	VDD_HV
433	TMS	VSS_HV
434	TCK	VDD_HV
435	TDO	VSS_HV
436	TDI	VDD_HV
437	TMS	VSS_HV
438	TCK	VDD_HV
439	TDO	VSS_HV
440	TDI	VDD_HV
441	TMS	VSS_HV
442	TCK	VDD_HV
443	TDO	VSS_HV
444	TDI	VDD_HV
445	TMS	VSS_HV
446	TCK	VDD_HV
447	TDO	VSS_HV
448	TDI	VDD_HV
449	TMS	VSS_HV
450	TCK	VDD_HV
451	TDO	VSS_HV
452	TDI	VDD_HV
453	TMS	VSS_HV
454	TCK	VDD_HV
455	TDO	VSS_HV
456	TDI	VDD_HV
457	TMS	VSS_HV
458	TCK	VDD_HV



Configure GPIO pins

- In this example, we are using Switch S1 as input and the 4 LEDs as outputs.
- Switch S1 is connected to PE0. Configure PE0 as S1 input
- LED1 is connected to PE4. Configure PE4 as output
- LED2 is connected to PE5. Configure PE5 as output
- LED3 is connected to PE6. Configure PE6 as output
- LED4 is connected to PE7. Configure PE7 as output



Configure GPIO pins

Pin Allocation - MPC5606B : baseconfig *

MPC5606B in 144 QFP

PB3	1	PA11	108
PC9	2	PA10	107
PC14	3	PA9	106
PC15	4	PA8	105
PG5	5	PA7	104
PG4	6	PE13	103
PG3	7	PF14	102
PG2	8	PF15	101
PA2	9	VDD_3V3	100
PE01	10	VSS_3V3	99
PA1	11	PG0	98
PE1	12	PG1	97
PE8	13	PCS_10_0	96
PE9	14	SCK_1_0	95
PE10	15	SOUT_1	94
PA0	16	SIN_1	93
PE11	17	PG12	92
VSS_HV	18	PG13	91
VDD_HV	19	PA3	90
VSS_HV	20	PG15	89
RESET	21	PG16	88
VSS_LV	22	PG14	87
VDD_LV	23	PG13	86
VDD_BV	24	PG12	85
PG9	25	PG11	84
PG8	26	PG10	83
CNRX_1	27	VDD_HV	82
CNTX_1	28	VSS_HV	81
PG7	29	PD11	80
PG6	30	PD10	79
PG0	31	PD9	78
PB1	32	PD8	77
PF9	33	PD7	76
PF8	34	PD6	75
PF12	35	PD5	74
PC8	36	PD4	73
PE7	37	PD3	72
PC7	38	PD2	71
PE6	39	PD1	70
PC12	40	PD0	69
PC12	41	PG5	68
PC15	42	PG4	67
PC13	43	PG3	66
PC13	44	PG2	65
PE5	45	PG1	64
PE5	46	PG0	63
PE4	47	PF8	62
PE4	48	PF7	61
PE3	49	PF6	60
PE3	50	PF5	59
PE2	51	PF4	58
PE2	52	PF3	57
PE1	53	PF2	56
PE1	54	PF1	55
PE0	55	PF0	54
PE0	56	PE10	53
PE0	57	PE9	52
PE0	58	PE8	51
PE0	59	PE7	50
PE0	60	PE6	49
PE0	61	PE5	48
PE0	62	PE4	47
PE0	63	PE3	46
PE0	64	PE2	45
PE0	65	PE1	44
PE0	66	PE0	43
PE0	67	PA13	42
PE0	68	PA14	41
PE0	69	PA15	40
PE0	70	PA11	39
PE0	71	PA10	38

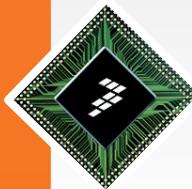
Functions **Input** **Output** **User Assigned Signal Name**

Functions	Input	Output	User Assigned Signal Name
PE0	<input checked="" type="checkbox"/> PE0	PE0	PE0_S1_Input
PE1	<input type="checkbox"/> PE1	PE1	
PE2	<input type="checkbox"/> PE2	PE2	
PE3	<input type="checkbox"/> PE3	PE3	
PE4	<input type="checkbox"/> PE4	PE4	PE4_LED1_Output
PE5	<input type="checkbox"/> PE5	PE5	PE5_LED1_Output
PE6	<input type="checkbox"/> PE6	PE6	PE6_LED1_Output
PE7	<input type="checkbox"/> PE7	PE7	PE7_LED1_Output

Previous **Next** **Exit Wizard** **View**

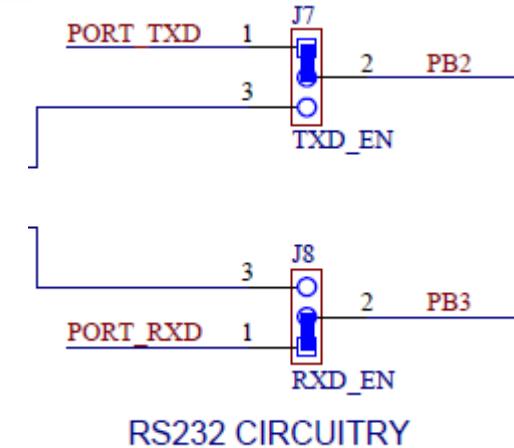
TCK PAD PH9 Allocated as Input
TMS PAD PH10 Allocated as Input
TDI PAD PC0 Allocated as Input
TDO PAD PC1 Allocated as Output
PA8 PAD PA8 Allocated as Input
PA9 PAD PA9 Allocated as Input
ADC_0_ADC_1 ANP_0 PAD PB4 Allocated as Input
DSPI_1 Chip Select 0 PAD PH3 Allocated as Output
DSPI_1 Clock PAD PH2 Allocated as Output
DSPI_1 Data Out PAD PH1 Allocated as Output
PE0 PAD PE0 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



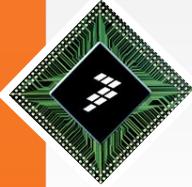


Configure LINFlex (UART) pins

- In this example, we will use Virtual serial port of TRK-MPC5606B board for serial communication.
- The PB2 and PB3 of microcontroller in TRK-MPC5606B board are connected to TX and RX pins of virtual serial port.
- We need to configure PB2 as LINFlex0 TX pin and PB3 as LINFlex0 RX pin.



Configure LINFlex (UART) pins



Pin Allocation - MPC5606B : baseconfig *

MP5606B Pin Allocation Wizard

Pin Number	Pin Name	Pad Name	Description
1	RXD_0	PA11	TCK PAD PH9 Allocated as Input
2	PC9	PA10	TMS PAD PH10 Allocated as Input
3	PC14	PA9	TDI PAD PC0 Allocated as Input
4	PC15	PA8	TDO PAD PC1 Allocated as Output
5	PG5	PA7	PA8 PAD PA8 Allocated as Input
6	PG4	PE13	PA9 PAD PA9 Allocated as Input
7	PG3	PE14	ADC_0_ADC_1 ANP 0 PAD PB4 Allocated as Input
8	PG2	PF15	DSPI_1 Chip Select 0 PAD PH3 Allocated as Output
9	PA2	PF16	DSPI_1 Clock PAD PH2 Allocated as Output
10	PE01	VDD_HV	DSPI_1 Data Out PAD PH1 Allocated as Output
11	PA1	VSS_HV	DSPI_1 Data In PAD PH0 Allocated as Input
12	PE1	PG1	CAN_1 Tx PAD PC10 Allocated as Output
13	PE8	PC10_0	CAN_1 Rx PAD PC11 Allocated as Input
14	PE9	SCK_I_O	PE0 PAD PE0 Allocated as Input
15	PE10	SOUT_1	PE4 PAD PE4 Allocated as Output
16	PA0	SIN_1	PE5 PAD PE5 Allocated as Output
17	PE11	PG12	PE6 PAD PE6 Allocated as Output
18	VSS_HV	PG13	PE7 PAD PE7 Allocated as Output
19	VDD_HV	PA3	LINFlex_0 Tx PAD PB2 Allocated as Output
20	VSS_HV	PB15	LINFlex_0 Rx PAD PB3 Allocated as Input
21	RESET	PD15	
22	VSS_HV	PD10	
23	VDD_HV	PD9	
24	VDD_HV	PD14	
25	PG9	PD13	
26	PG8	PD11	
27	CNRX_1	PD11	
28	CNTX_1	PD10	
29	PG7	PD9	
30	PG6	PD8	
31	PB0	PD7	
32	PB1	PD6	
33	PF9	PD5	
34	PF8	PD4	
35	PF12	PD3	
36	PC6	PD2	
37	PC7	PD1	
38	PF10	PD0	
39	PF11	AO0_0	
40	PA15		
41	PF13		
42	PA14		
43	PA4		
44	PA13		
45	PA12		
46	VDD_HV		
47	VSS_HV		
48	VDD_HV		
49	VSS_HV		
50	EXM0		
51	SDIO_JN		
52	PF8		
53	PF7		
54	PF10		
55	PF9		
56	PF11		
57	PF2		
58	PF3		
59	PF4		
60	PF5		
61	PF6		
62	PF7		
63	PD0		
64	PF1		
65	PF2		
66	PF3		
67	PF4		
68	PF5		
69	PF6		
70	PF7		
71	PF8		
72	PF9		
73	PF10		
74	PCO_0		
75	PCO_1		
76	PCO_2		
77	PCO_3		
78	PCO_4		
79	PCO_5		
80	PCO_6		
81	VDD_HV		
82	VDD_HV		
83	VSS_HV		
84	VSS_HV		
85	VDD_HV		
86	VDD_HV		
87	VB14		
88	PD15		
89	PD14		
90	PD13		
91	PD12		
92	PD11		
93	PD10		
94	PD9		
95	PD8		
96	PD7		
97	PD6		
98	PG0		
99	VSS_HV		
100	VDD_HV		
101	PF15		
102	PF14		
103	PF13		
104	PF12		
105	PF11		
106	PF10		
107	PF9		
108	PF8		

MP5606B in 144 QFP

Tab Selection: ADC, DSPI, FlexCAN, GPIO, I²C, **LINFlex**, I^MSC, eMIOS

Functions

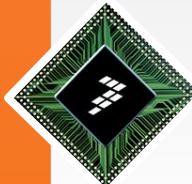
	Input	Output	User Assigned Signal Name
LINFlex_0 Tx	<input type="checkbox"/> PB0 <input checked="" type="checkbox"/> PB2	<input type="checkbox"/>	PB2_UART_Tx
LINFlex_0 Rx	<input type="checkbox"/> PB1 <input checked="" type="checkbox"/> PB3	<input type="checkbox"/>	PB3_UART_Rx
LINFlex_1 Tx	<input type="checkbox"/>	<input type="checkbox"/> PC6	
LINFlex_1 Rx	<input type="checkbox"/> PC7	<input type="checkbox"/>	

Buttons: Previous, Next, Exit Wizard, View

In LINFlex tab, configure PB2 as LINFlex_0 Tx and PB3 as LINFlex_0 Rx pins and add user signal names.

Select Next and skip next two windows by selecting Next





Configure Mode Entry

- RAppID tool generates code to set the microcontroller in DRUN mode at startup.
- In this example, we will use 16MHz Fast Internal Reference Clock (FIRC) as system clock source. Select 16MHz clock to be enabled in DRUN mode and enable XOSC in DRUN mode which is required as clock source for FlexCAN peripheral.

Configure Mode Entry



SystemClockTimers - MPC5606B : baseconfig *

Mode Entry | System Clock | Low Power Clocks | CMU | PCU | SWT | PIT | RTC | STM | Software Attribute |

General Configuration | Peripheral Configuration | Interrupt |

Mode Enable

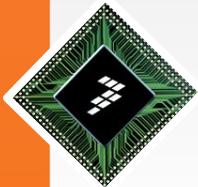
RESET Mode	EN	TEST Mode	DI
SAFE Mode	EN	DRUN Mode	EN
RUN0 Mode	EN	RUN1 Mode	DI
RUN2 Mode	DI	RUN3 Mode	DI
HALT0 Mode	DI	STOP0 Mode	DI
STANDBY0 Mode	DI		

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	DI	EN	EN	16MHz internal RC oscillator
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

Getting Started Ok Cancel Apply

In General Configuration tab, enable XOSC in DRUN mode. This is required to power FlexCAN



Configure Mode Entry

Configure the peripherals to be enabled during different operational modes.

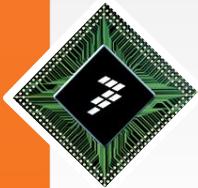
Select “Normal” configuration

This will enable the “peripheral run configuration” and “low power configuration” across different operational modes

Then it will assign these two configurations across all peripherals

The screenshot shows the "System Clock/Timers - MPC5606B : baseconfig*" configuration wizard with the "Mode Entry" tab selected. The "Peripheral Configuration" sub-tab is active. A red circle highlights the "Normal" button in the mode selection row. Below this, the "Run Peripheral Configuration" section displays a grid of peripheral run configurations for various peripheral configurations (Config0 to Config7). The "Low Power Mode Configuration" section shows a grid for low power modes (HALTO, STOP, STANDBY) across the same peripheral configurations. The "Peripheral Mode Configuration - Normal" section at the bottom lists peripherals (DSPI_0 to FlexCAN_1) and their assigned low-power and run-mode configurations. Two blue arrows point from the text above to the "Low Power Mode Configuration" and "Run Mode Configuration" columns in this table.

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



Disable Watchdog

By default, watchdog is enabled in MPC5606B. In this example, we will not use Watchdog feature.
Disable Watchdog Timer in SWT tab .

This completes basic system configuration. Select Next to begin Peripheral configuration.

System Clock/Timers - MPC5606B : baseconfig *

Mode Entry | System Clock | Low Power Clocks | CMU | PCC | **SWT** | RTC | STM | Software Attribute

Watchdog Control

Enable Watchdog Timer

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

Watchdog Timeout

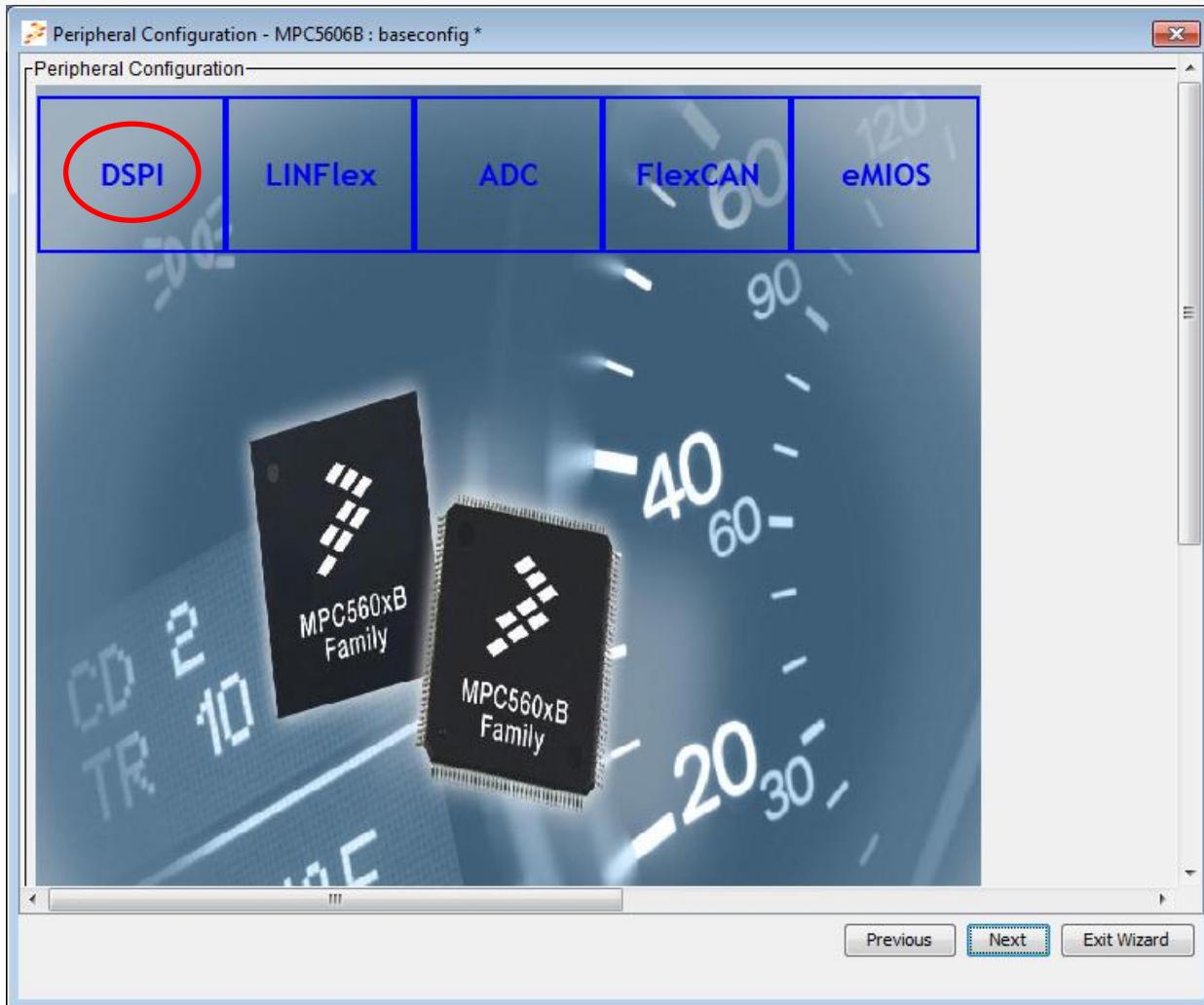
Clock Source Frequency: 128 KHz
Watchdog Timeout Range: 10 ms to 33554.43 sec
Watchdog Resolution: 0.008 ms
Watchdog Timeout in Clock Cycles: 1280
Watchdog Timeout: 10 ms

