



MEC140x/141x EVB Sample Code Guide

Rev 0.4

Date:12/22/2015

Copyright © 2015 Microchip or its subsidiaries. All rights reserved.

The information contained herein is confidential and proprietary to Microchip, shall be used solely in accordance with the agreement pursuant to which it is provided, and shall not be reproduced or disclosed to others without the prior written consent of Microchip. Although the information is believed to be accurate, no responsibility is assumed for inaccuracies. Microchip reserves the right to make changes to this document and to specifications and product descriptions at any time without notice. Neither the provision of this information nor the sale of the described semiconductor devices conveys any licenses under any patent rights or other intellectual property rights of Microchip or others. The product may contain design defects or errors known as anomalies, including but not necessarily limited to any which may be identified in this document, which may cause the product to deviate from published specifications. Microchip products are not designed, intended, authorized or warranted for use in any life support or other application where product failure could cause or contribute to personal injury or severe property damage. Any and all such uses without prior written approval of an officer of Microchip will be fully at the risk of the customer.

MICROCHIP DISCLAIMS AND EXCLUDES ANY AND ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND AGAINST INFRINGEMENT AND THE LIKE, AND ANY AND ALL WARRANTIES ARISING FROM ANY COURSE OF DEALING OR USAGE OF TRADE. IN NO EVENT SHALL MICROCHIP BE LIABLE FOR ANY DIRECT, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES; OR FOR LOST DATA, PROFITS, SAVINGS OR REVENUES OF ANY KIND; REGARDLESS OF THE FORM OF ACTION, WHETHER BASED ON CONTRACT; TORT; NEGLIGENCE OF MICROCHIP OR OTHERS; STRICT LIABILITY; BREACH OF WARRANTY; OR OTHERWISE; WHETHER OR NOT ANY REMEDY OF BUYER IS HELD TO HAVE FAILED OF ITS ESSENTIAL PURPOSE, AND WHETHER OR NOT MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

1 INTRODUCTION.....	5
1.1 AUDIENCE.....	5
1.2 REFERENCE DOCUMENTS.....	5
1.3 GLOSSARY OF TERMS AND ACRONYMS.....	5
2 FIRMWARE DESCRIPTION.....	6
2.1 FEATURES	6
2.2 SOFTWARE COPYRIGHT NOTICE.....	6
3 FIRMWARE/SYSTEM ARCHITECTURE	7
4 REQUIREMENT.....	8
4.1 HARDWARE REQUIREMENT.....	8
4.1.1 EVB Board configuration	8
4.1.2 MECC Card configuration.....	9
4.2 SOFTWARE REQUIREMENT	10
4.2.1 Build Tools/Versions	10
4.2.2 Environment Settings	10
5 BUILD PROCEDURE	11
5.1 BINARY FILE VERSION IDENTIFICATION	11
5.2 ON-SYSTEM FIRMWARE VERSION IDENTIFICATION	11
6 FUNCTION BLOCK DESCRIPTION	12
6.1 SAMPLE CODE BASE	12
6.2 UART	12
6.2.1 UART Configuration.....	13
6.2.2 UART SKERN Interface	13
6.2.3 UART test procedure	13
6.3 MUX AND GPIO	14
6.3.1 Mux and GPIO Configuration	14
6.3.2 Mux and GPIO SKERN Interface	15
6.3.3 Mux and GPIO test procedure.....	15
6.4 BASIC TIMER.....	15
6.4.1 Basic timer Configuration	15
6.4.2 Basic timer SKERN Interface	15
6.4.3 Basic timer test procedure.....	15
6.5 RTOS TIMER	16
6.5.1 RTOS timer Configuration	16
6.5.2 RTOS timer SKERN Interface	16
6.5.3 RTOS timer test procedure.....	16
6.6 HIBERNATION TIMER.....	17
6.6.1 Hibernation timer Configuration.....	17
6.6.2 Hibernation timer SKERN Interface.....	17
6.6.3 Hibernation timer test procedure	18
6.7 WATCHDOG TIMER	18
6.7.1 Watchdog timer Configuration	18
6.7.2 Watchdog timer SKERN Interface	19
6.7.3 Watchdog timer test procedure	19
6.8 nRESET_OUT	21
6.8.1 nRESET_OUT Configuration.....	21
6.8.2 nRESET_OUT SKERN Interface.....	21

6.8.3 nRESET_OUT test procedure	21
6.9 PS2	23
6.9.1 PS2 Configuration.....	23
6.9.2 PS2 SKERN Interface.....	24
6.9.3 PS2 test procedure	24
6.10 MATRIX KEYBOARD SCANNER.....	26
6.10.1 Matrix keyboard scanner Configuration.....	26
6.10.2 Matrix keyboard scanner SKERN Interface.....	26
6.10.3 Matrix keyboard scanner test procedure	26
6.11 SMBus	27
6.11.1 SMBus Configuration.....	27
6.11.2 SMBus SKERN Interface.....	28
6.11.3 SMBus test procedure	28
6.12 WEEK TIMER.....	28
6.12.1 Week Timer Configuration.....	28
6.12.2 Week timer SKERN Interface	29
6.12.3 Week Timer test procedure	29
6.13 VBAT-POWERED CONTROL INTERFACE	30
6.13.1 VBAT-powered control interface Configuration	30
6.13.2 VBAT-powered control interface SKERN Interface	31
6.13.3 VBAT-powered control interface test procedure	31
6.14 ADC AND DAC	31
6.14.1 ADC and DAC Configuration	31
6.14.2 ADC and DAC SKERN Interface	32
6.14.3 ADC and DAC test procedure	32
6.15 PWM & FAN TACH	33
6.15.1 PWM & Fan Tach Configuration.....	33
6.15.2 PWM & Fan Tach SKERN Interface.....	33
6.15.3 PWM & Fan Tach test procedure	33
6.16 BREATHING/BLINKING LED	35
6.16.1 Breathing/Blinking LED Configuration	35
6.16.2 Breathing/Blinking LED SKERN Interface	36
6.16.3 Breathing/Blinking LED test procedure	36
6.17 QUAD MODE SPI.....	36
6.17.1 QMSPI Configuration.....	36
6.17.2 QMSPI SKERN Interface.....	36
6.17.3 QMSPI test procedure	36
6.17.3.1 Single mode test	36
6.17.3.2 Quad mode test.....	39
6.18 BC-LINK	41
6.18.1 BC-Link Configuration	41
6.18.2 BC-Link SKERN Interface	41
6.18.3 BC_Link test procedure	41
6.19 SLEEP MODE	41
6.19.1 Sleep mode Configuration	41
6.19.2 Sleep mode SKERN Interface	42
6.19.3 Sleep mode test procedure	42
6.20 HOST INTERFACE EXERCISE	42
6.20.1 Host Interface configuration.....	43
6.20.2 Host Interface SKERN Interface.....	43
6.20.3 Host Interface test basic preparation.....	43
6.20.4 Test emulate 8042 interface	43
6.20.5 Test ACPI EC 0 interface – legacy.....	44
6.20.6 Test ACPI EC 0 interface – 4-byte mode	44
6.20.7 Test Port80 interface 0	44

MEC140x/141x EVB Sample Code Guide

6.20.8 Test Port80 interface 1	44
6.20.9 Test Mailbox interface	44
6.20.10 Test EMI interface.....	45
6.20.11 Test ChipMan utility	45
7 EXAMPLE FOR CODE AND DATA MEMORY RANGE RELOCATION.....	48
8 RELEASE HISTORY	49
9 TEST REPORT	51
10 KNOWN ISSUES	52
DOCUMENT REVISION HISTORY	53

1 Introduction

This document describes MEC140x/141x sample code architecture, usage and test methods. Most of items apply to standalone EVB board practice and some items may only apply to MECC card plus Intel CRB since it may work upon the DOS or Windows OS.

1.1 Audience

This document is written for developers who have a background in firmware development.

1.2 Reference Documents

- MEC140x_1x_DataSheet_2015-06-19.pdf
- MEC140x EVB Assy 6757 Rev A1p1 – SCH.pdf
- MEC140x 128VTQFP Solder DC Assy 6759 Rev A1p0 – SCH.pdf
- MEC140x LPC Modular EC Card - Assy_6761 Rev A1p1 – SCH.pdf
- ECE1105 DS Rev. 1.5 (01-27-09)
- ECE1099 DS Rev. 1.6 (02-21-11)

1.3 Glossary of Terms and Acronyms

This document contains the following terms, defined here for the purpose of convenience and general agreement:

Term	Description
EC	Embedded Controller
KBC	Keyboard Controller

2 Firmware Description

The sample code was developed based on Microchip skern kernel and all function blocks manipulation is following the skern operation.

2.1 Features

The SDK sample code contains the code for manipulating most of the MEC14xx functional block, most of the manipulation is upon the standalone EVB board, and Host Interface related functions were manipulated upon the MECC card and the Intel CRB.

The EVB manipulation items contain:

UART, Mux. & GPIO, Basic Timer, RTOS Timer, Hibernation Timer, Watchdog Timer, nReset_out, PS2, Matrix keyboard, SMBus, Week Timer, VBAT-Powered Control Interface, ADC & DAC, PWM & Fan tachometer, Blinking/Breathing LED, Quad-Mode SPI, BC-LINK, Power down mode(sleep).

MECC card manipulation items contain:

Emulated 8042, ACPI EC, Port 80, Mailbox, EMI and ChipMan utility exercise.

2.2 Software copyright notice

© [2014-2016] Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE TERMS.

3 Firmware/System architecture

```
\ SDK
|  mec1404_sdk
|  mec1406_sdk
|  mec1408_sdk
|  source
```

The code set contains 3 project settings for mec1404, mec1406 and mec1408, which share the same source code folder.

Under SOURCE folder, the basic codebase are as below,

```
\ source
|  include      - sample code include files
|  main         - C main function
|  skern        - skern kernel code
|  version      - sample code version control
|  ...
|  (other function modules...)
\
|  MEC140x.sdk
|  |  include      - SDK firmware stack include files
|  |  exception    - MIPS M14K exception files
|  |  proc         - chip processor
|  |  gpio        - chip GPIO
|  |  jtvic       - chip interrupt common module
|  |  jtvic_aggr   - interrupt - aggregated mode
|  |  jtvic_disaggr - interrupt - disaggregated mode
|  |  timer       - all timer collection
|  |  trace       - trace message
|  |  ...
|  |  (other function modules...)
```

The folder MEC140x.sdk contains the files which to be used as the Firmware Stack that can be easily transfer and porting to other project.