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 %

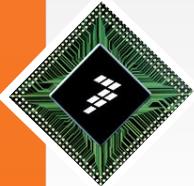
Previous | Next | Exit Wizard

Configure Peripherals



In Peripheral Configuration window, select DSPI to configure DSPI peripheral

Configure DSPI 1

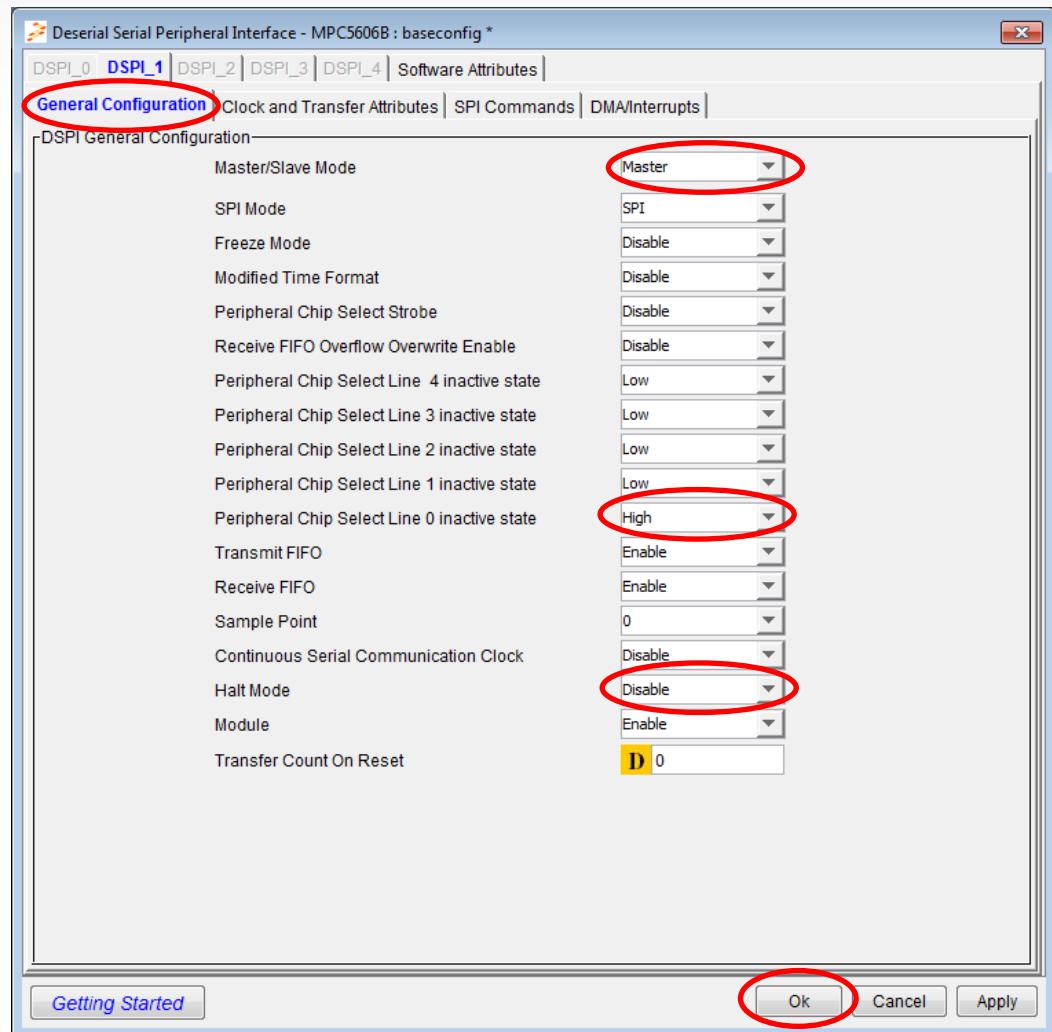


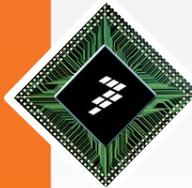
DSPI 1 should be set to master mode to send commands to SBC. Select Master mode

DSPI 1 Chip Select 0 is connected to SBC. Set Chip select 0 inactive state to High

Disable Halt mode

Select OK to finish DSPI 1 configuration





Configure LINFlex 0

- We will use UART of LINFlex 0 to communicate serially via virtual serial port of TRK-MPC5606B board
- We will use baud rate of 115,200
- When Baud Rate Factor and Fractional Baud Rate Factor values are selected, RAppID automatically calculates and displays the resulting baud rate.

Configure LINFlex 0



Set Baud Rate Factor to 8

Set Fractional Baud Rate Factor to 11/16

This should set the Baud rate to approx. 115,200

Screenshot of the Local Interconnect Network - MPC5606B : baseconfig * configuration window for LINFlex_0.

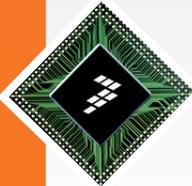
The window shows the following configuration settings:

Setting	Value
Master Mode	DI
Receiver Buffer Locked Mode	DI
LIN Master Break Length	10 bits
Bypass Filter	EN
Auto Wakeup	DI
Checksum Field	EN
Programmed Checksum	Enabled (multiple checkboxes checked)
Integer Baud Rate Factor	8 (highlighted with a red circle)
Baudrate(Symbols/Sec) =	115107.91 (highlighted with a red circle)
Loopback Mode	DI
Self Test Mode	DI
Slave Mode Break Detection Threshold	11 bits
LIN Auto Sync	DI
Idle on Bit Error	EN
Checksum Calculation	DI
Idle on Identifier Parity Error	EN
Fractional Baud Rate Factor	11/16 (highlighted with a red circle)

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

Buttons at the bottom: Getting Started, Ok, Cancel, Apply.

Configure LINFlex 0

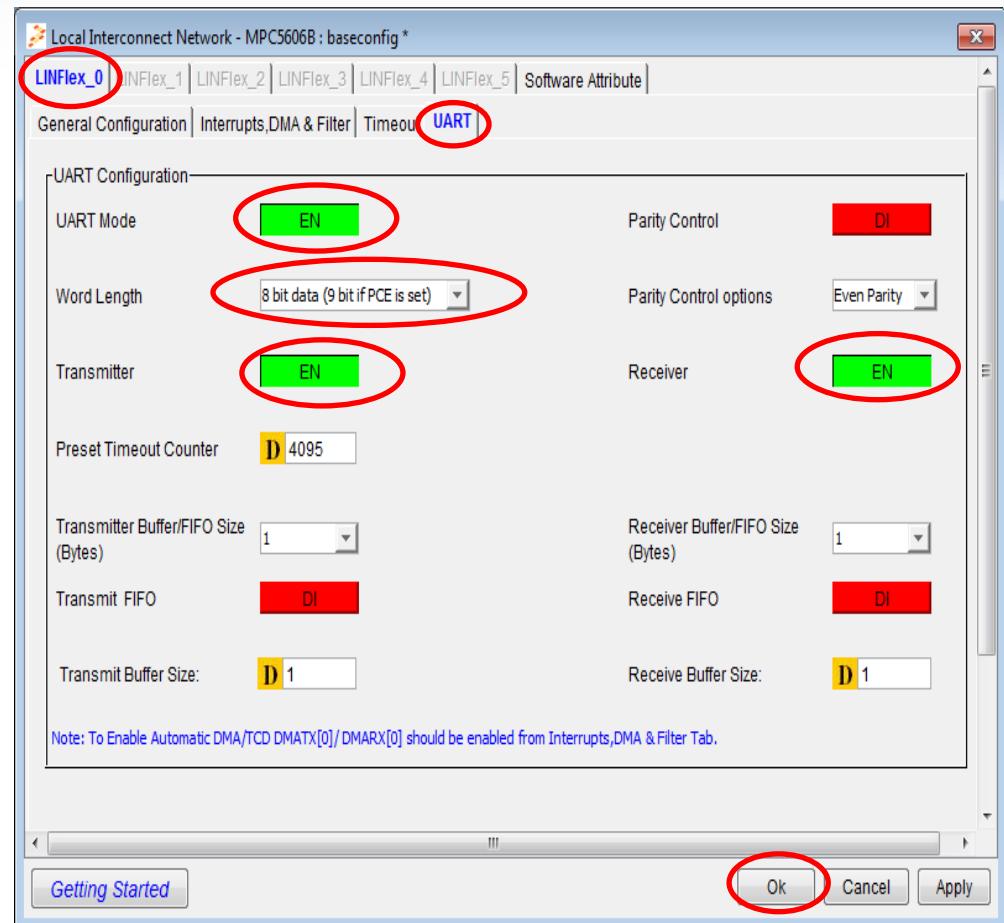


In UART tab, Enable UART

Set Word Length to 8 bit data

Enable Transmitter and Receiver

Select OK to finish LINFlex 0 configuration



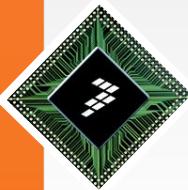
Configure ADC 0



In the Device Setup tab,
Disable Power Down option to
put ADC in normal mode

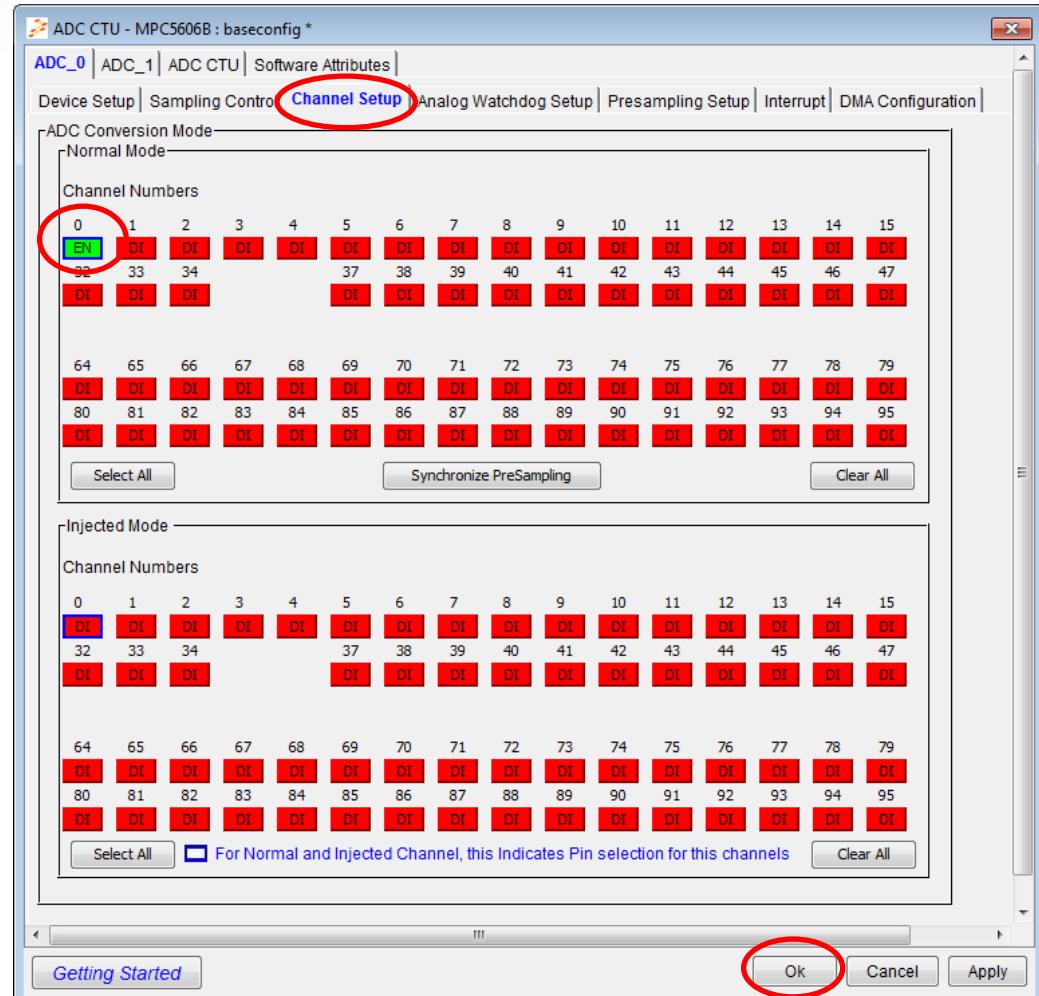
The screenshot shows the 'Device Setup' tab selected in the top navigation bar of the configuration software. The 'General Configuration' section includes fields for ADC System Clock (16 MHz), ADC Resolution (10 bits), Overwrite (DI), Auto Clock Off (DI), Power Down Delay (D0), ADC Clock Select (System clock / 2), Conversion Mode (One Shot), Alignment (Right), Decode Signal Delay (0 ADC Clock Cycles), and Power Down Enable (DI). The 'Power Down Delay' field and the 'Power Down Enable' button are highlighted with red circles.

Configure ADC 0

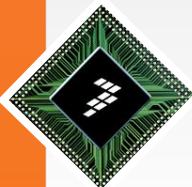


The Potentiometer in TRK-MPC5606B board is connected to PB4 which is ADC Channel 0. In this example we will use Normal conversion mode. Enable Channel 0 in Normal mode

Select OK to finish ADC peripheral configuration.



Configure FlexCAN 1



In TRK-MPC5606B board, the CAN 1 peripheral is connected to CAN transceiver in SBC.
In this example, we will use CAN speed of 500 k bits/sec.

Enable CAN1 module

Disable Freeze and Halt modes

Set Clock Source to System

Set CAN speed to 500. RAppID will configure Phase segments and Propagation segment values automatically.

Select OK to finish CAN 1 configuration.

The screenshot shows the 'FlexCAN - MPC5606B : baseconfig *' configuration window. The 'CAN_1' tab is active. The 'CAN Settings' tab is selected. The 'Module' dropdown is highlighted with a red circle and set to 'Enable'. Other settings include:

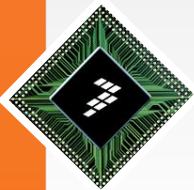
- Soft Reset: Disable
- Freeze Enable: Disable
- Halt FlexCAN: Disable
- Clock Source: System
- FlexCAN System Clock (MHz): 16
- CAN Speed (k bits/s): 500
- CAN Bit Timing:
 - Prescaler Division Factor: D 2
 - Phase Segment 1: 4
 - Phase Segment 2: 4
 - Propagation Segment: 7
 - Resynchronization Jump Width: 1
 - CAN Bit Sampling Mode: 1 Sample
 - Sample Point (%): 75
 - Serial Clock Frequency (MHz): 8
- Enable CAN2.0B Checks:

Notes:

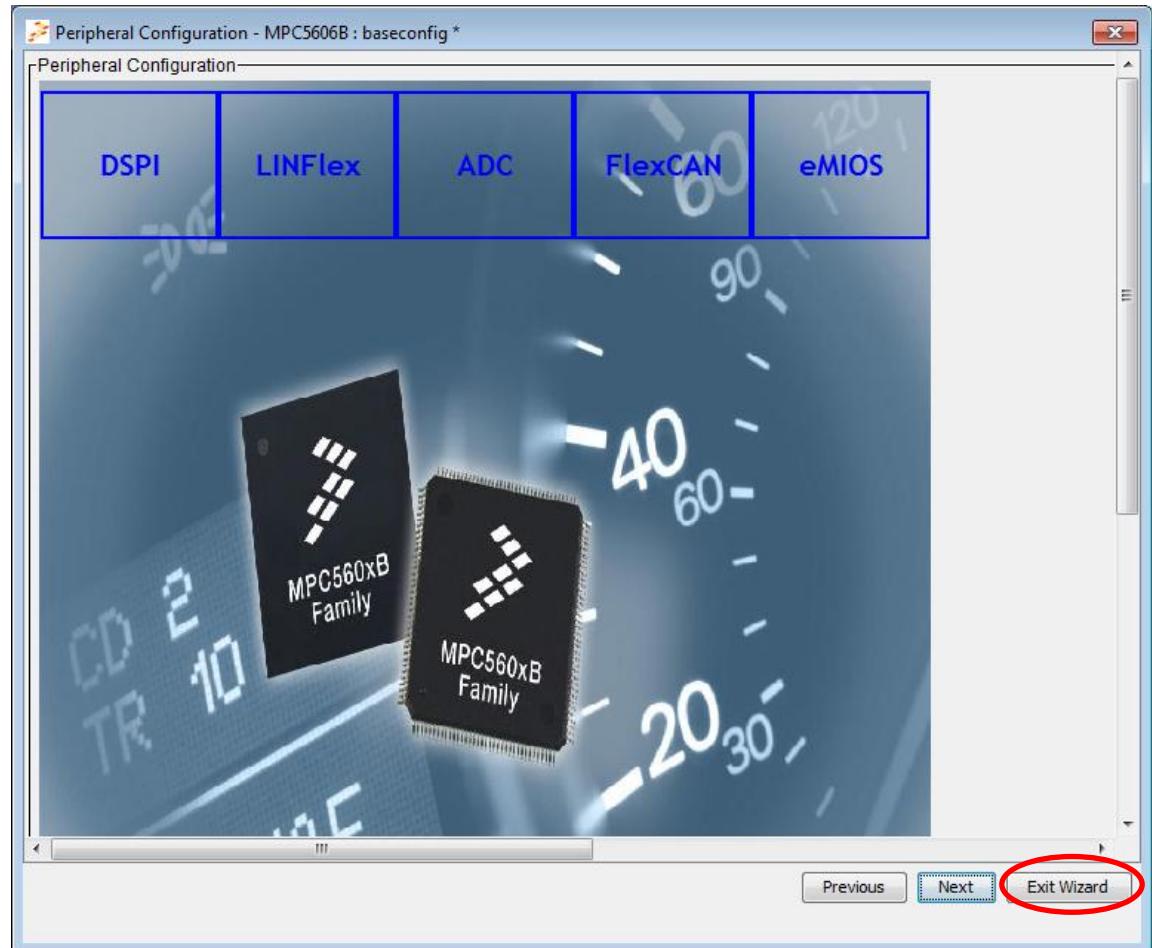
- Note: While setting Phase Segment 1, Phase Segment 2 and Propagation Segment following three condition should be considered
 - 1) Phase Segment 2 >= Phase Segment 1
 - 2) Phase Segment 1 + Phase Segment 2 + Propagation Segment + 1 >= 8
 - 3) Propagation Segment + Phase Segment 1 >= 4
- Note: CAN speed = FlexCAN System Clock (Prescaler*(Phase Segment 1+Phase Segment 2+Propagation Segment+1))

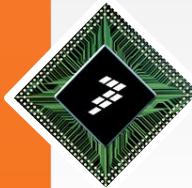
Buttons at the bottom: Getting Started, Ok (circled in red), Cancel, Apply.

Exit Wizard



This completes Peripheral configuration. Select Exit Wizard.

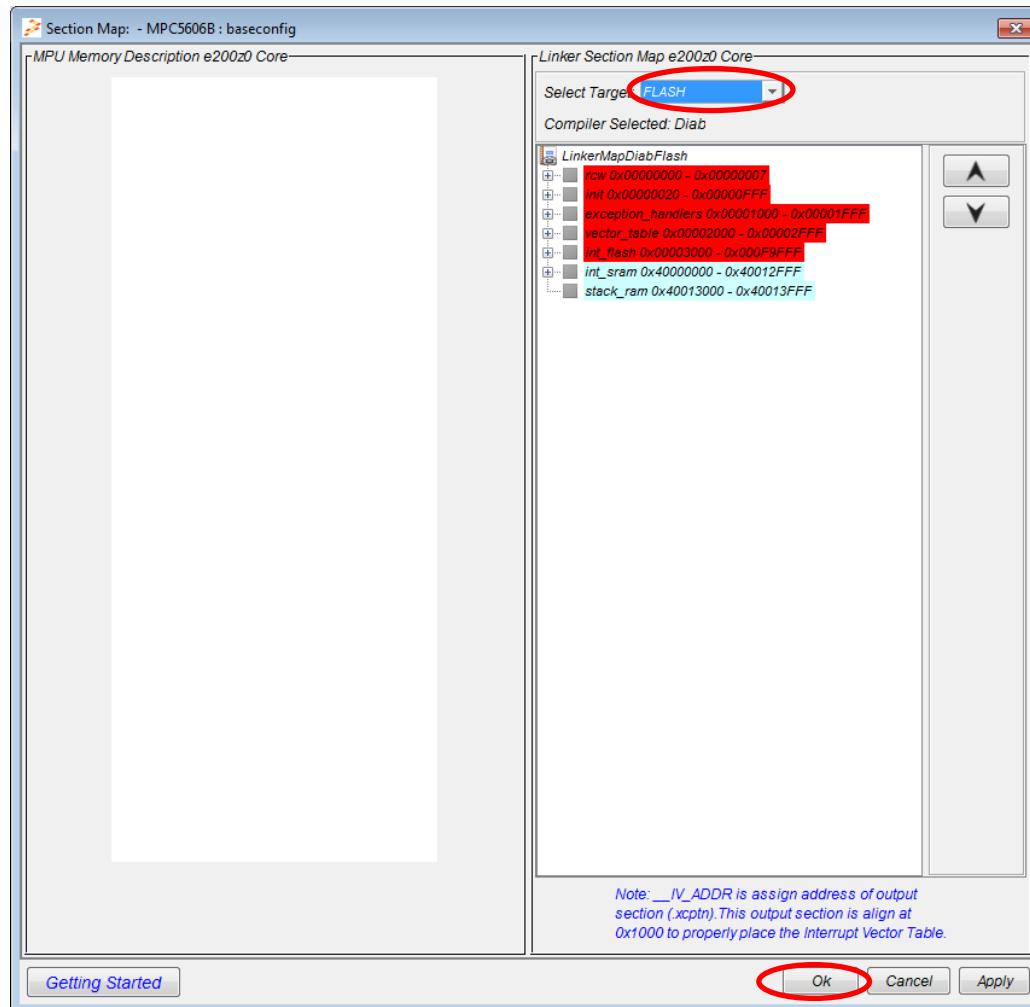




Code Generation

- We have completed all pin and peripheral configurations required for this LED project.
- Now we are ready to generate code. By default, RAppID generates code for RAM. In this example we will generate code for Flash.
- By default, RAppID generates code for all peripherals. In this example, we will select only the peripherals that are used in the example for code generation.

Configure Section Map



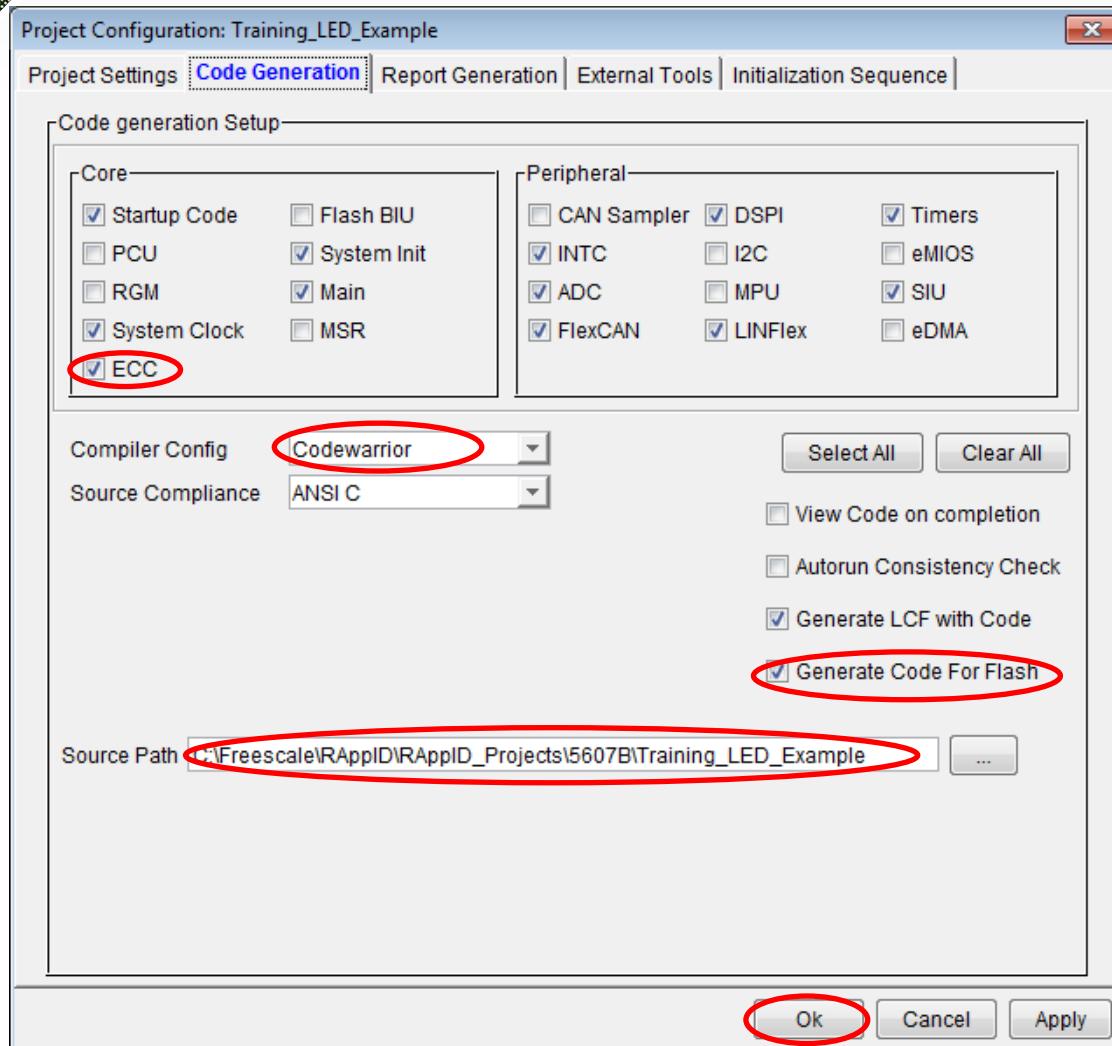
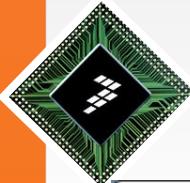
By Default, RAppID is configured to generate code for RAM. For this example, we will change the configuration to generate code for Flash.

From main RAppID window,
select menu *View->View Section
Map*.

Change Target to FLASH

Select Ok

Configure Code Generation



From main RAppID window,
select menu *Configuration->Code Generation*.

Unselect code generation option
for unused peripherals as shown.

Check ECC for code generation.
This is required to generate
initialization code for SRAM and
ECC register.