4 Requirement

4.1 Hardware requirement

1. Microchip MEC140x EVB Assy.6757
2. Microchip ECE1088, ECE1099 & ECE1105 EVB Assy.6484
3. Microchip Pegasus Debugger Assy.6345
4. Microchip MPLAB Pickit3 or ICD-3 or Real-ICE debugger
5. Fujitsu matrix keyboard (Part#: N860-7401-TOO1)
6. PS/2 mouse
7. PS/2 keyboard
8. Smart Battery
9. Dediprog SF-100 SPI flash programmer
10. Intel Broadwell (BDW Y LPDDR3 (Pearl Valley) CRB
11. Microchip MEC1408 MECC card Assy.6761

4.1.1 EVB Board configuration

for 5V AC Adapter configuration

name	action	purpose	memo
Power plane			
JP16	close	12V	
JP17	2-3 close	5V	
JP18	2-3 close	3.3V	
JP19	2-3 close	3.3V_STBY	
JP25	1-2 close	VBAT_chip	
JP26	1-2 close	1.8V_STBY	
JP28	1-2 close	VTR_33_18	
JP35	1-2 close		
JP44	1-2 close	GPIO pull-high V	
JP45	1-2 close		
JP62	2-3 close	Fan voltage	
Clock			On board OSC.
JP1 on DC card	Close		
JP3 on DC card	Open		
JP4 on Dc card	open		
JP3	2-3 close		
JP4	1-2 close		
JP5	Open		
JP6	Open		
TFDP			
JP53	close	Disable MAX3243	
JP8	1-2, 4-5 close		
JP55.1	Connect to TFDP board RESET pin		

JP13	17-18 short		
Shared SPI			
JP9	1-2 close	Shared SPI	CR_STRAP
JP39	All closed	GPIO mux.	
ICSP			
JP15	8-9 close 11-12 close	GPIO mux.	
Misc.			
JP23	1-2 close	LED0	
JP23	13-14 close	nRESET_OUT	
JP56	close	VCC_PWRGD	
JP11		comparator	CMP_STRAP
LPC			
JP2	close	nLRESET	

4.1.2 MECC Card configuration

name	Action	purpose	memo
Power plane			
JP1	2-3 close	VTR_33_18	
JP2	close	ADC_DAC_VREF	
JP4	1-2 close	VREF_CPU	
JP5	close	Chip's VTR	
JP6	2-3 close	VBAT selection	
JP8	close	V3.3_EC_CORE	
JP10	close	V3.3S	
JP11	close	V3.3A_RTC	
JP13	open	1.8V_eSPI	Reserved
ICSP			
JP14	Open	ICSP_MCLR	
TFDP			
JP7	1-3 close 2-4 close	TFDP	
Clock			
R11	short		
Misc.			
JP9		nRESET_IN	
		Comparator 0	Pull-high by R20
External power(option)			
J3.1 J3.4	VCC 3.3v GND	V3.3_EC_CORE	

4.2 Software requirement

Microchip Debug Trace FIFO Acquisizer Utility, DTF.EXE – a message collection and translation utility under Windows OS that used to receive the trace message from Microchip Pegasus debugger.

4.2.1 Build Tools/Versions

Development IDE: MPLAB X IDE v3.10 or later

Compiler/Linker: XC32 compiler set v1.40 or later

Note: It is NOT recommended to install the tool into the path contains the "space".

4.2.2 Environment Settings

No special setting required.

5 Build Procedure

Launch MPLAB X IDE, then open the corresponding project for MEC1404, MEC1406 or MEC1408 depends on the chip on the test board, and make sure all project files attribution are NOT the Read-Only.

- **Build project**
Click “Build Project” button or select “Build project” from drop-down menu will invoke the Compiling/Linking process, and finally generates a 16M bytes binary file located in “target” folder which can be updated to SPI flash via Dediprog SF-100.

To update binary file into SPI flash on EVB,

- Put the MEC140x in RESET state by connecting JP57 pin2 to ground.
- Connect SF-100 to J10.
- Power ON the board and launch Dediprog utility.

To update binary file into SPI flash on MEC1408 MECC,

- Put the MEC140x in RESET state by close JP9.
- Connect SF-100 to J2.
- Power ON the board and launch Dediprog utility.

NOTE:

When the chip is not powered, the SPI pins will be clamped to <1V by the ESD protection logic. So to powering-up the chip and hold the chip in RESET by lowing nRESET_IN to be able to program the SPI FLASH from an external device.

- **Debug Project**
Click “Debug Project” button or select “Debug project” from drop-down menu will invoke the Compiling process, and finally generates an ELF file that will be downloaded into MEC140x SRAM via MPLAB debugger (i.e. Pickit3/ICD-3/Real-ICE).

The download process is performed automatically after the successes in compiling process, so make sure the board is powered and MPLAB debugger is connect to J35 on EVB (or J1 on MEC1408 MECC) correctly.

5.1 Binary file version identification

TBD.

5.2 On-system firmware version identification

TBD.

6 Function Block Description

6.1 Sample Code Base

The sample code base is the smallest runnable code which contains most necessary files and modules. The most necessary files can refer to chapter 3 - Firmware/System architecture.

The sample code control valve which consists of sample code base configuration and other individual functional module compiler switches are defined in Config.h

Below are sample code base features,

Scheduler:

Skern scheduler

Basic function contains:

Basic timer 0 used as the 1ms periodic time ticker for Skern scheduler

Basic timer 1 used as the 100us one-shot timer can be used for special application

Basic timer 2 used as the micro-second delay timer can be used for precise delay application

Trace FIFO debugger default is enabled

GPIO157/LED0 will be toggled per second

JTVIC aggregated mode is selected

Clock:

Clocked by external single-ended 32 KHz input

Version control:

version.c, version.h

Firmware:

Config.h –

BOARD_TYPE	= EVB or MECC
PROC_CLK_DIVIDE	= Processor clock divider
TRACE_ENABLE	= enable trace message output
DEF_32K_CLOCK_CONTROL	= determine 32Khz clock domain
OSC_48M_CLK_REF	= determine 48Mhz calibration input clock
PACKAGE_128	= determine package, here is always 1 only.

6.2 UART

The sample code demonstrates the TXD/RXD bi-direction communication between host PC and EC EVB.

Member file: uart_app.c, uart_app.h, uart.c, uart.h

6.2.1 UART Configuration

EVB:

- Make sure external clock is configured well, either 32Khz single-ended or crystal
- TXD/RXD - JP8 1-2 short, 4-5 short
- Max3243 – JP53 open
- No trace FIFO - J30.9, J30.11 open

Firmware:

- Config.h – TRACE_ENABLE = 0, UART_EN=1
- Cfg.h – UART task should be enabled
- gpioCNFG.h –
GPIO116/117 – MUX = 2, Direction = OUTPUT, Interrupt detection = 4, others remain 0

PC Serial Terminal: (for example TeraTerm as shown in our setup below)

Baud rate: 115200bps

Parity checking: none

Data bits: 8

Stop bits: 1

Flow control: none

6.2.2 UART SKERN Interface

Interrupt task: Yes, receives commands from the host PC

Event task: none

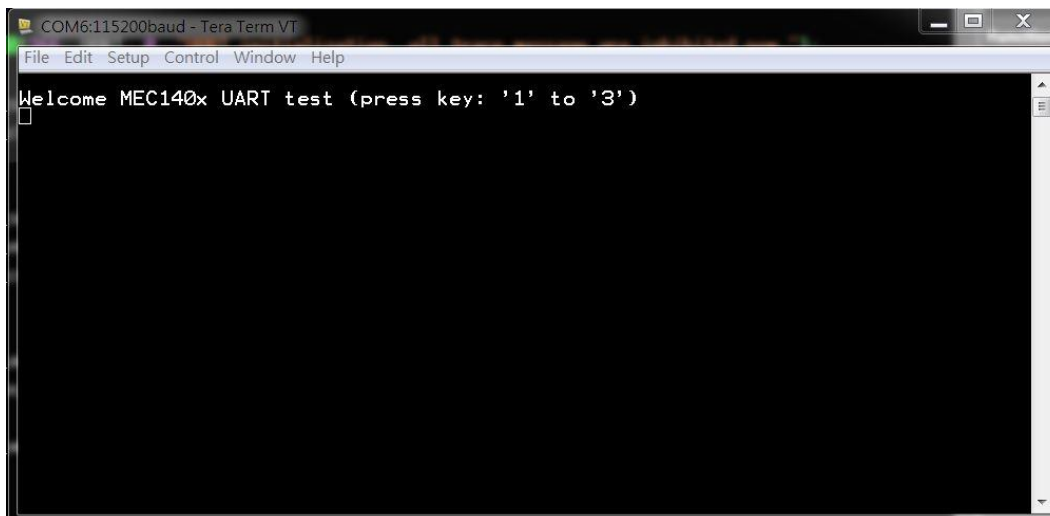
Timer task: none

6.2.3 UART test procedure

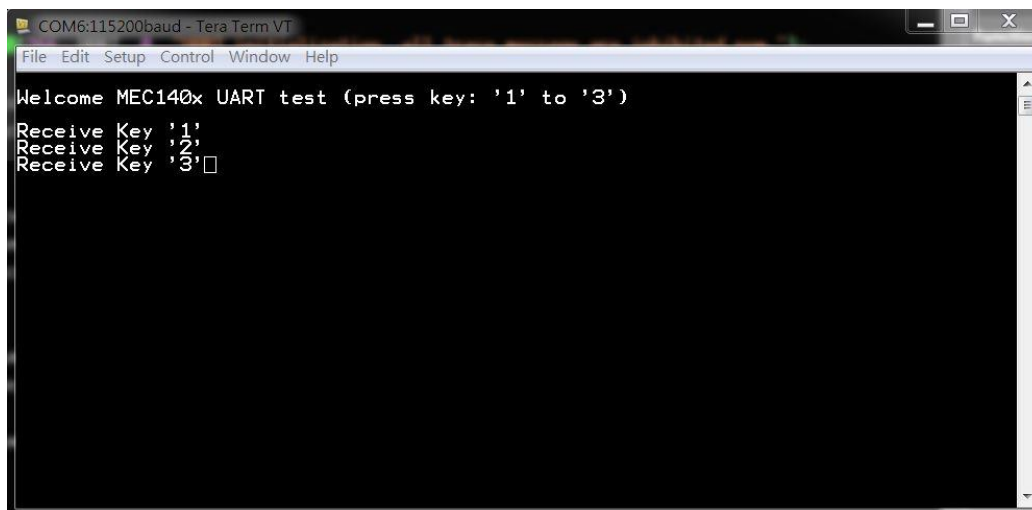
Connect host PC RS-232 serial port with EC EVB via Null modem cable.