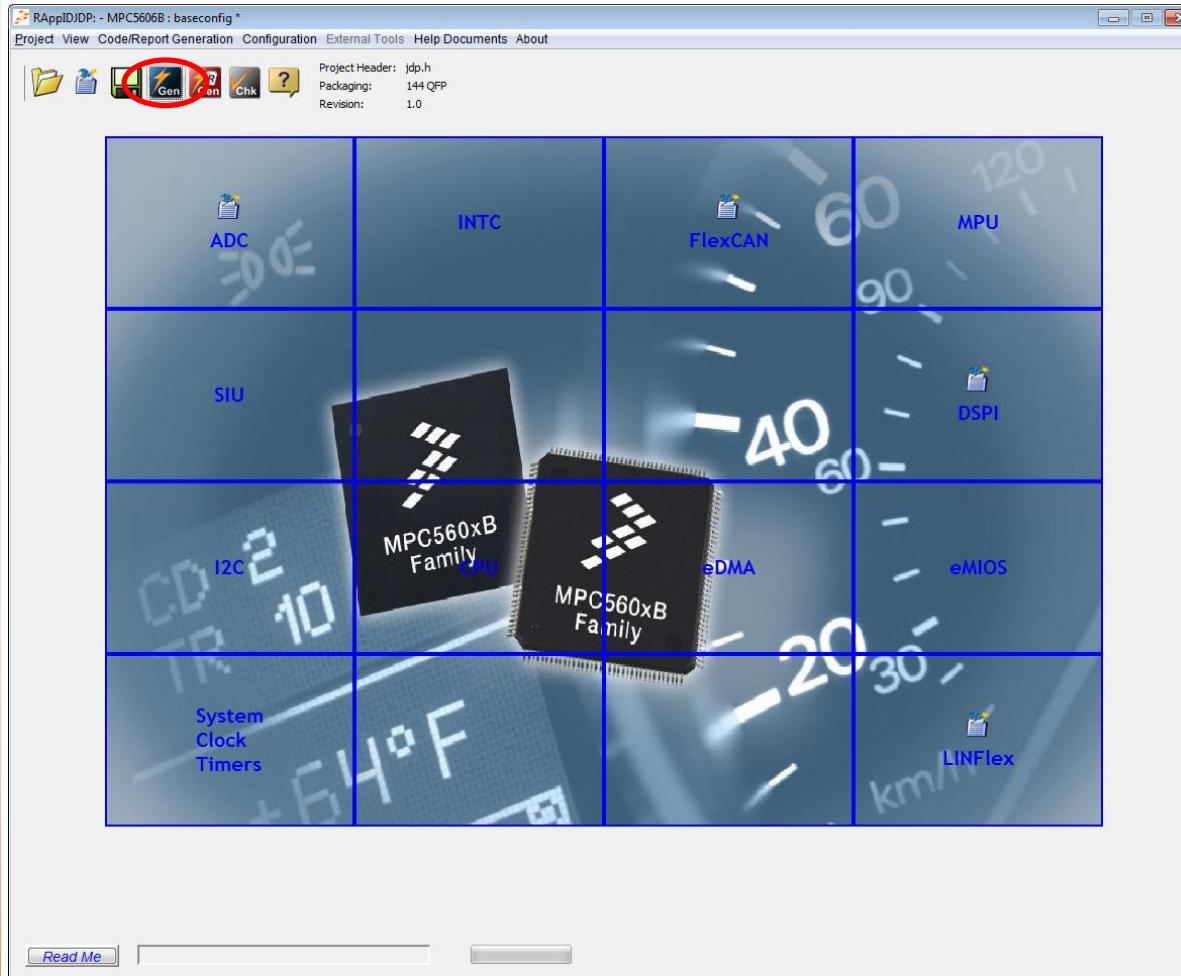
Select CodeWarrior compiler

Select Code for Flash option

Select the directory where code
should be generated

Select Ok

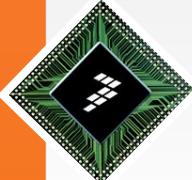
Generate code



Select Code Generation icon to generate code

Save the project when prompted

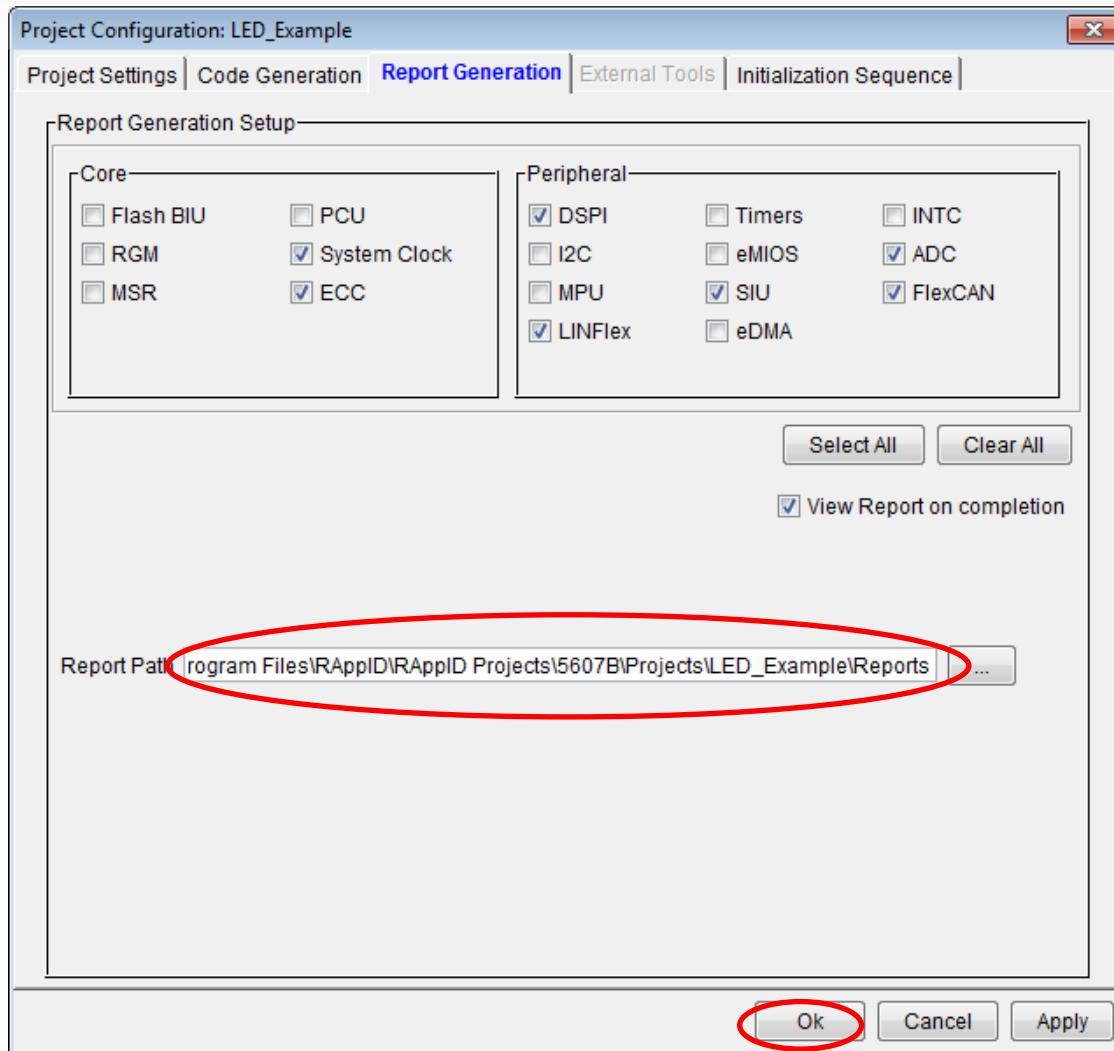
Now we are ready to build and run the project.



Report Generation

- RAppID init tool can generate a comprehensive report on the project
- Since we have finished with configuration for this example, we are ready to generate a report for this project.

Configure Report Generation



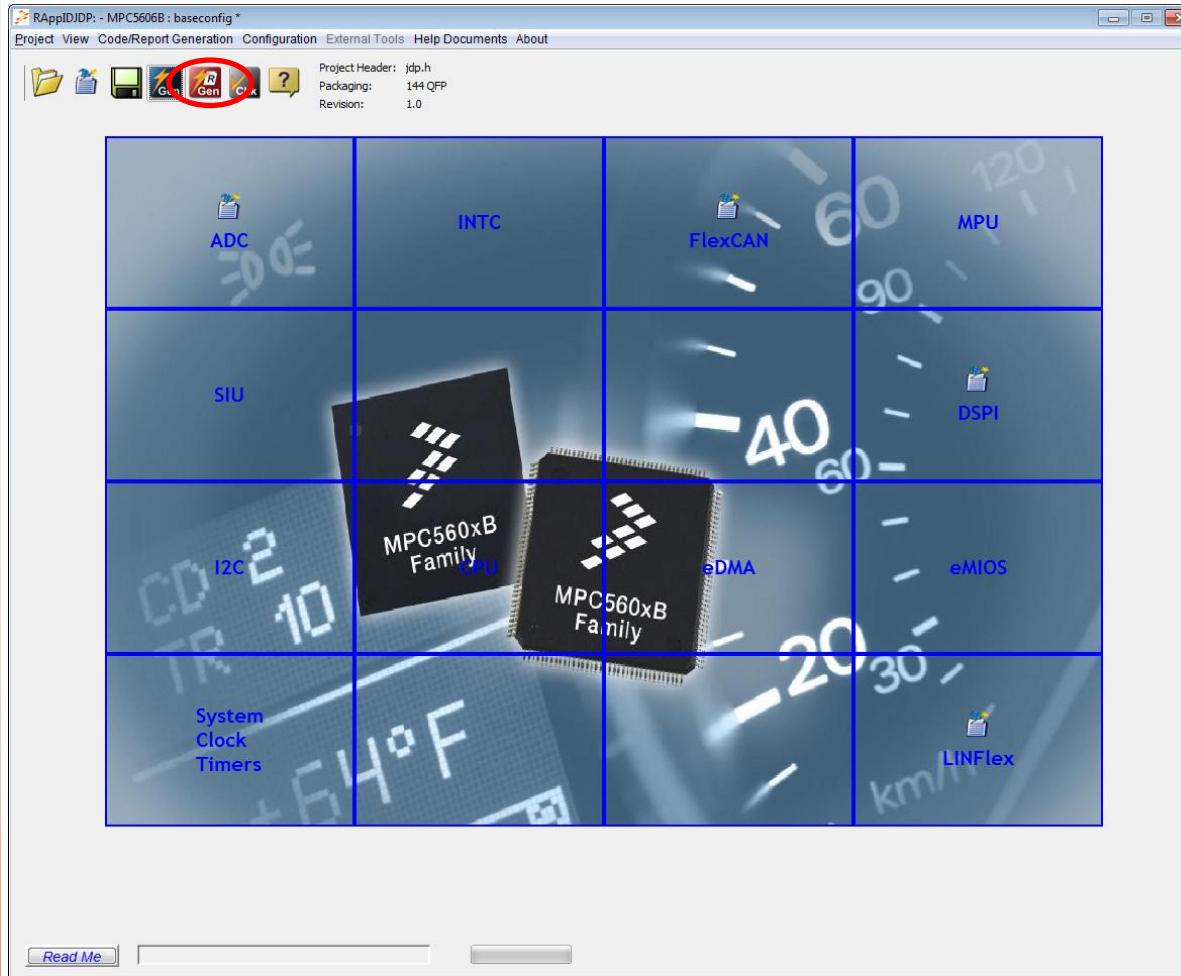
From main RAppID window,
select menu Configuration->Report Generation.

Unselect report generation option
for unused peripherals

Select the directory where report
should be generated

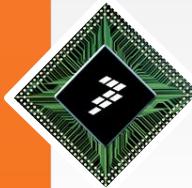
Select Ok

Generate Report



Select Report Generation icon to generate report

Save the project when prompted



RApID Init Project Report

- RApID tool generates a detailed report of the project that covers
 - Initialization Technique, both system and peripheral
 - Detailed Pin Allocation report
 - Section Map report
 - A detailed report of configuration of each peripheral

Project Report – Initialization Technique

Report.htm

file:///C:/Program%20Files/RAppID/RAppID%20Projects/5607B/Projects/LED_Example/Reports/Report.htm

Apps Imported From Fire...

Other bookmarks

Overview

Initialization Technique

Pin Allocation-I

Pin Allocation-II

Section Map

ADC

- ADC_0 Configuration
 - Device Setup
 - Sampling Control
 - Channel Setup
 - Analog Watchdog Setup
 - PreSampling Setup
 - Interrupt
 - DMA Configuration
- ADC_1 Configuration
 - Device Setup
 - Sampling Control
 - Channel Setup
 - Analog Watchdog Setup
 - PreSampling Setup
 - Interrupt
 - DMA Configuration
- ADC_CTU Configuration
 - CTU Setup
 - Registers
 - Software Attributes

DSPI

- DSPI0 Configuration
 - General
 - Clock & Transfer Attributes
 - SPI Commands
 - DMA/Interrupts
- DSPI1 Configuration
 - General
 - Clock & Transfer Attributes
 - SPI Commands
 - DMA/Interrupts
- DSPI2 Configuration
 - General
 - Clock & Transfer Attributes
 - SPI Commands

Configuration Report for MPC5606B
Package: 144 QFP

Project Name: LED_Example (Revision :1.0)
Compiler Compliancy: CodeWarrior, ANSI C
Note: RAppID bit description is in Motorola (Big Endian) format.

Last Save Date: Sun, 13 Oct 2013 22:51:25
Tool Version: 1.2.1.5

Device Settings for: Initialization Technique

System Main Initialization

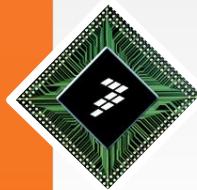
System Initialization (e200z0 Core)

- Interrupt Controller (INTC) Initialization
- Reset Generation Module (RGM) Initialization
- Software Watchdog (SWT) Initialization
- Main Status Register (MSR) Initialization
- System Clock, Mode Entry (ME) & CMU Initialization
- Real Time Clock (RTC) Initialization
- System Timer Module (STM) Initialization
- Periodic Interrupt Timer (PIT) Initialization
- Flash Bus Interface Unit (FBI) Initialization
- Memory Protection Unit (MPU) Initialization
- Enhanced Direct Memory Access (eDMA) Initialization
- FlexCAN Initialization
- CAN Sampler Initialization
- Analog to Digital Converter (ADC) Initialization
- Enhanced Modular I/O (eMOS) Initialization
- Deserial Serial Peripheral Interface (DSPI) Initialization
- Inter-Integrated Circuit Bus (I2C) Initialization
- LNFlex Initialization
- Mode Entry Post Configuration Initialization
- System Integration Unit (SIU) Initialization

Peripheral Initialization

ECC
- Initialization of RAM to GPR values or User Specified Value

Project Report – Pin Allocation



Report.htm file:///C:/Program%20Files/RAppID/RAppID%20Projects/5607B/Projects/LED_Example/Reports/Report.htm

Apps Imported From Fire... Other bookmarks

[Overview](#)

[Initialization Technique](#)

[Pin Allocation-I](#)

[Pin Allocation-II](#)

[Section Map](#)

ADC

- [ADC_0 Configuration](#)
 - [Device Setup](#)
 - [Sampling Control](#)
 - [Channel Setup](#)
 - [Analog Watchdog Setup](#)
 - [PreSampling Setup](#)
 - [Interrupt](#)
 - [DMA Configuration](#)
- [ADC_1 Configuration](#)
 - [Device Setup](#)
 - [Sampling Control](#)
 - [Channel Setup](#)
 - [Analog Watchdog Setup](#)
 - [PreSampling Setup](#)
 - [Interrupt](#)
 - [DMA Configuration](#)
- [ADC_CCU Configuration](#)
 - [CTU Setup](#)
 - [Registers](#)
 - [Software Attributes](#)

DSPI

- [DSPI0 Configuration](#)
 - [General](#)
 - [Clock & Transfer Attributes](#)
 - [SPI Commands](#)
 - [DMA/Interrupts](#)
- [DSPI1 Configuration](#)
 - [General](#)
 - [Clock & Transfer Attributes](#)
 - [SPI Commands](#)
 - [DMA/Interrupts](#)
- [DSPI2 Configuration](#)
 - [General](#)
 - [Clock & Transfer Attributes](#)
 - [SPI Commands](#)

Configuration Report for MPC5606B
Package: 144 QFP

Project Name: LED_Example (Revision 1.0)
Compiler Compliancy: Codewarrior, ANSI C
Note: RAppID/DUDP bit description is in Motorola (Big Endian) format.

Last Save Date: Sun, 13 Oct 2013 22:51:25
Tool Version: 1.2.1.5

DSPI PINS

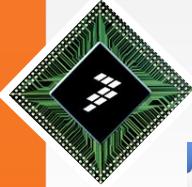
FlexCAN PINS
GPIO PINS
I2C PINS
LINFlex PINS
MISC PINS
eMIOS PINS
SIU-assigned WIZARD-notassigned
GPIO Tristate

Function Assignment for MPC5606B 144 QFP package

		Primary Function	Alternate Function 1	Alternate Function 2	Alternate Function 3	Input Function 0	Input Function 1	Analog Pad Control	Assigned Function						
SIU	Package PCR#	Port Pin #	GPIO Name	dir (PA=3'b000)	ALT 1 (PA=3'b001)	ALT 2 (PA=3'b010)	ALT 3 (PA=3'b011)	IN 0 dir	IN 1 dir	APC	dir	Assigned Function	dir	User Assigned Signal Name	
0	16	PA0	GPIO	IO eMIOS_0_0	IO CLKOUT	O eMIOS_0_13	IO					GPIO Tristate	--		
1	11	PA1	GPIO	IO eMIOS_0_1	IO				NMI I			GPIO Tristate	--		
2	9	PA2	GPIO	IO eMIOS_0_2	IO			O ADC_0_MA2	O			GPIO Tristate	--		
3	90	PA3	GPIO	IO eMIOS_0_3	IO TXD_5	O PCS_1_4	IO				ADC_1_AN50	I	GPIO Tristate	--	
4	43	PA4	GPIO	IO eMIOS_0_4	IO			O PCS_1_0	IO RXD_5 I			GPIO Tristate	--		
5	118	PA5	GPIO	IO eMIOS_0_5	IO TXD_4	O						GPIO Tristate	--		
6	119	PA6	GPIO	IO eMIOS_0_6	IO			O PCS_1_1	IO RXD_4 I			GPIO Tristate	--		
7	104	PA7	GPIO	IO eMIOS_0_7	IO TXD_3	O					ADC_1_AN51	I	GPIO Tristate	--	
8	105	PA8	GPIO	IO eMIOS_0_8	IO eMIOS_0_14	IO			O RXD_3 I			GPIO	I		
9	106	PA9	GPIO	IO eMIOS_0_9	IO			O PCS_1_2	IO			GPIO	I		



Project Report – Section Map



Report.htm file:///C:/Program%20Files/RAppID/RAppID%20Projects/5607B/Projects/LED_Example/Reports/Report.htm

Apps Imported From Firef...

Other bookmarks

Overview

Initialization Technique

Pin Allocation-I

Pin Allocation-II

Section Map

ADC

- ADC_0 Configuration
 - Device Setup
 - Sampling Control
 - Channel Setup
 - Analog Watchdog Setup
 - PreSampling Setup
 - Interrupt
 - DMA Configuration
- ADC_1 Configuration
 - Device Setup
 - Sampling Control
 - Channel Setup
 - Analog Watchdog Setup
 - PreSampling Setup
 - Interrupt
 - DMA Configuration
- ADC_CCU Configuration
 - CTU Setup
 - Registers
 - Software Attributes

DSPI

- DSPI0 Configuration
 - General
 - Clock & Transfer Attributes
 - SPI Commands
 - DMA Interrupts
- DSPI1 Configuration
 - General
 - Clock & Transfer Attributes
 - SPI Commands
 - DMA Interrupts
- DSPI2 Configuration
 - General
 - Clock & Transfer Attributes
 - SPI Commands

Configuration Report for MPC5606B
Package: 144 QFP

Project Name: LED_Example (Revision :1.0)
Compiler Compliancy: Codewarrior, ANSI C
Note: RAppID/DP bit description is in Motorola (Big Endian) format.

Last Save Date: Sun, 13 Oct 2013 22:51:25
Tool Version: 1.2.1.5

Section Map Report

Selected Compiler: Codewarrior
Selected Target: FLASH

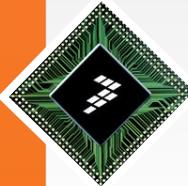
Linker Section Map e200z Core			
Group	O/P Section	I/P Section	File
LinkerMapCWFFlash			
>rom 0x00000000 - 0x00000007			
>init 0x00000020 - 0x00000FFF			
>exception_handlers 0x00001000 - 0x00001FFF			
>vector_table 0x00002000 - 0x00002FFF			
>start_flash 0x00003000 - 0x00009FFF			
>int_sram 0x40000000 - 0x40012FFF			
>stack_ram 0x40013000 - 0x40013FFF			

rw			
Group	O/P Section	I/P Section	File
Group: 1	bam_bootarea		

init			
Group	O/P Section	I/P Section	File
Group: 1	init	init_vie	init
			init_vie

exception_handlers			
Group	O/P Section	I/P Section	File
Group: 1	on_cause	on_cause	on_cause

Project Report – Peripheral Configuration



Report.htm file:///C:/Program%20Files/RAppID/RAppID%20Projects/5607B/Projects/LED_Example/Reports/Report.htm

Apps Imported From Firef... Other bookmarks

Overview
Initialization Technique
Pin Allocation-I
Pin Allocation-II
Section Map

ADC

- ADC_0 Configuration**
 - Device Setup
 - Sampling Control
 - Channel Setup
 - Analog Watchdog Setup
 - PreSampling Setup
 - Interrupt
 - DMA Configuration
- ADC_1 Configuration**
 - Device Setup
 - Sampling Control
 - Channel Setup
 - Analog Watchdog Setup
 - PreSampling Setup
 - Interrupt
 - DMA Configuration
- ADC_CTLU Configuration**
 - CTU Setup
 - Registers
 - Software Attributes

DSPI

- DSPI0 Configuration**
 - General
 - Clock & Transfer Attributes
 - SPI Commands
 - DMA/Interrupts
- DSPI1 Configuration**
 - General
 - Clock & Transfer Attributes
 - SPI Commands
 - DMA/Interrupts
- DSPI2 Configuration**
 - General
 - Clock & Transfer Attributes
 - SPI Commands

Configuration Report for MPC5606B
Package: 144 QFP

Project Name: LED_Example (Revision 1.0)
Compiler Compliancy: Codewarrior, ANSI C
Note: RAppIDDP bit description is in Motorola (Big Endian) format.

Last Save Date: Sun, 13 Oct 2013 22:51:25
Tool Version: 1.2.1.5

Device Settings for: ADC_0

ADC General Setup Information

General Device Settings

Normal Conversion will not start when external Start Signal is detected. Clock will not automatically switch off when ADC is idle.

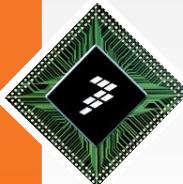
Conversion of ADC command shall be one channel chain at a time. After conversion data is right align also old conversion data will be not overwritten by newer result, hence newer result will be discarded.

Delay between external decode signal and start of sampling pulse is set to 0 clock cycles. Delay between external power signal and start of sampling pulse is set to 0 clock cycles.

External Trigger Injected is Disabled with Trigger Injection Edge as Falling.
Power Down Mode is Disabled. ADC Clock Selected as System clock / 2.
ADC/Cross Triggering unit is Disabled.

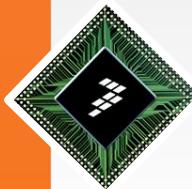
Register

ADC_0_MCR																
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	OWREN	WLSIDE	MODE	0	0	0	0	NSTART	0	JTRGEN	JEDGE	JSTART	0	0	CTUEN	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	CLK _{EL}	DRT _{AIN}	DRT _{AIN}	KO	0	0	0	0	DN



Edit main.c

- The main.c generated by RAppID initializes peripherals and includes an empty while loop. We have to add code to main.c to:
 - Turn on LED1 when Switch S1 is pressed and turn off when S1 is not pressed
 - Turn on LED2 when command ‘1’ is sent serially and turn off LED2 when ‘0’ is sent
 - Turn on LED3 when command ‘1’ is sent via CAN and turn off LED3 when ‘0’ is sent
 - Turn on LED4 when potentiometer input value is ≤ 500 otherwise turn off LED4
- We will be using driver code supplied with the installation disk to perform low level functions. Copy the driver code to the directory where RAppID code was generated



Edit main.c – Global variables

Add following global variables to main.c:

```
/* 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};
/* UART menu string */
unsigned char menuString[] = "Press 1 to turn on and 0 to turn off LED2";
unsigned char newLine[2] = {0x0A, 0x0D};
uint8_t switchState;
uint16_t potValue;
```

Note: This example CodeWarrior project along with source code is provided with the installation disk. You can use Import project option to create this example project within CodeWarrior.



Edit main function

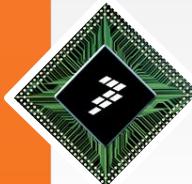
Add following code to *main* function:

Add a call to driver to initialize SBC to enable CAN communication:

```
SBC_Init_DBG();
```

The 4 LEDs are active low and they turn on at start up. Turn them off:

```
GPIO_SetState(68, 1);  
GPIO_SetState(69, 1);  
GPIO_SetState(70, 1);  
GPIO_SetState(71, 1);
```



Edit main function (Continued)

We will receive CAN messages with CAN ID = 1. Add a call to CAN driver to setup mailbox 0 with Id = 1:

```
SetCanRxFilter(1, 0, 0);
```

Add a call to UART driver to initialize UART buffer:

```
UartBufInit();
```

Add a call to function that transmits a menu string via UART asking the user to type 1 to turn on LED2 and type 0 to turn off LED2

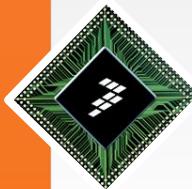
```
PrintMenu();
```



Edit main function (Continued)

Within while loop, add function calls to process GPIO, UART, CAN and ADC peripherals.

```
while(1)
{
    ProcessGPIO();
    ProcessUART();
    ProcessCAN();
    ProcessADC();
}
```



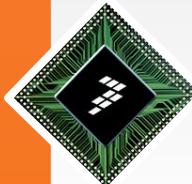
Add ProcessUART function

We will add function *ProcessUART* to process messages received via UART. This function will call the UART driver functions to check if any UART message byte is received. If a byte is received with value ‘1’ , it will turn On LED2 and if a byte is received with value ‘0’, it will turn Off LED2. Bytes with any other values will be ignored. If any message byte is received, it will transmit the menu instructing user to send command to turn On/Off LED2. The code is shown below:



Add ProcessUART function (Continued)

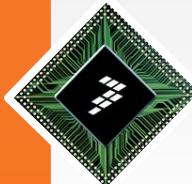
```
void ProcessUART(void)
{
    unsigned char data;
    UartRxFillBuf();           /* Check if serial message received */
    if(UartRxBufEmpty() != 1)
    {
        data = UartRxDataByte();      /* If received data byte is 1, turn on LED2 */
        if (data == '1')
        {
            GPIO_SetState(69, 0);
        }
        else if (data == '0')      /* If received data byte is 0, turn on LED2 */
        {
            GPIO_SetState(69, 1);
        }
        PrintMenu();
    }
}
```



Add PrintMenu function

We will add function *PrintMenu* to transmit a menu string via UART asking the user to type 1 to turn on LED2 and type 0 to turn off LED2. The code is shown below:

```
void PrintMenu(void)
{
    UartTxMsg((unsigned char *)newLine, 2);
    UartTxMsg((unsigned char *)menuString, 40);
    UartTxMsg((unsigned char *)newLine, 2);
}
```



Add ProcessCAN function

We will add function *ProcessCAN* to process messages received via CAN. This function will call the CAN driver functions to check if any CAN message is received. If a CAN message is received with first data byte value of 1, it will turn On LED3 and if the first data byte value is 0, it will turn Off LED3.

If the first data byte is either 0 or 1, it will transmit CAN message ***msgOKCAN***, other wise it will transmit CAN message ***msgErrorCAN***. The code is shown below:

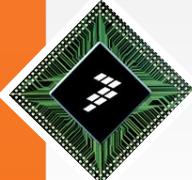


Add ProcessCAN function (Continued)

```
void ProcessCAN(void)
{
    can_msg_struct msgCanRX;

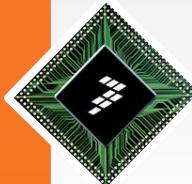
    if (CanRxMbFull(0) == 1)          /* Check if CAN message received */
    {
        msgCanRX = CanRxMsg(0);

        if (msgCanRX.data[0] == 0)      /* If received 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 received data byte is 1, turn on LED3 and send positive response */
        {
            GPIO_SetState(70, 0);
            CanTxMsg (2, 1, 8, (uint8_t *)msgOKCAN, 0);
        }
    }
}
```



Add ProcessCAN function (Continued)

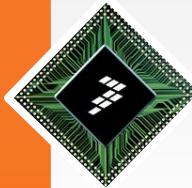
```
else          /* If received data byte is not 0 or 1, send a negative response */  
{  
    CanTxMsg (2, 1, 8, (uint8_t *)msgErrorCAN, 0);  
}  
}  
}
```



Add ProcessGPIO function

We will add function *ProcessGPIO* to process input switch S1. If switch S1 is pressed, it will turn On LED1, otherwise turn Off LED1. The code is shown below:

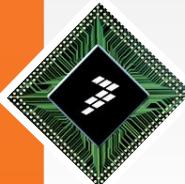
```
void ProcessGPIO(void)
{
    switchState = GPIO_GetState(64); /* Check switch S1 state */
    if (switchState)           /* If Switch S1 is pressed, turn on LED1, other wise turn off LED1*/
    {
        GPIO_SetState(68, 0);
    }
    else
    {
        GPIO_SetState(68, 1);
    }
}
```



Add ProcessADC function

We will add function *ProcessADC* to process potentiometer input. If potentiometer value is ≤ 500 , it will turn on LED1, otherwise turn off LED1. The code is shown below:

```
void ProcessADC(void)
{
    potValue = Pot_Get_Value();
    if(potValue <= 500)      /* If Potentiometer input is <= 500 turn on LED4, other wise turn off LED4 */
    {
        GPIO_SetState(71, 0);
    }
    else
    {
        GPIO_SetState(71, 1);
    }
}
```



Edit main.c – Add include files and function prototypes

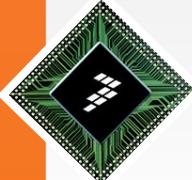
Add following include files to main.c:

```
#include "CANapi.h"  
#include "sbc_hld.h"  
#include "gpio_drv.h"  
#include "UART_drv_api.h"  
#include "pot_hld.h"
```

Add following function prototypes to main.c:

```
void ProcessUART(void);  
void PrintMenu(void);  
void ProcessCAN(void);  
void ProcessGPIO(void);  
void ProcessADC(void);
```

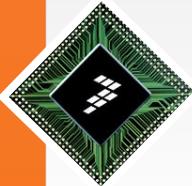
Note: This example CodeWarrior project along with source code is provided with the installation disk and is copied to your system during installation.



Create CodeWarrior project

- Now we are ready to compile and build the code. To do this,
 - create an empty CodeWarrior project
 - add RAppID generated code and driver code to the CodeWarrior project using CodeWarrior script utility – *cwpjmaker_0.1.exe*
 - Build the code using Codewarrior
- The next few slides explains these steps

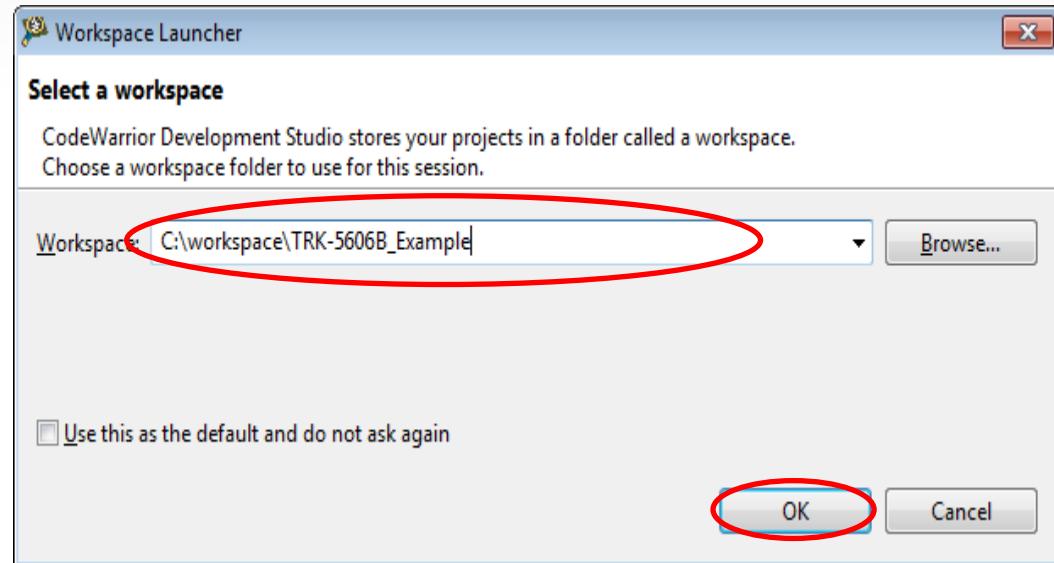
Create CodeWarrior project



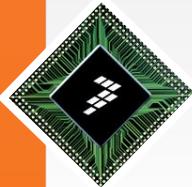
Launch CodeWarrior 10.5 from Windows Start menu