Note: If host PC serial port is not available, a USB-RS232 converter board can be used, for example Micorchip's MCP2200EV-VCP demo board.

After power on the EVB, the welcome message is show on PC as below,



After type key '1' or '2' or '3', then receive the message responded from EC EVB as below,



6.3 Mux and GPIO

The Multiplexing and GPIO has been implemented into the kernel as the base code, and the GPIO157 is multiplexed with LED0.

Member file: gpio.c, gpio.h, gpioCNFG.h

6.3.1 Mux and GPIO Configuration

EVB:

- GPIO157/LED0 - JP23 1-2 short

Firmware:

- No extra setting

6.3.2 Mux and GPIO SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.3.3 Mux and GPIO test procedure

Power on the EVB, the GPIO157/LED0 would be toggled per second.

6.4 Basic Timer

The basic timer 0 has been used as the codebase 1ms time ticker, the bTimer.c contains the basic setting while initial stage and kTimer.c contains the application for 1ms and 1s process.

Member file: btimer.c, btimer.h

6.4.1 Basic timer Configuration

EVB:

- LED0 - JP23 1-2 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short

Firmware:

- Config.h - TRACE_ENABLE=1, UART_EN=0

6.4.2 Basic timer SKERN Interface

Interrupt task: Yes

Event task: none

Timer task: none

6.4.3 Basic timer test procedure

Power on the EVB, the trace message would be sent out per second and the LED0 would be blinking at the rate of 0.5Hz.

6.5 RTOS Timer

Member file: rtostimer_app.c, rtostimer_app.h, RTOSTimer.c, RTOSTimer.h

6.5.1 RTOS timer Configuration

EVB:

- LED0 - JP23 1-2 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short

Firmware:

- Config.h - TRACE_ENABLE=1, UART_EN=0, RTOS_TIMER_EN=1

6.5.2 RTOS timer SKERN Interface

Interrupt task: Yes

Event task: none

Timer task: none

6.5.3 RTOS timer test procedure

The RTOS timer has been set to generate the trace message every 5s.

Powers on the EVB, the trace message “RTOS_Timer Interrupt” are generated by RTOS 5s timer. The message may look like as below,

Trace Capture Document

Trace Point Data. Acquisition began Monday, March 30th, 2015 4:05:51 pm

```

0: INIT *****
1: INIT Trace FIFO Debug Port initialized
2: INIT *****
3: PROC proc_init: VTR_RST just been DONE >>>>
4: PROC +-----+
5: PROC | Turn ON Switched Ext. single-end or Int1. ON when VTR=0 |
6: PROC +-----+
7: PROC proc_init: =====
8: PROC proc_init: PCR - 32.768KHz EXTERNAL clock input detected !!
9: PROC proc_init: =====
10: PROC proc_init: =====
11: PROC proc_init: VBAT - 32.768KHz External clock input detected !!
12: PROC proc_init: =====
13: PROC proc_init: =====
14: PROC proc_init: Internal 32.768KHz OSC. ready !!
15: PROC proc_init: =====
16: PROC proc_init: =====
17: PROC proc_init: Internal 48MHz OSC. Lock !!
18: PROC proc_init: =====
19: PROC proc_init: DONE
20: GPIO gpio_init: DONE
21: JTVIC_AG JTVIC_init: aggregate mode DONE
22: BTIMER BTimer_Init: DONE
23: INIT Initialization done.
24: KERNEL main : Init Task 0
25: KERNEL main : Init Task 1
26: KERNEL main : Init Task 2
27: RTOS RTOS_APP initialization DONE
28: TIMER # 1-second tick!
29: TIMER # 1-second tick!
30: TIMER # 1-second tick!
31: TIMER # 1-second tick!
32: RTOS [ RTOS_Timer Interrupt ]: #####
33: TIMER # 1-second tick!
34: TIMER # 1-second tick!
35: TIMER # 1-second tick!
36: TIMER # 1-second tick!
37: TIMER # 1-second tick!
38: RTOS [ RTOS_Timer Interrupt ]: #####
39: TIMER # 1-second tick!
40: TIMER # 1-second tick!
41: TIMER # 1-second tick!
42: TIMER # 1-second tick!
43: TIMER # 1-second tick!
44: RTOS [ RTOS_Timer Interrupt ]: #####

```

6.6 Hibernation Timer

Member file: htimer_app.c, htimer_app.h, htimer.c, htimer.h

6.6.1 Hibernation timer Configuration

EVB:

- LED0 - JP23 1-2 short
- TFD_P_CLK/TFD_P_DATA – JP8 1-2 short, 4-5 short

Firmware:

- Config.h - TRACE_ENABLE=1, UART_EN=0, HTIMER_EN=1

6.6.2 Hibernation timer SKERN Interface

Interrupt task: Yes

Event task: none

Timer task: none

6.6.3 Hibernation timer test procedure

The Hibernation timer has been set to generate the trace message in 5s after initialization process.

Powers on the EVB, the trace message “H_Timer Interrupt” is generated by RTOS 5s timer. The message may like as below,

```

30: INIT      Initialization done.
31: KERNEL    main : Init Task 0
32: KERNEL    main : Init Task 1
33: KERNEL    main : Init Task 2
34: KERNEL    main : Init Task 3
35: RTOS      Htimer_app initialization DONE
36: KERNEL    main : Init Task 4
37: TIMER     # 1-second tick!
38: TIMER     # 1-second tick!
39: TIMER     # 1-second tick!
40: TIMER     # 1-second tick!
41: RTOS      [ H_Timer Interrupt ]: #####
42: TIMER     # 1-second tick!
43: TIMER     # 1-second tick!
44: TIMER     # 1-second tick!
45: TIMER     # 1-second tick!
46: TIMER     # 1-second tick!
47: TIMER     # 1-second tick!

```

6.7 Watchdog Timer

Member file: wdtimer_app.c, wdtimer_app.h, wdtimer.c, wdtimer.h

6.7.1 Watchdog timer Configuration

EVB:

- LED0 - JP23 1-2 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short

Firmware:

- Config.h - TRACE_ENABLE=1, UART_EN=0, WDTIMER_EN=1

6.7.2 Watchdog timer SKERN Interface

Interrupt task: none

Event task: none

Timer task: Yes

6.7.3 Watchdog timer test procedure

The Watchdog timer has been set to 1007ms, and the periodic reload value setting in WDTIMER_TMR_EVENT for every 900ms and 1010ms are implemented separately for success and fail case as below.

This test can only be done upon SPI ROM, and it may not perform the success watchdog reset process while utilize the debugger.

1. Modify the value to 900 in below function where in both wdtimer_init_task() and wdtimer_tmr_proc(),

```
kTaskSetWakeTime(900, wdtimer_TIMER_ID);
```

Updated the SPI and powers on the EVB, the trace message "WDTimer Reload" can be seen in timely updated the watchdog timer before the reset. The message may like as below,

```

32: INIT      Initialization done.
33: KERNEL   main : Init Task 0
34: KERNEL   main : Init Task 1
35: KERNEL   main : Init Task 2
36: KERNEL   main : Init Task 3
37: KERNEL   main : Init Task 4
38: WDT       wdt_activate
39: KERNEL   Set task 4 TIMER = 900 ticks
40: WDT       wdtimer_app activate...
41: KERNEL   Set TIMER flag for task 4
42: KERNEL   Set task 4 TIMER = 900 ticks
43: WDTMR     [ WDTimer Reload ]: !!!
44: TIMER     # 1-second tick!
45: KERNEL   Set TIMER flag for task 4
46: KERNEL   Set task 4 TIMER = 900 ticks
47: WDTMR     [ WDTimer Reload ]: !!!
48: TIMER     # 1-second tick!
49: KERNEL   Set TIMER flag for task 4
50: KERNEL   Set task 4 TIMER = 900 ticks
51: WDTMR     [ WDTimer Reload ]: !!!
52: TIMER     # 1-second tick!
53: KERNEL   Set TIMER flag for task 4
54: KERNEL   Set task 4 TIMER = 900 ticks
55: WDTMR     [ WDTimer Reload ]: !!!
56: TIMER     # 1-second tick!
57: KERNEL   Set TIMER flag for task 4
58: KERNEL   Set task 4 TIMER = 900 ticks
59: WDTMR     [ WDTimer Reload ]: !!!
60: TIMER     # 1-second tick!

```

2. Modify the value to 1010 in below function where in both `wdtimer_init_task()` and `wdtimer_tmr_proc()`,

```
kTaskSetWakeTime(1010, wdtimer_TIMER_ID);
```

Updated the SPI and powers on the EVB, after watchdog timer is activated, the "WDTimer Reload" is not quick enough to reload the timer, so that the VTR reset is invoked.

```

1238: INIT      Initialization done.
1239: KERNEL    main : Init Task 0
1240: KERNEL    main : Init Task 1
1241: KERNEL    main : Init Task 2
1242: KERNEL    main : Init Task 3
1243: KERNEL    main : Init Task 4
1244: WDT       wdt_activate
1245: KERNEL    Set task 4 TIMER = 1010 ticks
1246: WDT       wdtimer_app activate...
1247: TIMER     # 1-second tick!
1248: ROMI      BootROM ON_RESET
1249: ROMM      ROM Cxx Main
1250: ROMM      Addr(rom_log first word) = 0xbfd1ff00
1251: ROMM      Addr(rom_log last word)  = 0xbfd1fffc
1252: ROMM      ROM QA Mode Key CRC32 = 0xe67982d4
1253: ROMT      Apply Trim
1254: ROMT      eFUSE Configured bit set
1255: ROMT      SRAM Size Field = 0x02
1256: ROMT      Write 0x02 to 32K OSC Trim Register

```

6.8 nRESET_OUT

Member file: lpc.c, lpc.h, lpc_app.c, lpc_app.h

6.8.1 nRESET_OUT Configuration

EVB:

- LED0 - JP23 13-14 short
- VCC_PWRGD – JP56 closed
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- nLRESET – JP2 closed

Firmware:

- Config.h - TRACE_ENABLE=1, UART_EN=0, LPC_EN=1

6.8.2 nRESET_OUT SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.8.3 nRESET_OUT test procedure