Provide a workspace name

Select OK

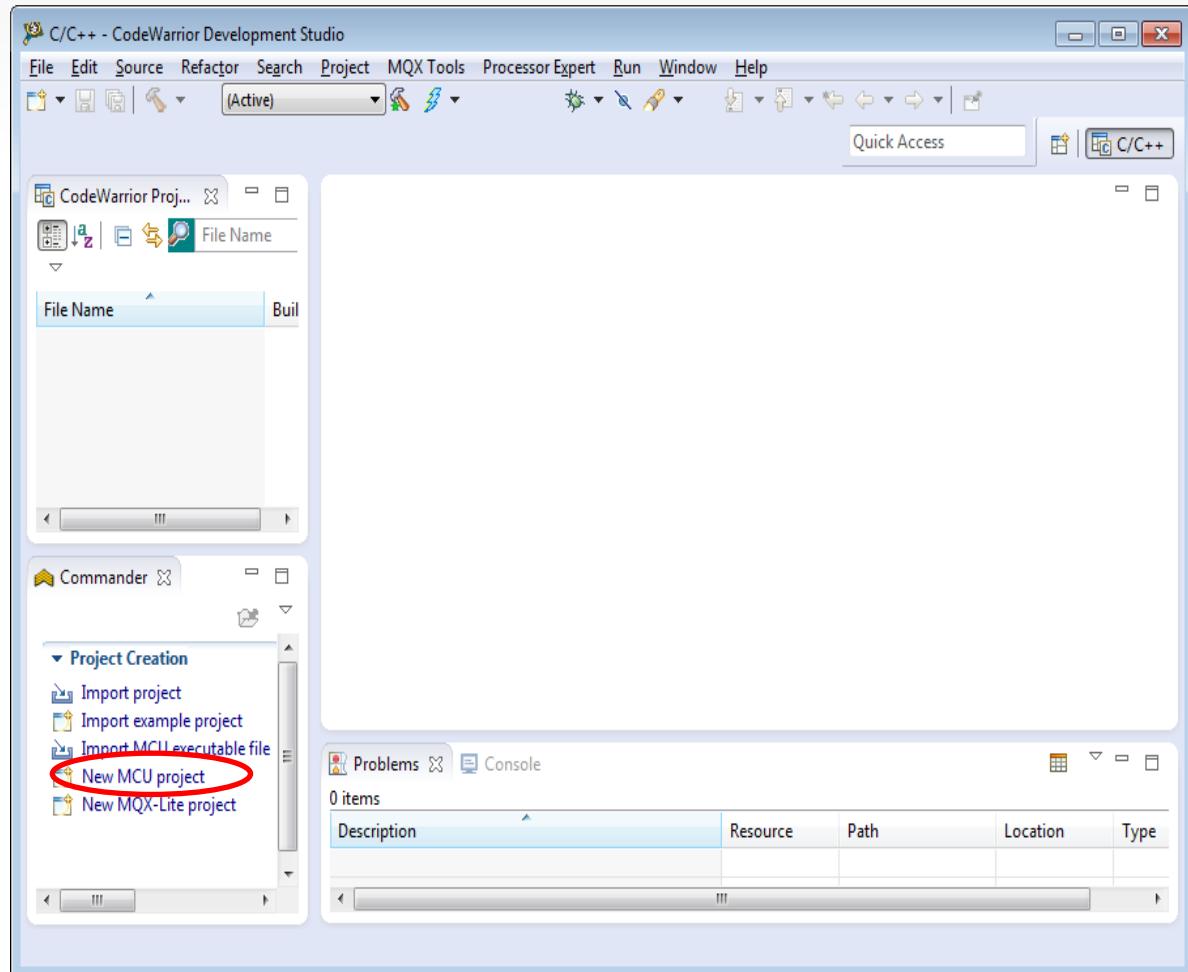


Create CodeWarrior project



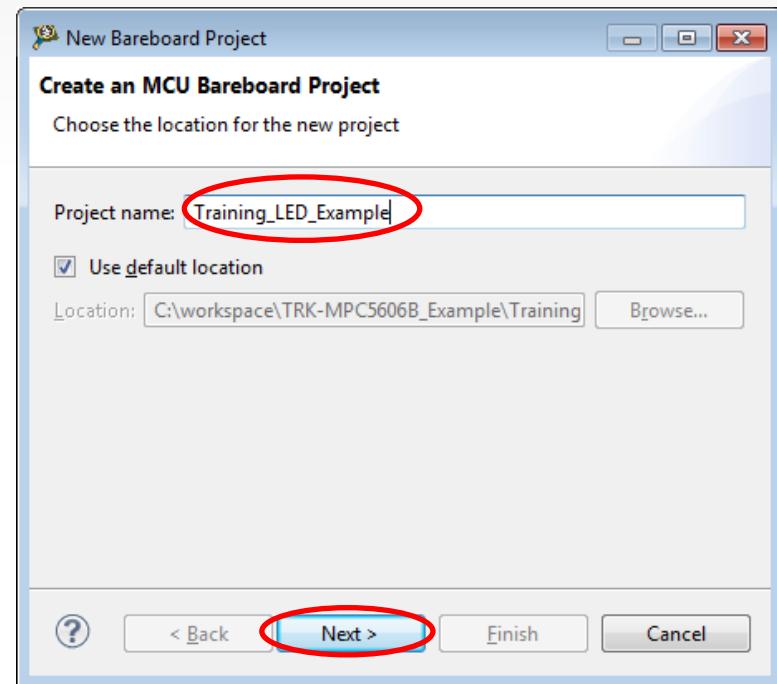
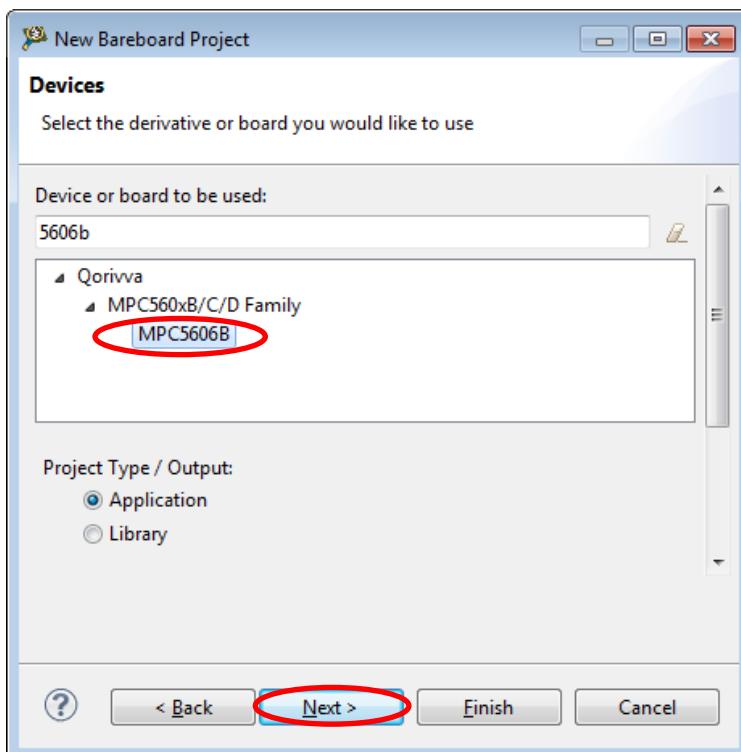
We will first create a Bareboard project for MPC5606B

To create a new Bareboard project, Select “New MCU project” from Commander window.



Create CodeWarrior project

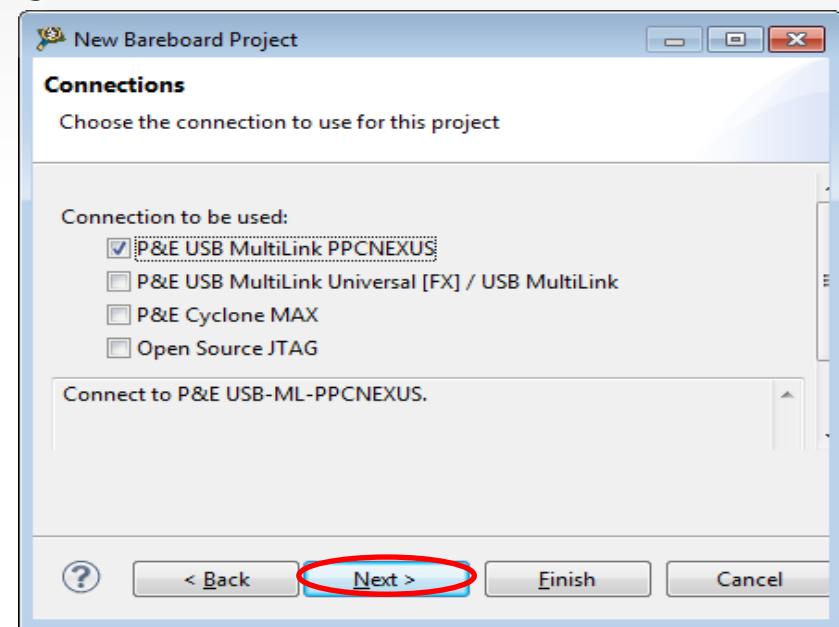
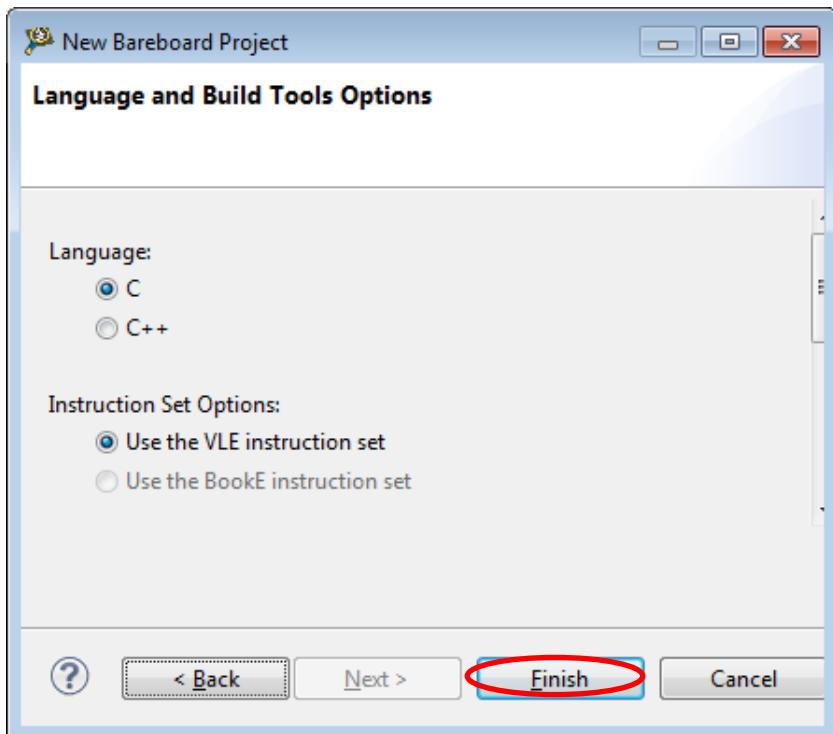
Provide a name for the project –
Training_LED_Example
select Next



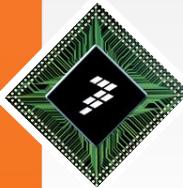
Select Qorivva->MPC560xB/C/D->MPC5606B device from the list
Select Next

Create CodeWarrior project

Accept the default connection option and select Next



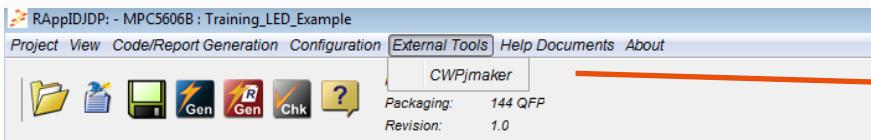
We will be using C code and VLE instructions for this example project. Accept the default options and select Finish



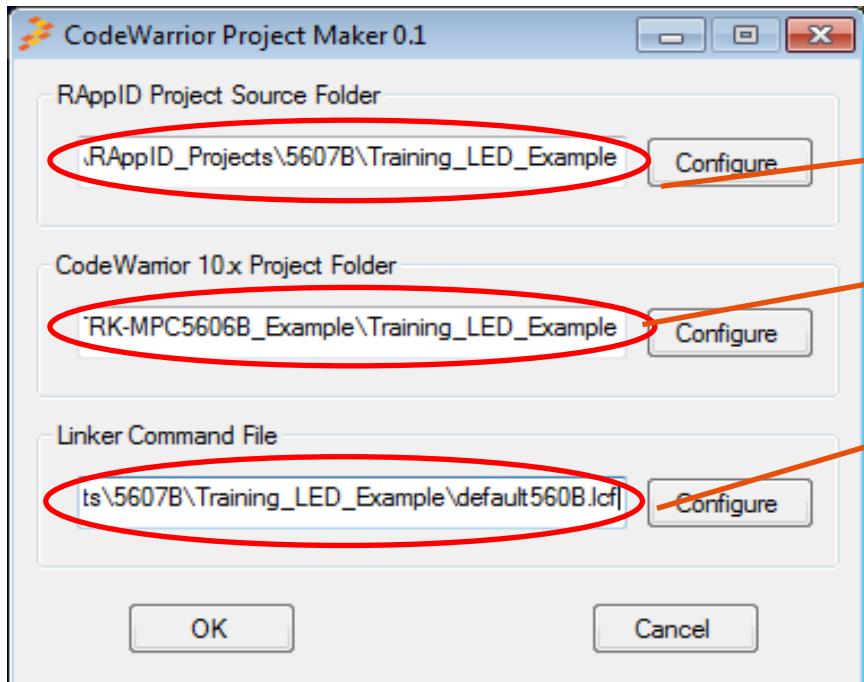
Create CodeWarrior project

- When Finish is selected, CodeWarrior creates a new bareboard project with CodeWarrior generated code and linker file included in the project.
- Since we have generated code using RAppID for LED_Example project, we want to remove all the CodeWarrior generated code and add RAppID generated code and linker file along with the driver code to be included in the CodeWarrior project.
- This can be done using CodeWarrior project maker utility – *cwpjmaker_0.1.exe*
- First close the Training_LED_Example project by selecting menu Project->Close Project
- Run *cwpjmaker* utility which is provided with the installation disk: This can be done by selecting the file from the installed directory or by using the menu in RAppID - *External Tools->CWPjmaker*

Run cwpjmaker utility



Execute cwpjmaker utility from RAppID menu



Provide directory path where RAppID code is generated

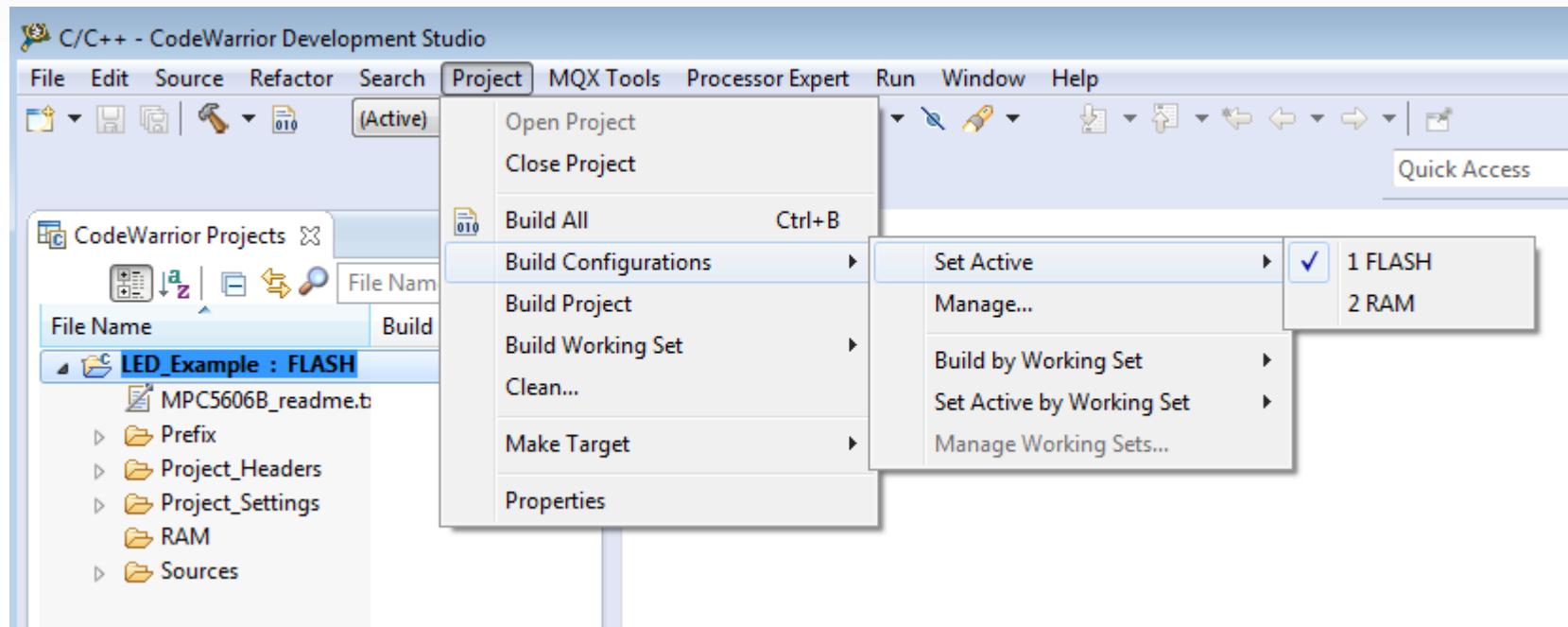
Provide directory path where CodeWarrior project is created

Provide the path of linker file created by RAppID tool

The script will remove CodeWarrior generated code from the project and will add all the code in the directory specified in the “RAppID Project Source Folder” edit box.