Press and hold the button S2, then power on the EVB, this is used to hold the VCC_PWRGD in low, at the time the LED10 should be ON, because the nRESET_OUT is LOW, the trace message may like as below,


```

0: INIT *****
1: INIT Trace FIFO Debug Port initialized
2: INIT *****
3: PROC =====
4: PROC Normal POR...
5: PROC =====
6: PROC proc_init: VTR_RST just been DONE >>>>
7: PROC +-----+
8: PROC | Turn ON Switched Ext. single-end or Intl. ON when VTR=0 |
9: PROC +-----+
10: PROC proc_init: =====
11: PROC proc_init: PCR - 32.768KHz EXTERNAL clock input detected !!
12: PROC proc_init: =====
13: PROC proc_init: =====
14: PROC proc_init: VBAT - 32.768KHz External clock input detected !!
15: PROC proc_init: =====
16: PROC proc_init: =====
17: PROC proc_init: Internal 32.768KHz OSC. ready !!
18: PROC proc_init: =====
19: PROC proc_init: =====
20: PROC proc_init: Internal 48MHz OSC. Lock !!
21: PROC proc_init: =====
22: PROC proc_init: DONE
23: GPIO gpio_init: DONE
24: JTVIC_AG JTVIC_init: aggregate mode DONE
25: BTIMER BTimer_Init: DONE
26: AP3M M14K CP0.STATUS = 0x00000000
27: AP3M M14K CP0.CAUSE = 0x00800000
28: AP3M M14K CP0.CONFIG = 0xa4210582
29: AP3M M14K CP0.CONFIG3 = 0x8023e060
30: AP3M M14K CP0.EBASE = 0xbfd00000
31: AP3M M14K CP0.IntCtl = 0x00000020
32: AP3M M14K CP0.IntCtl(new) = 0x00400020
33: INIT Initialization done.
34: KERNEL main : Init Task 0
35: KERNEL main : Init Task 1
36: KERNEL main : Init Task 2
37: KERNEL main : Init Task 3
38: KERNEL main : Init Task 4
39: KERNEL main : Init Task 5

```

After release the button S2, then the LED10 should be OFF, because the nRESRT_OUT is HIGH, the trace message shows the code is keep running may like as below,

```

 9: PROC      +-----+
10: PROC      proc_init: =====
11: PROC      proc_init: PCR - 32.768KHz EXTERNAL clock input detected !!
12: PROC      proc_init: =====
13: PROC      proc_init: =====
14: PROC      proc_init: VBAT - 32.768KHz External clock input detected !!
15: PROC      proc_init: =====
16: PROC      proc_init: =====
17: PROC      proc_init: Internal 32.768KHz OSC. ready !!
18: PROC      proc_init: =====
19: PROC      proc_init: =====
20: PROC      proc_init: Internal 48MHz OSC. Lock !!
21: PROC      proc_init: =====
22: PROC      proc_init: DONE
23: GPIO      gpio_init: DONE
24: JTVIC_AG  JTVIC_init: aggregate mode DONE
25: BTIMER    BTimer_Init: DONE
26: AP3M      M14K CP0.STATUS      = 0x00000000
27: AP3M      M14K CP0.CAUSE       = 0x00800000
28: AP3M      M14K CP0.CONFIG      = 0xa4210582
29: AP3M      M14K CP0.CONFIG3     = 0x8023e060
30: AP3M      M14K CP0.EBASE       = 0xbfd00000
31: AP3M      M14K CP0.IntCtl      = 0x00000020
32: AP3M      M14K CP0.IntCtl(new) = 0x00400020
33: INIT      Initialization done.
34: KERNEL    main : Init Task 0
35: KERNEL    main : Init Task 1
36: KERNEL    main : Init Task 2
37: KERNEL    main : Init Task 3
38: KERNEL    main : Init Task 4
39: KERNEL    main : Init Task 5
40: LPC       lpc: Get VCC_PWRGD asserted !!!
41: LPC       lpc: Get LRESET asserted !!!
42: LPC       lpc_init: DONE
43: KERNEL    main : Init Task 6
44: PROC      ps2_init: DONE
45: TIMER     # 1-second tick!
46: TIMER     # 1-second tick!
47: TIMER     # 1-second tick!
48: TIMER     # 1-second tick!

```

6.9 PS2

Member file: ps2.c, ps2.h, ps2_app.c, ps2_app.h

6.9.1 PS2 Configuration

EVB:

- LED0 - JP23 13-14 short
- VCC_PWRGD – JP56 closed
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- nLRESET – JP2 closed

Firmware:

- Config.h - TRACE_ENABLE=1, UART_EN=0, LPC_EN=1, PS2_EN=1 (other setting under PS2 are disabled)

6.9.2 PS2 SKERN Interface

Interrupt task: yes

Event task: yes

Timer task: yes

6.9.3 PS2 test procedure

1. Power on the EVB, then plug-in Ps2 mouse into Port 0.
2. The firmware will receive the PS2 POR (0xAA and 0x00), then issuing the Enable command(0xF4) to PS2 mouse and get the ACK(0xFA)
3. Click the mouse button or move the mouse will get the corresponding data.


```

93: TIMER      # 1-second tick! VBAT Good!
94: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = AAh <<
95: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
96: PS2        Send_To_Ps2_Port - port:0, SEND = F4h >>
97: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = FAh <<
98: TIMER      # 1-second tick! VBAT Good!
99: TIMER      # 1-second tick! VBAT Good!
100: TIMER     # 1-second tick! VBAT Good!
101: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 09h <<
102: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
103: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
104: TIMER      # 1-second tick! VBAT Good!
105: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 08h <<
106: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
107: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
108: TIMER      # 1-second tick! VBAT Good!
109: TIMER      # 1-second tick! VBAT Good!
110: TIMER      # 1-second tick! VBAT Good!
111: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 0Ah <<
112: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
113: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
114: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 08h <<
115: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
116: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
117: TIMER      # 1-second tick! VBAT Good!
118: TIMER      # 1-second tick! VBAT Good!
119: TIMER      # 1-second tick! VBAT Good!
120: TIMER      # 1-second tick! VBAT Good!
121: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 08h <<
122: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 02h <<
123: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
124: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 08h <<
125: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 01h <<
126: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
127: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 08h <<
128: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 01h <<
129: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
130: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 08h <<
131: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 01h <<
132: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<
133: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 08h <<
134: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 03h <<
135: PS2_IRQ    ISR_Ps2_Port_0: PS2 RECEIVE = 00h <<

```

6.10 Matrix keyboard scanner

Member file: keyscan.c, keyscan.h, keyscan_app.c, keyscan_app.h

6.10.1 Matrix keyboard scanner Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- KSI – JP8 8-9, 11-12, 14-15, 17-18, 20-21, 23-24 short
(if needs pull-up, then JP54 all closed)
- KSO – JP12 2-3, 5-6, 8-9, 11-12, 14-15, 17-18 short

Firmware:

- Config.h - TRACE_ENABLE=1, UART_EN=0, KEYSCAN=1

6.10.2 Matrix keyboard scanner SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.10.3 Matrix keyboard scanner test procedure

Connect a membrane keyboard to EVB, then power on the EVB, hit any key then check the trace message.

```

63: INIT      Initialization done.
64: KERNEL    main : Init Task 0
65: KERNEL    main : Init Task 1
66: KERNEL    main : Init Task 2
67: KERNEL    main : Init Task 3
68: KERNEL    main : Init Task 4
69: KERNEL    main : Init Task 5
70: KERNEL    main : Init Task 6
71: KERNEL    main : Init Task 7
72: KEYS SCAN  Scanner: initial
73: KEYS SCAN  Scanner: KEYS SCAN->KSI_INPUT FFh
74: KEYS SCAN  Scanner: KEYS SCAN->KSI_STATUS 00h
75: KEYS SCAN  Scanner: JTVIC_GIRQ->REGS[GIRQ17_ID].SOURCE bit6 00000000h
76: KEYS SCAN  Scanner: JTVIC_GIRQ->REGS[GIRQ17_ID].EN_SET bit6 00000040h
77: KEYS SCAN  Scanner: Wait_KeyScan_Irq =>NO key pressed...
78: KEYS SCAN  keyscan_init: DONE
79: KERNEL    main : Init Task 8
80: KERNEL    main : Init Task 9
81: KERNEL    main : Init Task 10
82: KERNEL    main : Init Task 11
83: KERNEL    main : Init Task 12
84: KERNEL    main : Init Task 13
85: KERNEL    main : Init Task 14
86: KERNEL    main : Init Task 15
87: KERNEL    main : Init Task 16
88: TIMER     # 1-second tick! VBAT Good!
89: TIMER     # 1-second tick! VBAT Good!
90: TIMER     # 1-second tick! VBAT Good!
91: TIMER     # 1-second tick! VBAT Good!
92: TIMER     # 1-second tick! VBAT Good!
93: INIT      ISR_keyscan: INTR#####
94: TIMER     # 1-second tick! VBAT Good!
95: INIT      ISR_keyscan: INTR#####
96: TIMER     # 1-second tick! VBAT Good!
97: TIMER     # 1-second tick! VBAT Good!

```

6.11 SMBus

Member file: smb_app.c, smb_app.h, smb_comm.c, smb_comm.h, smb_0.c, smb_0.h

6.11.1 SMBus Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- SMBus port 01 on (GPIO007,GPIO010)

Firmware:

- Config.h - TRACE_ENABLE=1, SMB_EN=1

6.11.2 SMBus SKERN Interface

Interrupt task: none

Event task: none

Timer task: yes

6.11.3 SMBus test procedure

Connect the Smart Battery to EVB SMBus port 01(CLK, DATA and GND).
Power on the EVB then the trace message as below,

```

74: KERNEL    main : Init Task 7
75: KERNEL    main : Init Task 8
76: SMB_0     SMB_0: Set slave receive mode
77: SMB_0     smb_init_task: SMB_0 nBB sync
78: SMB       smb_init_task: DONE
79: TIMER     # 1-second tick! VBAT Good!
80: SMB_0     SMB_0: gen START
81: SMB       smb_init_task: send addr: 16h
82: SMB       smb_init_task: sts: 00h
83: SMB       smb_init_task: send cmd: 0Dh
84: SMB       smb_init_task: sts: 00h
85: SMB_0     SMB_0: gen RepSTART
86: SMB       smb_init_task: send addr: 17h
87: SMB_0     SMB_0: gen NACK
88: SMB_0     SMB_0: gen STOP
89: SMB       smb_init_task: remaing capacity - 85%
90: TIMER     # 1-second tick! VBAT Good!
91: TIMER     # 1-second tick! VBAT Good!
92: TIMER     # 1-second tick! VBAT Good!