The script will also add/modify all the CodeWarrior project settings required to build Training_LED_Example project

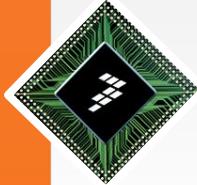
Create CodeWarrior project



Now re-open the LED_Example project by selecting menu Project->Open Project

Since we generated code for Flash, we need to change the build configuration to Flash. To do this, select menu Project->Build Configurations->Set Active->FLASH

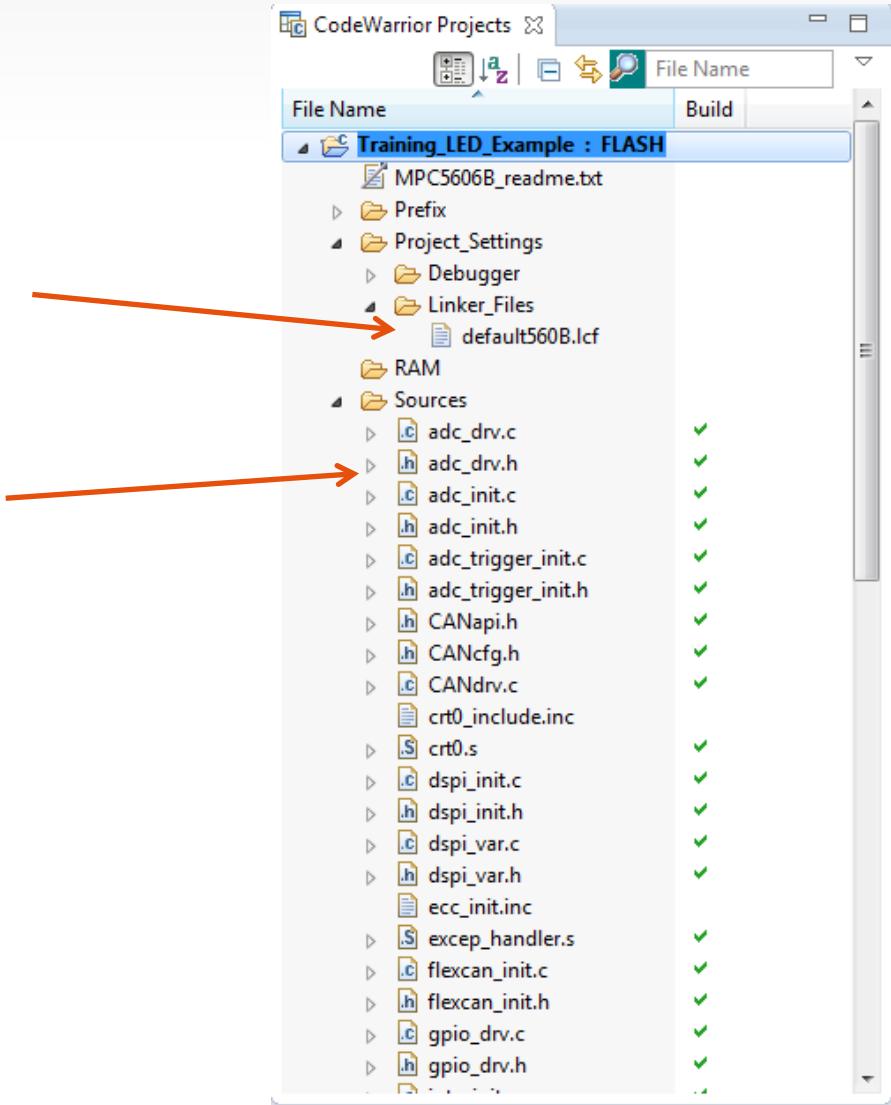
Create CodeWarrior project



Now the project should contain:

RAppID generated linker file

RAppID generated source files and driver code copied from the installation disk



Create CodeWarrior project

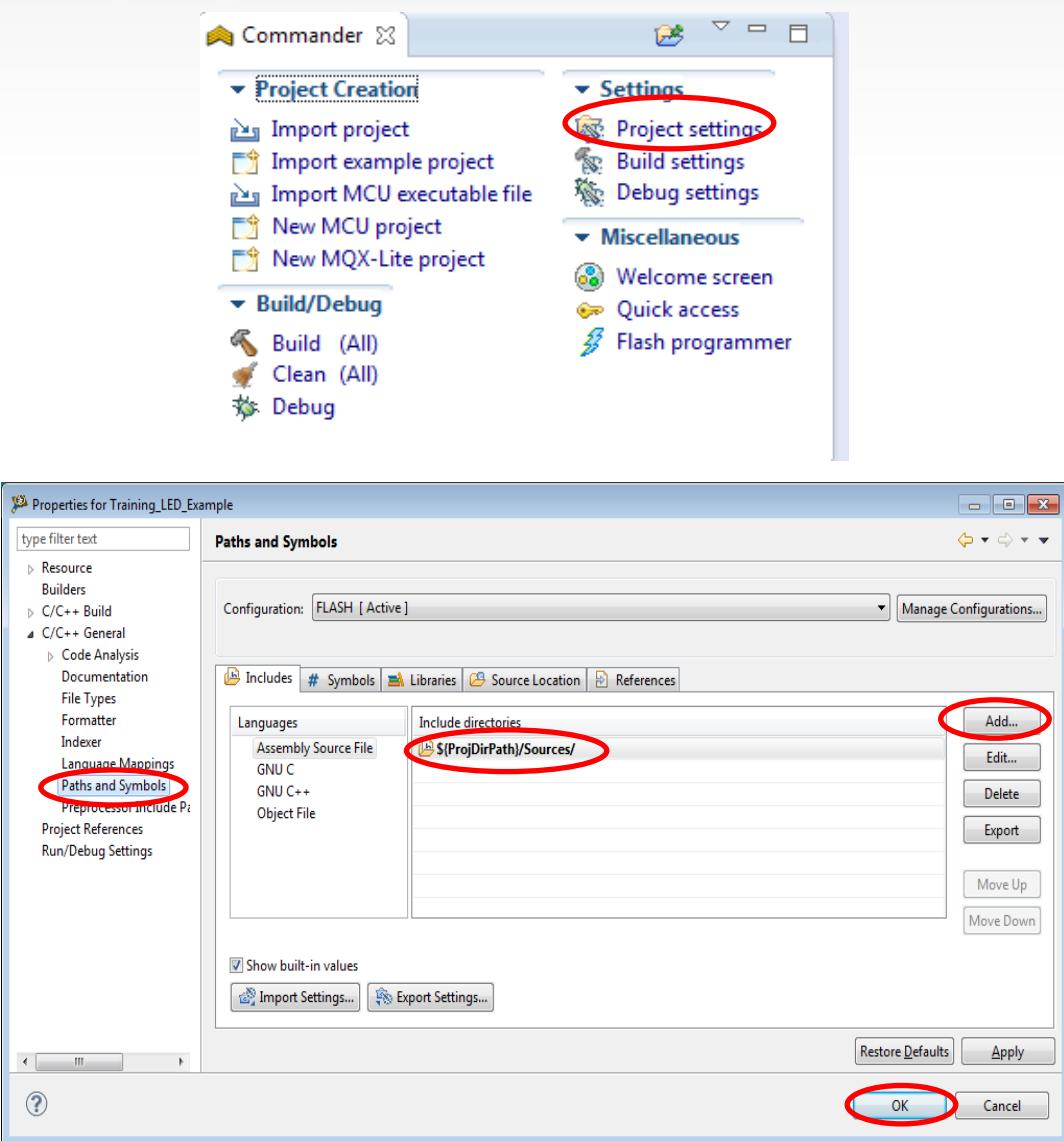


We have to add path of RAppID generated assembly source files to CodeWarrior path.

Select *Project Settings* menu from Commander window.

Under *Paths and Symbols*->*Assembly Source Files*, add a new include directory:
\${ProjDirPath}/Sources/

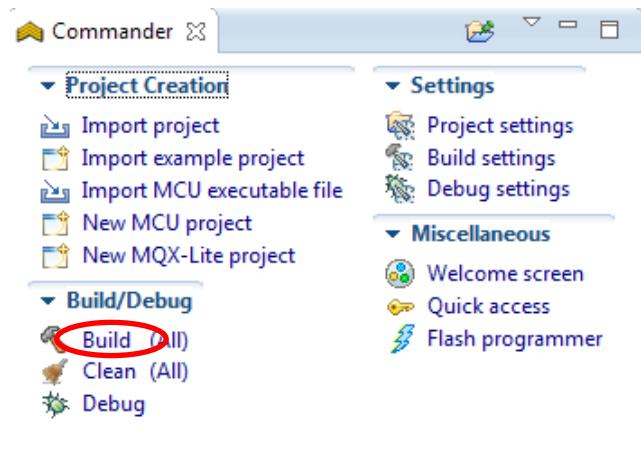
Select OK



Build project



To build Training_LED_Example project , select Build from Commander window



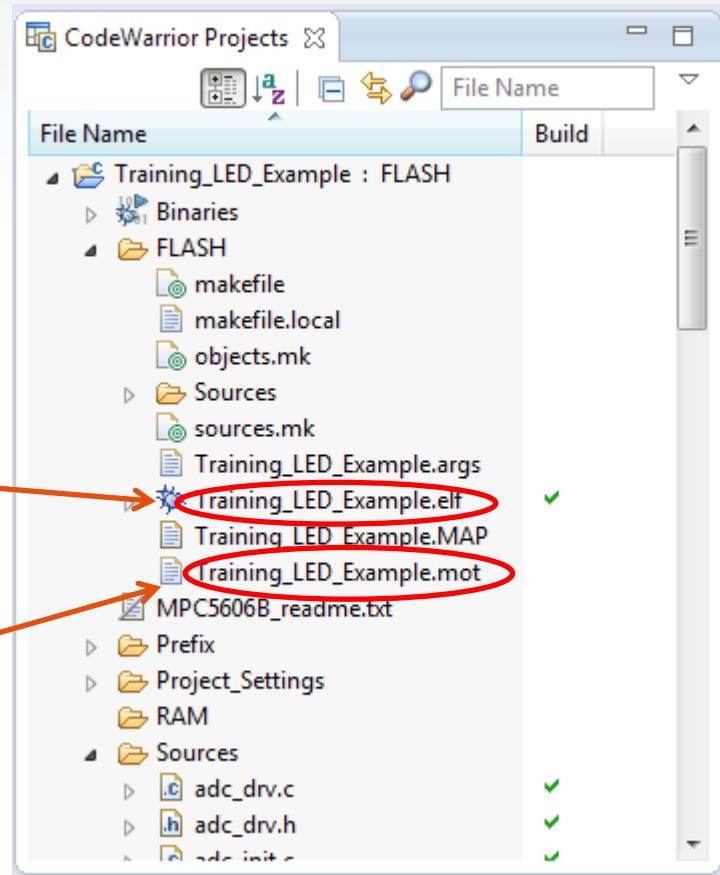
Build project



The build command will produce 2 executable files:

Training_LED_Example.elf
(executable with debug symbols)

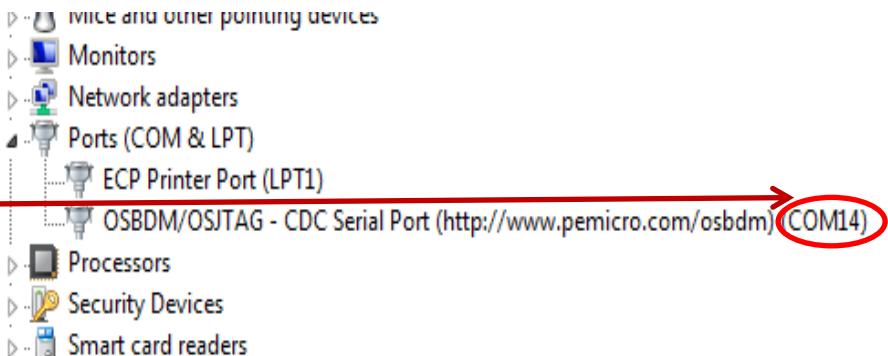
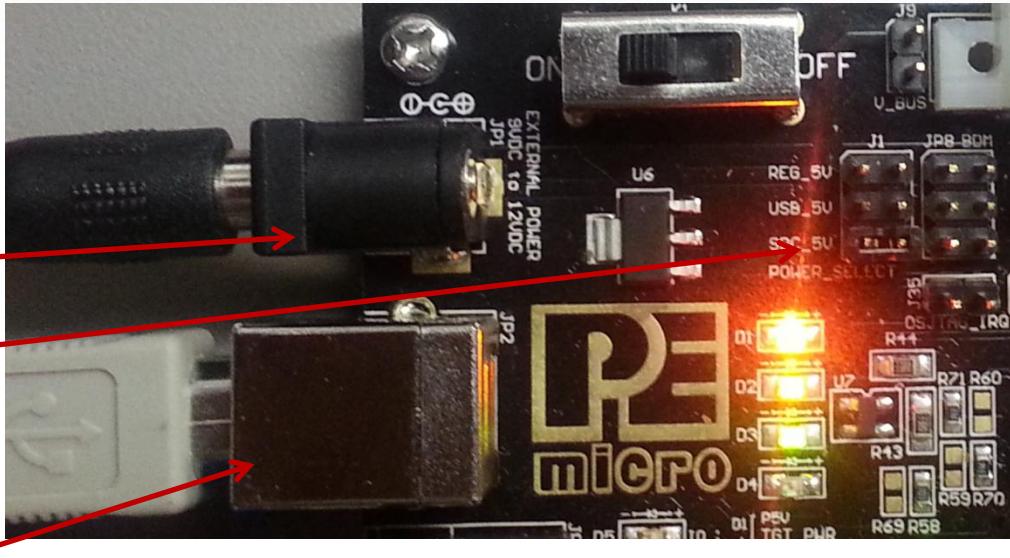
Training_LED_Example.mot
(S-record)

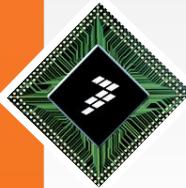




Jumper settings and Power connections

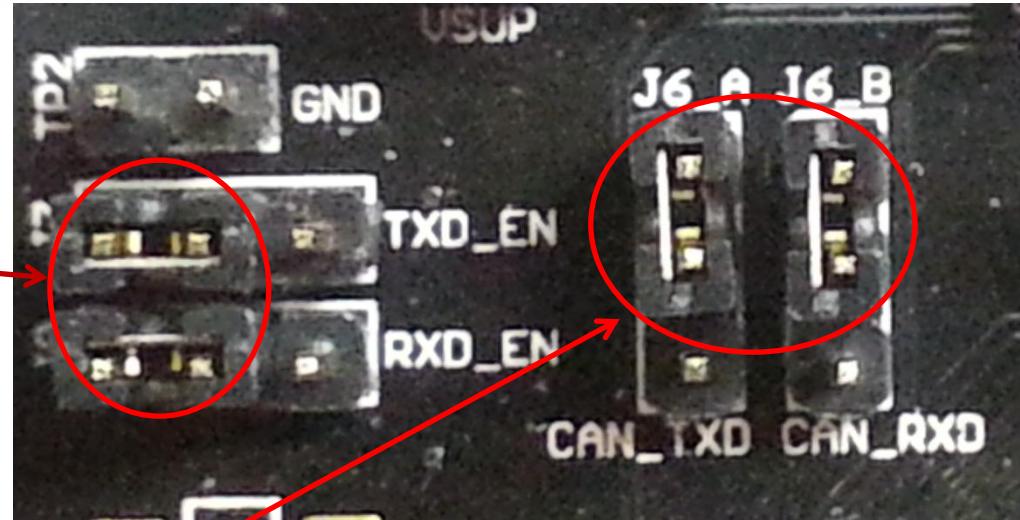
- Since we are using CAN in this example, we need to enable CAN.
- To enable CAN, the board needs to power SBC using external 12V supply.
- Connect jumpers across SBC_5V of J1
- Connect External power to JP1
- Connect your computer to JP2 via USB cable. This connection provides virtual serial port. Confirm this by checking available COM ports in Windows Device manager. In the example shown, the COM port assigned is COM14

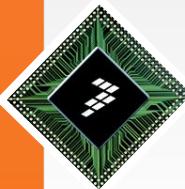




Jumper settings and Power connections

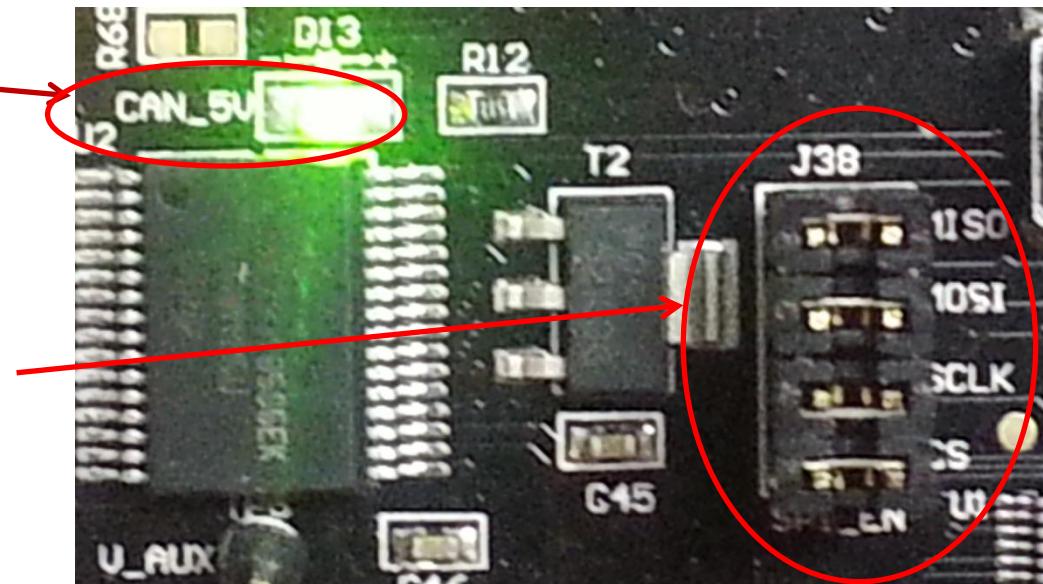
- In this example, we will use Virtual serial port for UART communication. To enable this, connect jumper J7 (TXD_EN) and J8 (RXD_EN) to position 1-2
- To enable CAN communication, connect J6_A (CAN_TXD) and J6_B (CAN_RXD) to position 1-2.