```

6.12 Week Timer

Member file: wktimer_app.c, wktimer_app.h, wktimer.c, wktimer.h

6.12.1 Week Timer Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDG_CLK/TFDG_DATA – JP8 1-2 short, 4-5 short
- SYSPWR_PRESS JP10 4-5 short

Firmware:

- Config.h - TRACE_ENABLE=1, WKTIMER_EN=1, SUB_WK_TIMER_TICK can be 1 or 4

6.12.2 Week timer SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.12.3 Week Timer test procedure

Set SUB_WK_TIMER_TICK to 1 and power on the evb,

- #87, 89... - The sub-second interrupt would be triggered every 500ms
- #88, 92... - The One-second interrupt would be triggered every 1 second
- #98 - The sub-Week timer would be triggered every 3 second
- #125 - The Week Timer would be triggered every 10 second


```

86: WKTMR      Week Timer initialization DONE
87: WKTMR      ISR - sub second#####
88: WKTMR      ISR - One second#####
89: WKTMR      ISR - sub second#####
90: TIMER      # 1-second tick! VBAT Good!
91: WKTMR      ISR - sub second#####
92: WKTMR      ISR - One second#####
93: WKTMR      ISR - sub second#####
94: TIMER      # 1-second tick! VBAT Good!
95: WKTMR      ISR - sub second#####
96: WKTMR      ISR - One second#####
97: WKTMR      ISR - sub second#####
98: WKTMR      ISR - sub-Week Timer#####
99: TIMER      # 1-second tick! VBAT Good!
100: WKTMR     ISR - sub second#####
101: WKTMR     ISR - One second#####
102: WKTMR     ISR - sub second#####
103: TIMER     # 1-second tick! VBAT Good!
104: WKTMR     ISR - sub second#####
105: WKTMR     ISR - One second#####
106: WKTMR     ISR - sub second#####
107: TIMER     # 1-second tick! VBAT Good!
108: WKTMR     ISR - sub second#####
109: WKTMR     ISR - One second#####
110: WKTMR     ISR - sub second#####
111: TIMER     # 1-second tick! VBAT Good!
112: WKTMR     ISR - sub second#####
113: WKTMR     ISR - One second#####
114: WKTMR     ISR - sub second#####
115: TIMER     # 1-second tick! VBAT Good!
116: WKTMR     ISR - sub second#####
117: WKTMR     ISR - One second#####
118: WKTMR     ISR - sub second#####
119: TIMER     # 1-second tick! VBAT Good!
120: WKTMR     ISR - sub second#####
121: WKTMR     ISR - One second#####
122: WKTMR     ISR - sub second#####
123: TIMER     # 1-second tick! VBAT Good!
124: WKTMR     ISR - sub second#####
125: WKTMR     ISR - Week Timer#####
126: WKTMR     ISR - One second#####

```

6.13 VBAT-powered control interface

Member file: vci_app.c, vci_app.h, vci.c, vci.h

6.13.1 VBAT-powered control interface Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- JP10 7-8 short, 10-11 short, 13-14 short, 16-17 short
- JP58 1-2 short, 3-4 short
- JP59 3-4 short

Firmware:

- Config.h - TRACE_ENABLE=1, VCI_EN=1

6.13.2 VBAT-powered control interface SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.13.3 VBAT-powered control interface test procedure

1. Hardware feature test,
Insert coin-cell battery into BT1, the LED13 would be OFF
 - a. press button S4 or S5 then LED13 should become ON
 - b. VCI_OVRD_IN by open JP59 pin3-4 then LED13 should become ON
2. Firmware feature test
Most of VCI feature are powered by VBAT and hardware controlled except the PWRUP-EVENT, so here example the effect of this event.

Power-up EVB then the LED13 should be OFF and then ON after 5 second.

6.14 ADC and DAC

Member file: adcdac_app.c, adcdac_app.h, adc.c, adc.h, dac.c, dac.h

6.14.1 ADC and DAC Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- DAC0 - JP13 7-8 short, 10-11 short
- DAC_VREF – JP30 4-5 short
- ADC_VREF – JP29 4-5 short
- DAC0 to ADC0 Routing - JP36 – 1-2 short, 3-4 short
- ADC0 – JP7 13-14 short, 10-11 short

Firmware:

- Config.h - TRACE_ENABLE=1, ADCDAC_EN=1, DAC_EN=1, ADC_EN=1

6.14.2 ADC and DAC SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.14.3 ADC and DAC test procedure

Since the DAC0 is routed to ADC0 for single conversion and DAC1 is routed to ADC1 for repeat conversion, so power on the evb, the trace message may like as below,

The test firmware first starts Repeat conversion with Zero delay, so the ADC1 reading would be performed prior to ADC0.

```

75: KERNEL      main : Init Task 11
76: ADCDAC      ADCDAC initialization DONE
77: ADCDAC      ADCDAC - ADC1 reading: 40
78: ADCDAC      ADCDAC - ADC0 reading: 501
79: ADCDAC      ADCDAC - ADC1 reading: 19
80: TIMER       # 1-second tick! VBAT Good!
81: ADCDAC      ADCDAC - ADC1 reading: 49
82: TIMER       # 1-second tick! VBAT Good!
83: ADCDAC      ADCDAC - ADC1 reading: 76
84: TIMER       # 1-second tick! VBAT Good!
85: ADCDAC      ADCDAC - ADC1 reading: 101
86: TIMER       # 1-second tick! VBAT Good!
87: ADCDAC      ADCDAC - ADC1 reading: 126
88: TIMER       # 1-second tick! VBAT Good!
89: ADCDAC      ADCDAC - ADC1 reading: 151
90: TIMER       # 1-second tick! VBAT Good!
91: ADCDAC      ADCDAC - ADC1 reading: 176
92: TIMER       # 1-second tick! VBAT Good!
93: ADCDAC      ADCDAC - ADC1 reading: 201
94: TIMER       # 1-second tick! VBAT Good!
95: ADCDAC      ADCDAC - ADC1 reading: 226
96: TIMER       # 1-second tick! VBAT Good!
97: ADCDAC      ADCDAC - ADC1 reading: 252
98: TIMER       # 1-second tick! VBAT Good!
99: ADCDAC      ADCDAC - ADC1 reading: 275
100: TIMER      # 1-second tick! VBAT Good!
101: ADCDAC      ADCDAC - ADC1 reading: 301
102: TIMER      # 1-second tick! VBAT Good!
103: ADCDAC      ADCDAC - ADC1 reading: 325
104: TIMER      # 1-second tick! VBAT Good!
105: ADCDAC      ADCDAC - ADC1 reading: 350
106: TIMER      # 1-second tick! VBAT Good!
107: ADCDAC      ADCDAC - ADC1 reading: 375

```


6.15 PWM & Fan Tach

Member file: pwmtach_app.c, pwmtach_app.h, pwm.c, pwm.h, tach.c, tach.h

6.15.1 PWM & Fan Tach Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- V-FAN – JP62 2-3 short for +5V test fan
- TACH channel selection JP60 1-2 short

Firmware:

- Config.h - TRACE_ENABLE=1, PWMTACH_EN=1, PWM_EN=1, TACH_EN=1, TACH_METHOD= 0 or 1

6.15.2 PWM & Fan Tach SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.15.3 PWM & Fan Tach test procedure

1. Set TACH_METHOD = 0

The sample code increases the PWM duty cycle from 0% to 100% in the unit of 10% every 3 second.

Also read fan tachometer reading count and calculate a “rough” RPM.

```

108: PWMTACH  TACH0 reading - 752
109: PWMTACH  Method 0 - RPM is 2420
110: PWMTACH  set new PWM duty cycle - 50%
111: TIMER    # 1-second tick! VBAT Good!
112: TIMER    # 1-second tick! VBAT Good!
113: TIMER    # 1-second tick! VBAT Good!
114: PWMTACH  TACH0 reading - 1028
115: PWMTACH  Method 0 - RPM is 2760
116: PWMTACH  set new PWM duty cycle - 60%
117: TIMER    # 1-second tick! VBAT Good!
118: TIMER    # 1-second tick! VBAT Good!
119: TIMER    # 1-second tick! VBAT Good!
120: PWMTACH  TACH0 reading - 1355
121: PWMTACH  Method 0 - RPM is 3270
122: PWMTACH  set new PWM duty cycle - 70%
123: TIMER    # 1-second tick! VBAT Good!
124: TIMER    # 1-second tick! VBAT Good!
125: TIMER    # 1-second tick! VBAT Good!
126: PWMTACH  TACH0 reading - 1703
127: PWMTACH  Method 0 - RPM is 3480
128: PWMTACH  set new PWM duty cycle - 80%
129: TIMER    # 1-second tick! VBAT Good!
130: TIMER    # 1-second tick! VBAT Good!
131: TIMER    # 1-second tick! VBAT Good!
132: PWMTACH  TACH0 reading - 2074
133: PWMTACH  Method 0 - RPM is 3710
134: PWMTACH  set new PWM duty cycle - 90%
135: TIMER    # 1-second tick! VBAT Good!
136: TIMER    # 1-second tick! VBAT Good!
137: TIMER    # 1-second tick! VBAT Good!
138: PWMTACH  TACH0 reading - 2459
139: PWMTACH  Method 0 - RPM is 3850
140: PWMTACH  set new PWM duty cycle - 100%

```

2. Set TACH_METHOD = 1

The sample code increases the PWM duty cycle from 10% to 100% in the unit of 10% every 10 revolutions.

Also read fan tachometer reading count and calculate a “rough” RPM.

```

123: PWMTACH  set new PWM duty cycle - 10%
124: PWMTACH  TACH0 reading for INTR_REV revolution - 50049
125: PWMTACH  Method 1 - RPM is 1198
126: PWMTACH  set new PWM duty cycle - 20%
127: TIMER    # 1-second tick! VBAT Good!
128: PWMTACH  TACH0 reading for INTR_REV revolution - 47410
129: PWMTACH  Method 1 - RPM is 1265
130: PWMTACH  set new PWM duty cycle - 30%
131: PWMTACH  TACH0 reading for INTR_REV revolution - 38023
132: PWMTACH  Method 1 - RPM is 1577
133: PWMTACH  set new PWM duty cycle - 40%
134: TIMER    # 1-second tick! VBAT Good!
135: PWMTACH  TACH0 reading for INTR_REV revolution - 31023
136: PWMTACH  Method 1 - RPM is 1934
137: PWMTACH  set new PWM duty cycle - 50%
138: PWMTACH  TACH0 reading for INTR_REV revolution - 26393
139: PWMTACH  Method 1 - RPM is 2273
140: PWMTACH  set new PWM duty cycle - 60%
141: PWMTACH  TACH0 reading for INTR_REV revolution - 23192
142: PWMTACH  Method 1 - RPM is 2587
143: PWMTACH  set new PWM duty cycle - 70%
144: PWMTACH  TACH0 reading for INTR_REV revolution - 20905
145: PWMTACH  Method 1 - RPM is 2870
146: PWMTACH  set new PWM duty cycle - 80%
147: PWMTACH  TACH0 reading for INTR_REV revolution - 19185
148: PWMTACH  Method 1 - RPM is 3127
149: PWMTACH  set new PWM duty cycle - 90%
150: TIMER    # 1-second tick! VBAT Good!
151: PWMTACH  TACH0 reading for INTR_REV revolution - 17867
152: PWMTACH  Method 1 - RPM is 3358
153: PWMTACH  set new PWM duty cycle - 100%
154: PWMTACH  TACH0 reading for INTR_REV revolution - 16706
155: PWMTACH  Method 1 - RPM is 3591

```

6.16 Breathing/Blinking LED

Member file: bbled_app.c, bbled_app.h, bbled.c, bbled.h

6.16.1 Breathing/Blinking LED Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- LED0 and 1 - JP23 1-2 short, 3-4 short

Firmware:

- Config.h - TRACE_ENABLE=1, BBLED_EN=1

6.16.2 Breathing/Blinking LED SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.16.3 Breathing/Blinking LED test procedure

The sample code only shows the Breathing and Blinking mode, no PWM mode and PWM watchdog.

Power on the evb then see the LED0 operated in Breathing mode and LED1 is blinking at rate of 1hz.

6.17 Quad Mode SPI

Member file: bbled_app.c, bbled_app.h, bbled.c, bbled.h

6.17.1 QMSPI Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- JP39, JP40 all closed for shared SPI and general SPI

Firmware:

- Config.h - TRACE_ENABLE=1, SPI_EN=1, SPI_MODE_SEL=0 or 1

6.17.2 QMSPI SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.17.3 QMSPI test procedure

The test firmware support Single mode and Quad mode only, no Dual mode.

6.17.3.1 Single mode test

Set SPI_MODE_SEL = 0, rebuilt the code

Make sure Shared SPI flash has valid data pre-programmed.

MEC140x/141x EVB Sample Code Guide

Power on EVB, then the test firmware will read 8 bytes data from Shared SPI where start from address of 0, and program to General SPI. The trace message may like as below,


```

58: QMSPI    ### spi_SINGLE_mode_demo ###
59: QMSPI    shared SPI controller status = 00002200h
60: QMSPI    SPI - RESET command done.
61: QMSPI    shared SPI status 1 reg = 00h
62: QMSPI    shared SPI status 2 reg = 00h
63: QMSPI    shared SPI status 3 reg = 60h
64: QMSPI    shared SPI - BUSY bit checking succeeded
65: QMSPI    SPI - spi_single_read
66: QMSPI    SPI - spi_single_read done
67: QMSPI    shared SPI data =10h
68: QMSPI    shared SPI data =00h
69: QMSPI    shared SPI data =00h
70: QMSPI    shared SPI data =F7h
71: QMSPI    shared SPI data =00h
72: QMSPI    shared SPI data =03h
73: QMSPI    shared SPI data =00h
74: QMSPI    shared SPI data =6Ah
75: QMSPI    general SPI controller status = 00002200h
76: QMSPI    SPI - RESET command done.
77: QMSPI    general SPI status 1 reg = 00h
78: QMSPI    general SPI status 2 reg = 00h
79: QMSPI    general SPI status 3 reg = 60h
80: QMSPI    general SPI - BUSY bit checking succeeded
81: QMSPI    spi_set_WEL - done
82: QMSPI    SPI - Sector_Erase completed.
83: QMSPI    SPI - spi_single_read
84: QMSPI    SPI - spi_single_read done
85: QMSPI    before update, general SPI data =FFh
86: QMSPI    before update, general SPI data =FFh
87: QMSPI    before update, general SPI data =FFh
88: QMSPI    before update, general SPI data =FFh
89: QMSPI    before update, general SPI data =FFh
90: QMSPI    before update, general SPI data =FFh
91: QMSPI    before update, general SPI data =FFh
92: QMSPI    before update, general SPI data =FFh
93: QMSPI    spi_set_WEL - done
94: QMSPI    spi_page_program - waiting BUSY bit.....
95: QMSPI    spi_page_program - completed.
96: QMSPI    SPI - spi_single_read
97: QMSPI    SPI - spi_single_read done
98: QMSPI    after update, general SPI data =10h
99: QMSPI    after update, general SPI data =00h
100: QMSPI    after update, general SPI data =00h
101: QMSPI    after update, general SPI data =F7h
102: QMSPI    after update, general SPI data =00h
103: QMSPI    after update, general SPI data =03h
104: QMSPI    after update, general SPI data =00h
105: QMSPI    after update, general SPI data =6Ah
106: QMSPI    SPI Single mode initialization DONE
107: TIMER    # 1-second tick! VBAT Good!

```

6.17.3.2 Quad mode test

Set SPI_MODE_SEL = 1, rebuilt the code

Make sure Shared SPI flash has valid data pre-programmed.

Power on EVB, then the test firmware will read 8 bytes data from Shared SPI where start from address of 0, and program to General SPI. The trace message may like as below,

MEC140x/141x EVB Sample Code Guide

```

89: QMSPI      SPI - spi_quad_read
90: QMSPI      spi_set_WEL - done
91: QMSPI      spi_set_QE_mode - done.
92: QMSPI      spi_set_WEL - done
93: QMSPI      spi_set_SINGLE_mode - done.
94: QMSPI      SPI - spi_quad_read DONE
95: QMSPI      shared SPI data =10h
96: QMSPI      shared SPI data =00h
97: QMSPI      shared SPI data =00h
98: QMSPI      shared SPI data =F7h
99: QMSPI      shared SPI data =00h
100: QMSPI     shared SPI data =03h
101: QMSPI     shared SPI data =00h
102: QMSPI     shared SPI data =6Ah
103: QMSPI     general SPI controller status = 00002200h
104: QMSPI     SPI - RESET command done.
105: QMSPI     general SPI status 1 reg = 00h
106: QMSPI     general SPI status 2 reg = 00h
107: QMSPI     general SPI status 3 reg = 60h
108: QMSPI     general SPI - BUSY bit checking succeeded
109: QMSPI     spi_set_WEL - done
110: QMSPI     SPI - Sector_Erase completed.
111: QMSPI     SPI - spi_quad_read
112: QMSPI     spi_set_WEL - done
113: QMSPI     spi_set_QE_mode - done.
114: QMSPI     spi_set_WEL - done
115: QMSPI     spi_set_SINGLE_mode - done.
116: QMSPI     SPI - spi_quad_read DONE
117: QMSPI     before update, general SPI data =FFh
118: QMSPI     before update, general SPI data =FFh
119: QMSPI     before update, general SPI data =FFh
120: QMSPI     before update, general SPI data =FFh
121: QMSPI     before update, general SPI data =FFh
122: QMSPI     before update, general SPI data =FFh
123: QMSPI     before update, general SPI data =FFh
124: QMSPI     before update, general SPI data =FFh
125: QMSPI     SPI - spi_quad_page_program
126: QMSPI     spi_set_WEL - done
127: QMSPI     spi_set_QE_mode - done.
128: QMSPI     spi_set_WEL - done
129: QMSPI     spi_set_WEL - done
130: QMSPI     spi_set_SINGLE_mode - done.
131: QMSPI     spi_quad_page_program - waiting BUSY bit.....
132: QMSPI     spi_quad_page_program - completed.
133: QMSPI     SPI - quad_page_program done
134: QMSPI     SPI - spi_quad_read
135: QMSPI     spi_set_WEL - done
136: QMSPI     spi_set_QE_mode - done.
137: QMSPI     spi_set_WEL - done
138: QMSPI     spi_set_SINGLE_mode - done.
139: QMSPI     SPI - spi_quad_read DONE
140: QMSPI     after update, general SPI data =10h
141: QMSPI     after update, general SPI data =00h
142: QMSPI     after update, general SPI data =00h
143: QMSPI     after update, general SPI data =F7h
144: QMSPI     after update, general SPI data =00h
145: QMSPI     after update, general SPI data =03h
146: QMSPI     after update, general SPI data =00h
147: QMSPI     after update, general SPI data =6Ah

```


6.18 BC-Link

Member file: bclink_app.c, bclink_app.h, bclink.c, bclink.h

6.18.1 BC-Link Configuration

EVB:

- LED0 - JP23 13-14 short
- TFDP_CLK/TFDP_DATA – JP8 1-2 short, 4-5 short
- BC-Link 0 - JP14 – 1-2, 4-5, 7-8, 10-11 short
- BC-Link 1 – no need

BCLink EVB:

ECE1088, ECE1099 & ECE1105 EVB Assy.6484

Firmware:

- Config.h - TRACE_ENABLE=1, BCLINK_EN=1

6.18.2 BC-Link SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.18.3 BC_Link test procedure

Wire the BLink board to EVB BCLINK channel 0 (ECE1105)
After power on the board, then we can get below trace message.

```
83: BCLINK    BCLINK Device ID = 42h
84: BCLINK    BCLINK Current version = 10h
85: BCLINK    BCLINK Vendor ID = 1055h
```

6.19 Sleep mode

Member file: sleep_app.c, sleep_app.h, proc.c, proc.h

6.19.1 Sleep mode Configuration

EVB: (remove all jumpers which not mentioned here!!)

- LEDs - JP23 1-2, 3-4, 5-6, 11-12, 13-14 short
- CR_STRAP – JP9 1-2 short

- CMP-STRAP0 – JP11 – 2-3 short, comparator 0 disabled in default.
- VCC_PWRGD – JP56 pin2 ground
- VBAT - JP25 1-2 short
- VTR_33_18 – JP35 1-2 short
- VREF_CPU – JP42 pin 2 ground
- ICSP_MCLK – J32 pin 1 ground
- ADC_VREF – JP29 pin 1-4 short
- DAC_VREF – JP30 pin 1-4 short
- Clock source – JP3 2-3, JP4 1-2 short
- SPI ROM – JP39 all closed
- Others power jumper setting,
JP16 close
JP17 2-3 short
JP18 2-3 short
JP19 2-3 short
JP26 1-2 short
JP31 1-2 short
JP33 1-2 short
JP44 1-2 short
JP45 1-2 short
JP62 2-3 short

Daughter card Assy 6759:

- Clock source - JP1 1-2 short

Firmware:

- Config.h - SLEEP_EN=1

6.19.2 Sleep mode SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.19.3 Sleep mode test procedure

Measure the current between JP28 pin1-2, then can get around 0.17mA (or 142uA)

The sample code utilizes sleep_all bit to handle chip's power down process and measure the current for deepest sleep mode only.

6.20 Host Interface exercise

This chapter contains the entire host interface related functional blocks test, several basic hardware and software configuration are as below.

Member file: hif_lpc_app.c, hif_lpc_app.h

6.20.1 Host Interface configuration

Use MEC1408 MECC Card Assy. 6761 + Intel Broadwell CRB to test emulated 8042, ACPI EC interface, Mailbox interface, Port80 interface and EMI interface.