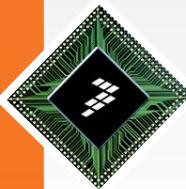




Jumper settings and Power connections

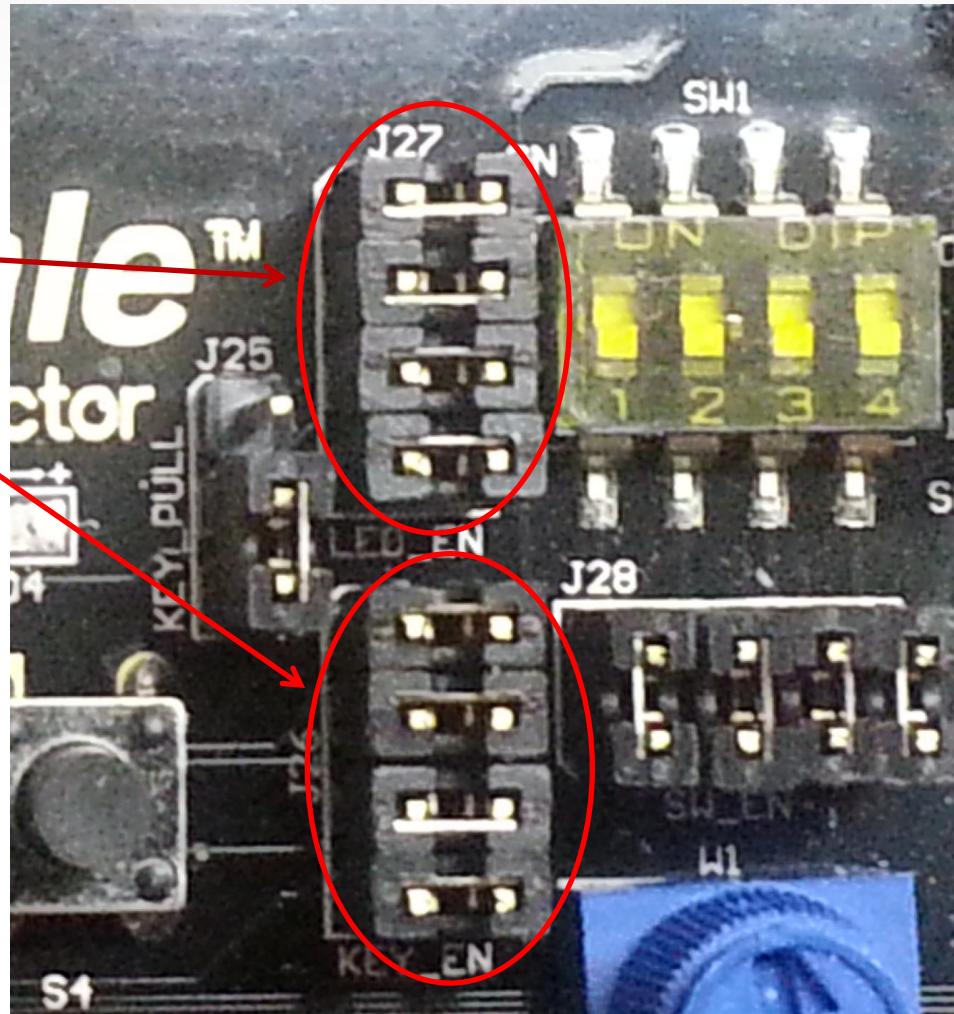
- The CAN_5v should be lit when SBC is powered
- In this example, we will use DSPI 1 to communicate with SBC. To enable DSPI, connect all 4 jumpers in J38

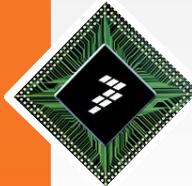




Jumper settings and Power connections

- To enable LEDs, connect all 4 jumpers in J27
- To enable input switches, connect all 4 jumpers in J26





Flash code to target

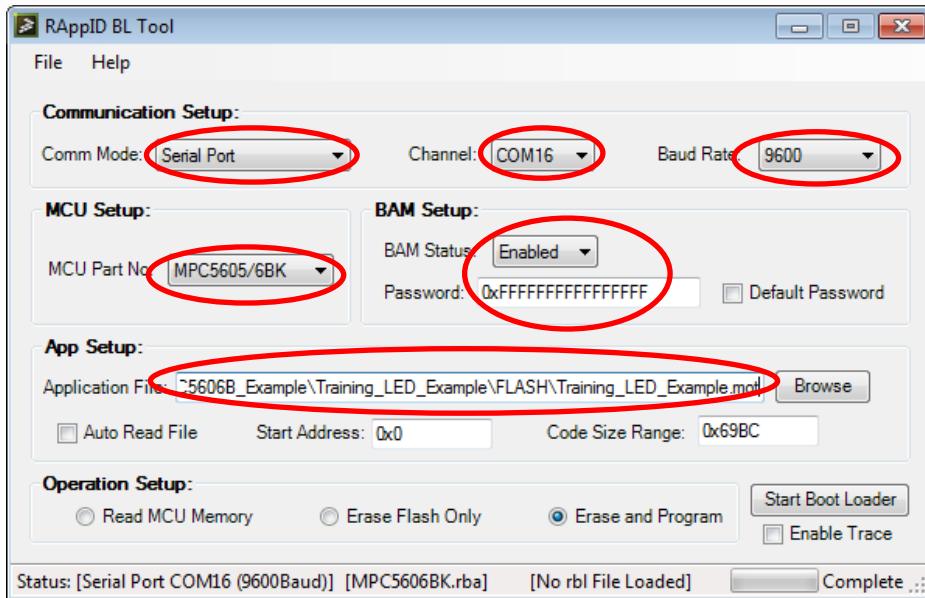
- Before Flashing the code, make sure the TRK-MPC5606B board is connected to external power and to your computer via USB cable
- Set the jumper of J17 to position 1-2 which pulls FAB high and jumper J18 to position 2-3 to set ABS low
- We will use RAppID Bootloader utility to flash S-record file Training_LED_Example.mot using serial port

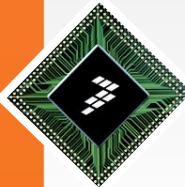




Flash code to target

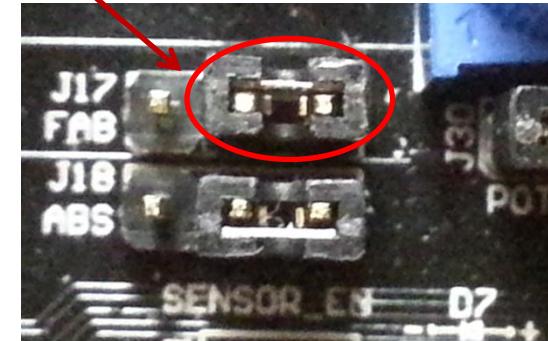
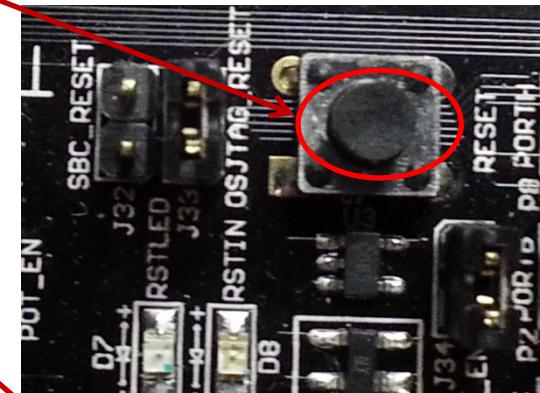
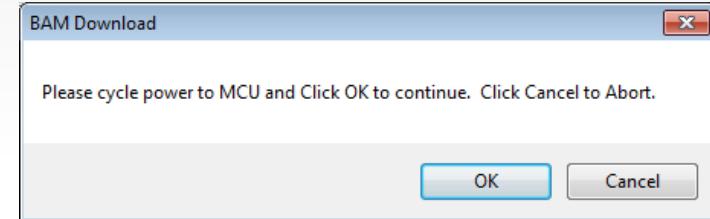
- Launch RAppID bootloader utility from Windows Start menu
- Select Serial Port as Comm Mode
- Select the correct COM channel
- Select 9600 baud
- Select MPC5605/6BK as MCU part number
- Set BAM status as Enabled
- Enter password as 0xFFFFFFFFFFFFFF
- Enter the path for the file to be flashed:
Training_LED_Example.mot
- Select Start Boot Loader button to start the flash process

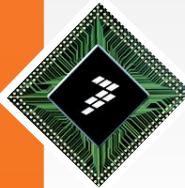




Flash code to target

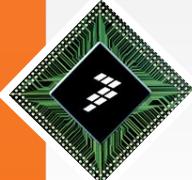
- When asked to cycle power to MCU, press the reset button on the board. Flashing process should start.
- After Flash is complete, Move the jumper J17 to position 2-3 to pull FAB low.
- Turn the power off to the module and re-apply the power
- The code should be running now.





Testing code

- Press switch S1 to turn on LED1 and release S1 to turn it off
- Send serial command 1 to turn on LED2 and 0 to turn it off
- Send CAN command with first byte = 1 to turn on LED3 and first byte = 0 to turn off LED3
- Turn the potentiometer halfway to observe LED4 turning On/Off

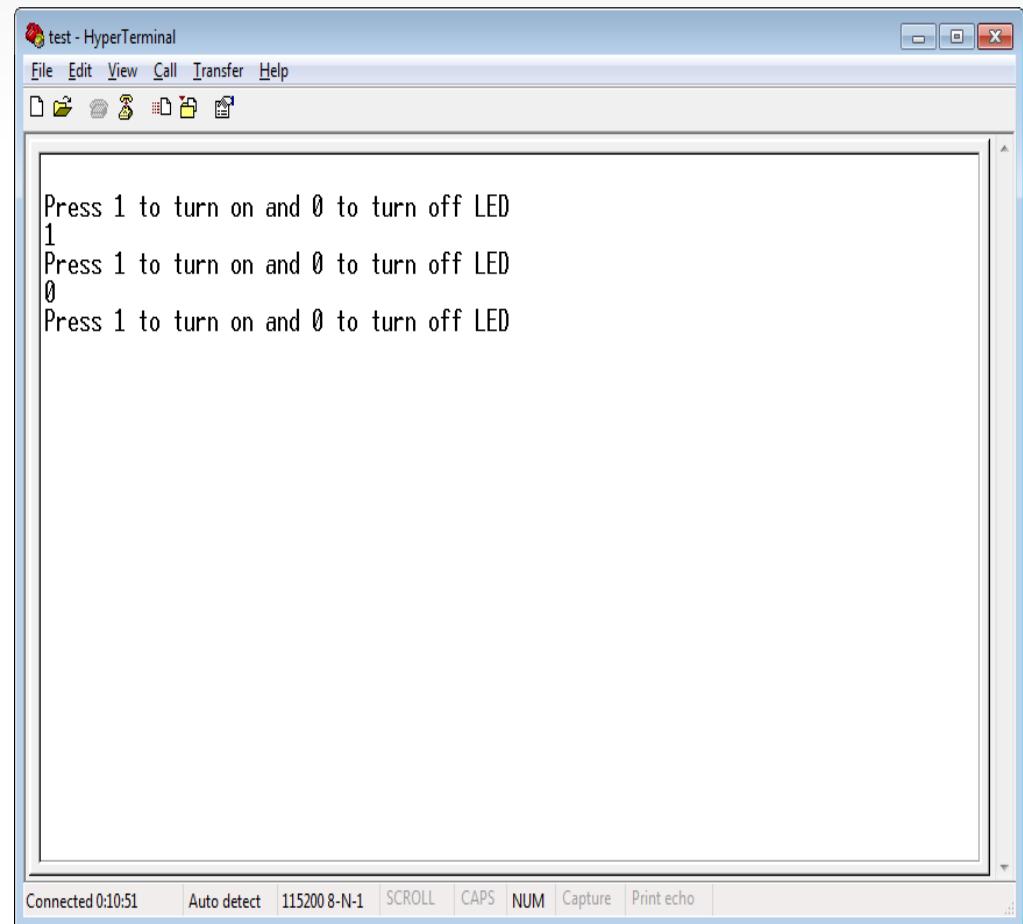


Serial command to turn On/Off LED2

Use a terminal tool like HyperTerminal or PuTTY to send serial command to turn LED2 On/Off

Set the connection parameters to 115200 baud, 8 Data bits, No parity and 1 Stop bits

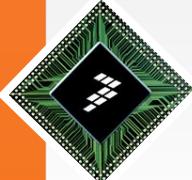
When you press 1, LED2 should turn On and when you press 2, LED2 should turn Off.



```
test - HyperTerminal
File Edit View Call Transfer Help
[Icons]
Press 1 to turn on and 0 to turn off LED
1
Press 1 to turn on and 0 to turn off LED
0
Press 1 to turn on and 0 to turn off LED

Connected 0:10:51 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo
```

CAN command to turn On/Off LED3



Minimon V3 by IXXAT

File View Functions Options Help

IXXAT Interfaces

- USB-to-CAN compact
 - CAN 1: SJA 1000

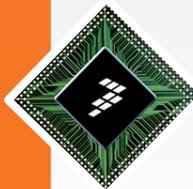
Time / 10 mSec	Identifier	Format	Flags	Data
00:00:06.77		1 Std	Self	01 00 00 00 00 00 00 00
00:00:06.77		2 Std		01 01 00 00 00 00 00 00
00:00:10.01		1 Std	Self	00 00 00 00 00 00 00 00
00:00:10.01		2 Std		01 01 00 00 00 00 00 00
00:00:11.88		1 Std	Self	12 00 00 00 00 00 00 00
00:00:11.88		2 Std		01 FF 00 00 00 00 00 00

Tx Identifier Ext. Rtr Data

1	12 00 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

Using a CAN communication tool like CANalyzer or IXXAT MiniMon,
Send CAN command ID = 1 and first data byte = 1 to turn on LED3
Send CAN command ID = 1 and first data byte = 0 to turn off LED3



Training Summary

- Overview of tools provided with Fast Start Kit for TRK-MPC5606B
- Utilized RAppID Init Tools for fast and easy infrastructure configuration and code generation
- Generated comprehensive report on the project using RAppID Init tool
- Utilized supporting tools provided with the kit to help build and flash code on to the target
- Described setting up and using the TRK-MPC5606B evaluation board