Intel Broadwell CRB –

None. (Keep original success-boot setting)

Firmware:

- Config.h - TRACE_ENABLE=1, MEC1408_MECC_BROADWELL=1, BOARD_TYPE=1

6.20.2 Host Interface SKERN Interface

Interrupt task: none

Event task: none

Timer task: none

6.20.3 Host Interface test basic preparation

1. Use AC adapter and power on Intel CRB in S5 state, then MEC1408 MECC card get +3.3v power from CRB and make sure LED1 is ON. Press power button to turn on the system to DOS.
2. Configure the PCH Bus:0, Device:31, Function:0, Regs:8Ch-8Fh to 0x003C0401, that's to enable I/O address from 400h to 43Fh positive decode.
3. The I/O address mapping for each module in MEC140x sample code are,
(The LPC device I/O port is configured at 43Eh/43Fh)

Emulate 8042	- 410h/414h
ACPI EC0(legacy)	- 420h/424h
ACPI EC0(4-byte)	- 420h-425h(427h)
Port 80-1	- 41Ah
Port 80-2	- 41Bh
Mailbox	- 41Eh/41Fh
EMI	- 400h-40Fh

4. Launch DEBUG.EXE

6.20.4 Test emulate 8042 interface

- O 410 55
- I 414 (make sure OBF is set)
- I 410 (obtain the FAh)
- O 414 aa
- I 414 (make sure OBF is set.)

- I 410 (obtain the FAh)

6.20.5 Test ACPI EC 0 interface – legacy

Set EC_IF_MODE=0,

- O 420 55
- I 424 (make sure OBF is set)
- I 420 (obtain the 78h)
- O 424 aa
- I 424 (make sure OBF is set.)
- I 420 (obtain the 78h)

6.20.6 Test ACPI EC 0 interface – 4-byte mode

Set EC_IF_MODE=1, when access to data byte 3, the corresponding IBF/OBF should be set/clear.

- O 420 A0
- O 421 A1
- O 422 A2
- O 423 A3 (data byte 3)
- I 424 (make sure OBF is set)
- I 420 (obtain 78h)
- I 421 (obtain 56h)
- I 422 (obtain 34h)
- I 423 (obtain 12h) (data byte 3)
- I 424 (make sure IBF is clear)

6.20.7 Test Port80 interface 0

- O 41A 11
- (check if firmware received is 11h via trace message)

6.20.8 Test Port80 interface 1

- O 41B 22
- (check if firmware received is 22h via trace message)

6.20.9 Test Mailbox interface

Use polling method for the communication with EC.

- O 41E 0
- O 41F 11
- I 41F

(check if the register is reset to 0(not the 11h) that means EC has responded the ORed=80h in EC2OS register)

- O 41E 1
- I 41F (it should be 91h, that 11h ORed with 80h)
- O 41F FF (clear EC2OS register)
- I 41F (confirm it is reset to 0(not the 91h))

6.20.10 Test EMI interface

- O 403 A0 (access emi_memory_0[])
- I 404 (obtain 88h) make sure emi_memory_0 is all 88h
- I 405 (obtain 88h)
- I 406 (obtain 88h)
- I 407 (obtain 88h)
- O 403 B0 (access emi_memory_1[])
- I 404 (obtain EEh) make sure emi_memory_0 is all EEh
- I 405 (obtain EEh)
- I 406 (obtain EEh)
- I 407 (obtain EEh)
- O 403 A0
- O 404 81
- O 405 82
- O 406 83
- O 407 84
- O 400 55 (notify EC that host has updated data in emi_memory_0[])
(make sure byte 0 to 3 are 81h to 84h in trace message)
- I 401
(make sure received is AAh, that mean EC has responded in emi_memory_1)
- O 403 B0
- I 404 (obtain 12h)
- I 405 (obtain 34h)
- I 406 (obtain 56h)
- I 407 (obtain 78h)

6.20.11 Test ChipMan utility

ChipMan utility requirement:

- EC firmware shall set up the EMI base 0 address at 0x000F_0100, and set both Read Limit and Write Limit to 0x7FFC.
- Host or EC firmware shall configure the EMI BAR I/O at 400h-40Fh and the LPC interface BAR I/O(logical device 0Ch), i.e. 2Eh/2Fh. The LPC interface I/O address can be in any available address since it was configurable in ChipMan.

MEC140x/141x EVB Sample Code Guide

- Enable PCH LPC I/O address decode for both LPC interface(same as last setting) and EMI(400h-40Fh).

Known issue:

The EMI exercise in Debug I/O of ChipMan is NOT supported.

For ChipMan detail manipulations please refer to *Chipman User Guide.pdf*.

Test process:

1. Set CHIPMAN=1 and rebuild the code. (When CHIPMAN=1, the GPIO157/LED0 1-second toggling is stopped, in order to support later test.)
2. Install Microchip ChipMan utility v4.17.0 to Windows 10.
3. Launch ChipMan, this may invoke Microchip Porttalk driver.
4. In “option\select device”, enter the LPC I/O address.
5. After the success communication with the chip, below screen can be shown and make sure the “Last value is NOT all 00h or FFh.

Register Name	Address	R/W	Last Value	Units	Abbreviation	Bus Type
LPC Interface Activate	30	RW	01	Hex	LPC_IF	LPC
SIRQ_IRQ0 Configuration	40	RW	FF	Hex	SIRQ0_CFG	LPC
SIRQ_IRQ1 Configuration	41	RW	FF	Hex	SIRQ1_CFG	LPC
SIRQ_IRQ2 Configuration	42	RW	FF	Hex	SIRQ2_CFG	LPC
SIRQ_IRQ3 Configuration	43	RW	FF	Hex	SIRQ3_CFG	LPC
SIRQ_IRQ4 Configuration	44	RW	FF	Hex	SIRQ4_CFG	LPC
SIRQ_IRQ5 Configuration	45	RW	FF	Hex	SIRQ5_CFG	LPC
SIRQ_IRQ6 Configuration	46	RW	FF	Hex	SIRQ6_CFG	LPC
SIRQ_IRQ7 Configuration	47	RW	FF	Hex	SIRQ7_CFG	LPC
SIRQ_IRQ8 Configuration	48	RW	FF	Hex	SIRQ8_CFG	LPC
SIRQ_IRQ9 Configuration	49	RW	FF	Hex	SIRQ9_CFG	LPC
SIRQ_IRQ10 Configuration	4A	RW	FF	Hex	SIRQ10_CFG	LPC
SIRQ_IRQ11 Configuration	4B	RW	FF	Hex	SIRQ11_CFG	LPC
SIRQ_IRQ12 Configuration	4C	RW	FF	Hex	SIRQ12_CFG	LPC
SIRQ_IRQ13 Configuration	4D	RW	FF	Hex	SIRQ13_CFG	LPC
SIRQ_IRQ14 Configuration	4E	RW	FF	Hex	SIRQ14_CFG	LPC
SIRQ_IRQ15 Configuration	4F	RW	FF	Hex	SIRQ15_CFG	LPC
Configuration Port Base Address Control	61:60	RW	8C01	Hex	CFG_CFG	LPC
Configuration Base I/O Address	63:62	RW	043E	Hex	BAR_BASE	LPC
Embedded Memory Interface 0 Base Address...	65:64	RW	800F	Hex	CFG_EMI	LPC
Embedded Memory Interface 0 Base I/O Add...	67:66	RW	0400	Hex	BAR_EMI	LPC
8042 Embedded Keyboard Controller Address...	69:68	RW	8104	Hex	BAR_EMI	LPC
8042 Embedded Keyboard Controller Address	6B:6A	RW	0410	Hex	BAR_EMI	LPC
ACPI EC 0 Base Address Control	6D:6C	RW	8304	Hex	CFG_ECO	LPC
ACPI EC 0 Base I/O Address	6F:6E	RW	0420	Hex	BAR_ECO	LPC
ACPI EC 1 Base Address Control	71:70	RW	0407	Hex	CFG_ECI	LPC
ACPI EC 1 Base I/O Address	73:72	RW	0066	Hex	BAR_ECI	LPC
ACPI PM1 Base Address Control	75:74	RW	0507	Hex	CFG_PM1	LPC
ACPI PM1 Base I/O Address	77:76	RW	0000	Hex	BAR_PM1	LPC
Legacy Keyboard Interface Base Address Co...	79:78	RW	0600	Hex	CFG_KBD	LPC
Legacy Keyboard Interface Base I/O Address	7B:7A	RW	0092	Hex	BAR_KBD	LPC
UART 0 Base Address Control	7D:7C	RW	0707	Hex	CFG_UART0	LPC
UART 0 Base I/O Address	7F:7E	RW	0000	Hex	BAR_UART0	LPC
MMCR Base I/O Address	41:40	RW	FFFF	Hex	BAR_MMCR	LPC
Mailbox Interface Base Address Control	81:80	RW	8901	Hex	CFG_MBX	LPC
Mailbox Interface Base I/O Address	83:82	RW	041E	Hex	BAR_MBX	LPC
ACPI EC 2 Base Address Control	85:84	RW	0A07	Hex	CFG_ECO	LPC
ACPI EC 2 Base I/O Address	87:86	RW	0000	Hex	BAR_ECO	LPC
ACPI EC 3 Base Address Control	89:88	RW	0B07	Hex	CFG_ECO	LPC
ACPI EC 3 Base I/O Address	8B:8A	RW	0000	Hex	BAR_ECO	LPC
Port 80 BIOS Debug Port 0 Control	8D:8C	RW	9300	Hex	PORT80_D...	LPC
Port 80 BIOS Debug Port 0 Address	8F:8E	RW	041A	Hex	PORT80_D...	LPC
Port 80 BIOS Debug Port 1 Control	91:90	RW	9600	Hex	PORT80_D...	LPC
Port 80 BIOS Debug Port 1 Address	93:92	RW	041B	Hex	PORT80_D...	LPC
LPC Host Address Memory BAR Address	A3:A0	RW	00000000	Hex	BAR_LPC	LPC
LPC Host Address Memory BAR Configuration	A7:A4	RW	00000000	Hex	CFG_LPCB...	LPC

Bit Field Name	Bit(s)	Last Value	Translation
Reserved	7-1	0	
Activate	0	1	Active

Block 1: Configuration / Group Logical Device 0Ch: LPC Interface

MEC1408 found

6. Click “10.MMCR Registers” and select “GPIO” in left pane
7. Click “GPIO157” in right Pane

MEC140x/141x EVB Sample Code Guide

- In right-bottom pane, change the value of bit16, GPIO output data, and measure the TP38 pin 1 to see if this pin's state can be chaged by the Chipman configuration.

Microchip Chip Manager - Microchip Confidential

File View Options Control Help

MEC1408

- 1: Configuration
- 2: Embedded Memory Interface
- 3: 8042 Emulated Keyboard Controller Registers
- 4: Port92 Registers
- 5: Mailbox Registers
- 6: ACPI EC 0-1
- 7: ACPI PM1
- 8: UART
- 9: RTC Registers
- 10: MMCR Registers
- GPIO
 - Global Configuration Registers
 - Hibernation Timer
 - IMAP
 - ITVIC
 - Keyboard Scan Interface
 - LED Registers
 - LPC
 - Mailbox Interface Registers
 - PCR
 - PECI
 - PS/2
 - PWM
 - Quad SPI Master Controller
 - RTOS Timer Registers
 - SMB Device Interface Registers
 - TACH
 - Trace FIFO Debug Port
 - UART
 - VBAT Registers
 - VBAT-Powered Control Interface Registers
 - Watchdog Timer Interface Registers
 - Week Timer Registers
 - BIOS Debug Port
 - DMA
 - ACPI EC Interface
 - ACPI PM1
 - BC-Link Address Registers
 - Basic Timer
 - EC_REG_BANK
 - ADC Registers
 - DAC Activate Registers
 - 8042 Host Interface

Register Name	Address	R/W	Last Value	Units	Abbreviation	Bus Type
GPIO0130 Pin Control	81163:8...	RW	00000041	Hex	GPIO0130...	EMI
GPIO0131 Pin Control	81167:8...	RW	00000041	Hex	GPIO0131...	EMI
GPIO0132 Pin Control	8116B...	RW	01000041	Hex	GPIO0132...	EMI
GPIO0133 Pin Control	8116F:8...	RW	01000041	Hex	GPIO0133...	EMI
GPIO0134 Pin Control	81173:8...	RW	01000041	Hex	GPIO0134...	EMI
GPIO0135 Pin Control	81177:8...	RW	01000041	Hex	GPIO0135...	EMI
GPIO0136 Pin Control	8117B...	RW	01000041	Hex	GPIO0136...	EMI
GPIO0140 Pin Control	81183:8...	RW	00000041	Hex	GPIO0140...	EMI
GPIO0141 Pin Control	81187:8...	RW	01001340	Hex	GPIO0141...	EMI
GPIO0142 Pin Control	8118B...	RW	01001340	Hex	GPIO0142...	EMI
GPIO0143 Pin Control	8118F:8...	RW	01000041	Hex	GPIO0143...	EMI
GPIO0144 Pin Control	81193:8...	RW	01000041	Hex	GPIO0144...	EMI
GPIO0145 Pin Control	81197:8...	RW	00000041	Hex	GPIO0145...	EMI
GPIO0146 Pin Control	8119B...	RW	00000041	Hex	GPIO0146...	EMI
GPIO0147 Pin Control	8119F:8...	RW	01000041	Hex	GPIO0147...	EMI
GPIO0150 Pin Control	811A3...	RW	01000041	Hex	GPIO0150...	EMI
GPIO0151 Pin Control	811A7...	RW	01000041	Hex	GPIO0151...	EMI
GPIO0152 Pin Control	811AB...	RW	01000041	Hex	GPIO0152...	EMI
GPIO0153 Pin Control	811AF...	RW	00000041	Hex	GPIO0153...	EMI
GPIO0154 Pin Control	811B3...	RW	01000041	Hex	GPIO0154...	EMI
GPIO0155 Pin Control	811B7...	RW	01000041	Hex	GPIO0155...	EMI
GPIO0156 Pin Control	811BB...	RW	01000041	Hex	GPIO0156...	EMI
GPIO0157 Pin Control	811BF...	RW	00000241	Hex	GPIO0157...	EMI
GPIO0160 Pin Control	811C3...	RW	01000041	Hex	GPIO0160...	EMI
GPIO0161 Pin Control	811C7...	RW	01000041	Hex	GPIO0161...	EMI
GPIO0162 Pin Control	811CB...	RW	01000041	Hex	GPIO0162...	EMI
GPIO0163 Pin Control	811CF...	RW	01000041	Hex	GPIO0163...	EMI
GPIO0164 Pin Control	811D3...	RW	01000041	Hex	GPIO0164...	EMI
GPIO0165 Pin Control	811D7...	RW	01000041	Hex	GPIO0165...	EMI
GPIO0166 Pin Control	811DB...	RW	01000041	Hex	GPIO0166...	EMI
Output GPIO[000:036]	81283:8...	RW	00400002	Hex	OUTPUT_G...	EMI
Output GPIO[040:076]	81287:8...	RW	00000000	Hex	OUTPUT_G...	EMI
Output GPIO[100:127]	8128B...	RW	00803000	Hex	OUTPUT_G...	EMI

Bit Field Name	Bit(s)	Last Value	Translation
Reserved	31-25	0	
GPIO input from pad	24	0	
Reserved	23-17	0	
GPIO output data	16	0	
Reserved	15-14	0	
Mux Control	13-12	0	GPIO Function Selected
Polarity	11	0	Non-inverted
GPIO Output Control Select	10	0	
GPIO Direction	9	1	Output
Output Buffer Type	8	0	Push-Pull
Edge Enable (edge_en)	7	0	Edge detection disabled
Interrupt Detection (int_det)	6-4	4	
Power Gating Signals	3-2	0	VTR
PU/PD (PU_PD)	1-0	1	Pull Up

Block 10: MMCR Registers / Group GPIO

MEC1408 found

Db1 click value in Last Value column to edit

7 Example for Code and Data memory range relocation

In some applications may expect to relocate the code and data memory range. The amount of memory space is always the same but just to perform the space adjustment. Here is an 50K data memory example for adjusting the Code/Data from 96K/32K to 78K/50K in MEC1404 device.

1. Modify Linker Script File, also refer to pMEC1404_50K_data.LD
Specify the Program(Code) and Data memory as below.

```
MEMORY
{
/*kseg0_program_mem    (rx) : ORIGIN = 0xBFD00000, LENGTH = 0x18000 */
  kseg0_program_mem    (rx) : ORIGIN = 0xBFD00000, LENGTH = 0x13800
  sfrs                 : ORIGIN = 0xA0000000, LENGTH = 0x100000
/*kseg1_data_mem       (rwx): ORIGIN = 0xBFD18000, LENGTH = 0x8000 */
  kseg1_data_mem       (rwx): ORIGIN = 0xBFD13800, LENGTH = 0xC800
}
```

2. MPLAB X IDE

- Make sure the MPLAB X IDE is NOT installed in the path contains the "space".
- Modify PIC file
Extract(unzip) the PIC file from
<Installed directory>Microchip\MPLABX\mplab_ide\mplablibs
modules\ext\crownking.edc.jar
- Find the PIC file under the location below, here is MEC1404.PIC
Content/eds/32xxxx/
- PIC file Modification
Find the "GPRDataSector" for data memory range and "CodeSector" for code memory range

Modify the content as below,

```
<edc:GPRDataSector edc:beginaddr="0x1fd13800" edc:endaddr="0x1fd20000"
edc:kseg0="0x80000000" edc:kseg1="0xA0000000" edc:ksegdef="2"
edc:regionid="kseg1_data_mem"/>
<edc:CodeSector edc:beginaddr="0x1fd00000" edc:endaddr="0x1fd13800"
edc:kseg0="0x80000000" edc:kseg1="0xa0000000" edc:ksegdef="2"
edc:kuseg="0x60000000" edc:regionid="code"/>
```

- After the edit then copy this PIC file back to the same location with JAR file, no need to ZIP back to JAR file.

8 Release History

- The **Version** is the formal firmware release version.
- The **Perforce label** is the label in Perforce version control system, the build # in tail of label also contains the kernel firmware version.
- The code size that enveloped with parentheses means when TRACE ALL ON.

Version	0.5
Date	2015/12/22
Perforce label	MEC140x_evb_sample_code_build_0005
Signature	None
Binary file	spi16M_mec140x_sdk.bin
Description	1. Add support for MEC1406. So now support MEC1404, MEC1406 and MEC1408.

Version	0.4
Date	2015/11/25
Perforce label	MEC140x_evb_sample_code_build_0004
Signature	None
Binary file	spi16M_mec140x_sdk.bin
Description	1. Simplify PS2 module 2. Host Interface 8042, ACPI EC0(legacy and 4-byte mode), Port80, mailbox, EMI and ChipMan exercise.

Version	0.3
Date	2015/10/6
Perforce label	MEC140x_evb_sample_code_build_0003
Signature	None
Binary file	spi16M_mec140x_sdk.bin
Description	1. Feature contains: UART, GPIO, Basic Timer, RTOS Timer, Hibernation Timer, Watchdog Timer, nReset_out, PS2, Matrix keyboard, SMBus, Week Timer, VBAT-Powered Control Interface, ADC, DAC, PWM, Fan tachometer, Blinking/Breathing LED, SPI, BC-LINK, Sleep mode 2. Simplify Keyscan module

Version	0.2
Date	2015/6/10
Perforce label	MEC140x_evb_sample_code_build_0002

MEC140x/141x EVB Sample Code Guide

Signature	None
Binary file	spi16M_mec140x_sdk.bin
Description	<ol style="list-style-type: none"> 1. Feature contains: UART, GPIO, Basic Timer, RTOS Timer, Hibernation Timer, Watchdog Timer, nReset_out Timer, PS2, Matrix keyboard, SMBus, Week Timer, VBAT-Powered Control Interface. 2. Update universal spi generator utility 3. Update MMCR, mec1404.h, #15 in Perforce 4. Update code to meet datasheet 2015/6/9

Version	0.1
Date	2015/3/19
Perforce label	MEC140x_evb_sample_code_build_0001
Signature	None
Binary file	spi16M_mec140x_sdk.bin
Description	<ol style="list-style-type: none"> 1. Initial – trace message and LED0 blinking per 1 second

9 Test Report

TBD.

10 Known issues

TBD.

Document Revision History

Rev	Date	Author/Editor	Summary of Changes (include page numbers & sections)
0.4	2015/12/22	L.K.	1. Add a note into chapter 5 for describing the unpowered chip state while programming the standalone SPI flash. 2. Update introduction 3. Update sample code base 4. Update doc for MEC1406 support
0.3	2015/11/25	L.K.	- Update chapter - Host Interface - Integrate EVB & MECC configuration document - release for sample code build_0004
0.2	2015/11/16	Tom T.	General review and clean-up
0.1	2015/11/6	L.K.	release for sample code build_0003