ITE Embedded Controller
Firmware Programing Guide

ITE TECH. INC.

All specifications are subject to change without notice. The material contained in this document supersedes all previous documentation issued for the related products included herein. Please contact ITE Tech. Inc. for the latest document(s).

All sales are subject to ITE's Standard Terms and Conditions, a copy of which is included in the back of this document.

Additional copies of this manual or other ITE literature may be obtained from:

ITE Tech. Inc.
Marketing Department
7F, No. 233-1, Baociao Rd., Sindian City,
Taipei County 23145, Taiwan, R.O.C.

Phone: (02) 29126889
Fax: (02) 2910-2551, 2910-2552

If you have any marketing or sales questions, please contact:

P.Y. Chang, at ITE Taiwan: E-mail: p.y.chang@ite.com.tw, Tel: 886-2-29126889 X6052, Fax: 886-2-29102551

To find out more about ITE, visit our World Wide Web at:

http://www.ite.com.tw

Or e-mail **itesupport@ite.com.tw** for more product information/services

Version History

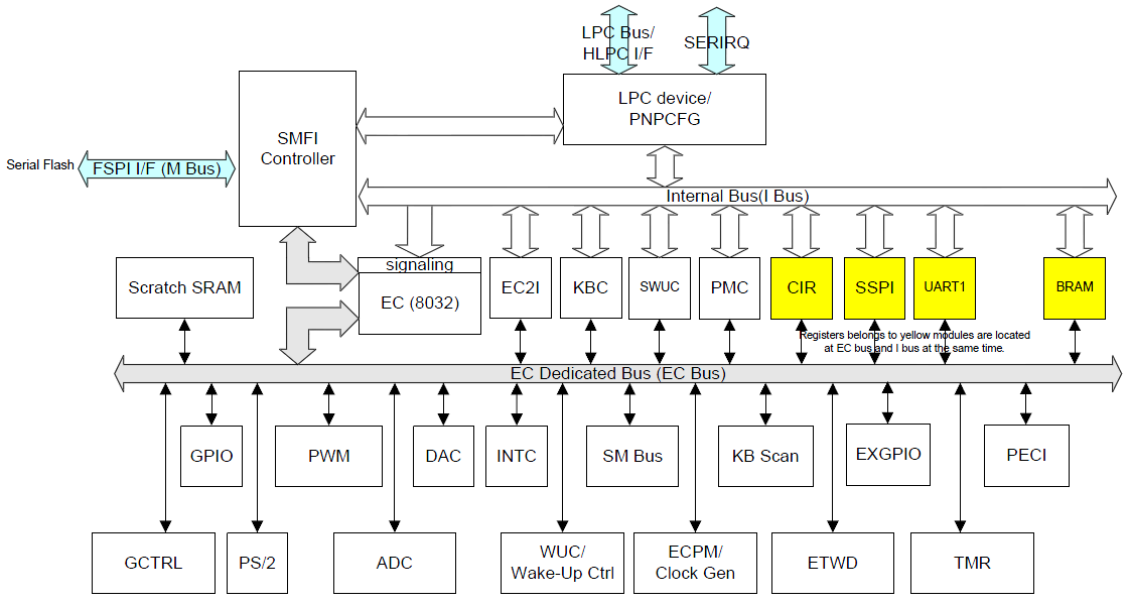
Data	Version	Description
04/09/2012	1.0	First release

CONTENTS

1. Introduction

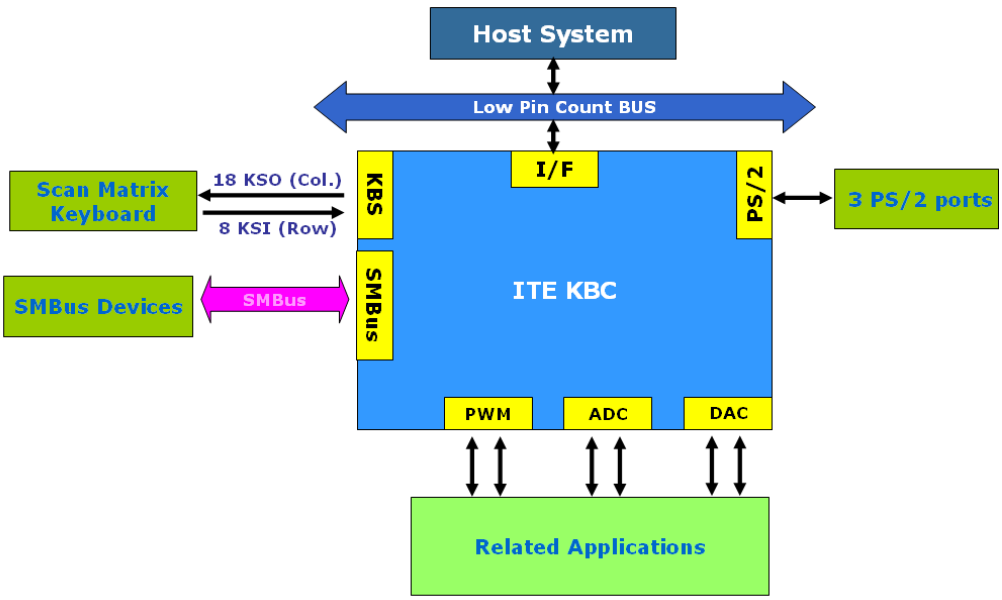
1.1 General Description

The ITE Embedded Controller (EC) is a highly integrated controller which is suitable for mobile system applications. It's embedded with the 8032 microcontroller which can execute the EC firmware (EC code) to dynamically program interfaces and general purpose input/output. The following figure shows the block diagram of the EC:



When the EC is powered on, the 8032 microcontroller will fetch the EC firmware at the physical address 0x0000 of the SPI Flash ROM. The most frequently used features are implemented in the EC Codebase as the kernel code which contains keyboard scanning, PS/2 data collection, host interface 64/60 port commands, and the SMBus API functions.

The applications of EC can be shown as the following example:



2. Compiler Environment

The Keil C components (C51, A51, and BL51) are requested for generating bin file of EC firmware. Please make sure Keil C is properly installed and containing at least these components before generating bin file. Other necessary firmware utilities are included in the source code.

2.1 Source Code Folder Definition

The IDE environment and command line environment are supported for generating EC firmware bin file. In Figure 2-1-1, folder “DOS_Example” and “DOS_Pure” are command line versions. Folder “uVision_Example” and “uVision_Pure” are IDE versions.

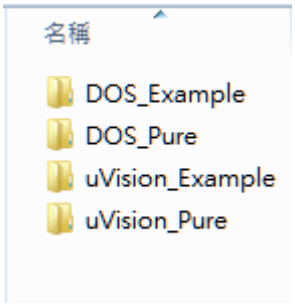


Figure 2-1-1

In Figure 2-1-2, all versions have the same contents in “CHIP” and “CORE” of “Code” folder. “DOS_Example” and “uVision_Example” have the same contents of “OEM” of “Code” and including more example functions for reference. “DOS_Pure” and “uVision_Pure” have the same contents of “OEM” of “Code”. Pure version is suggested, you are free to add OEM function.

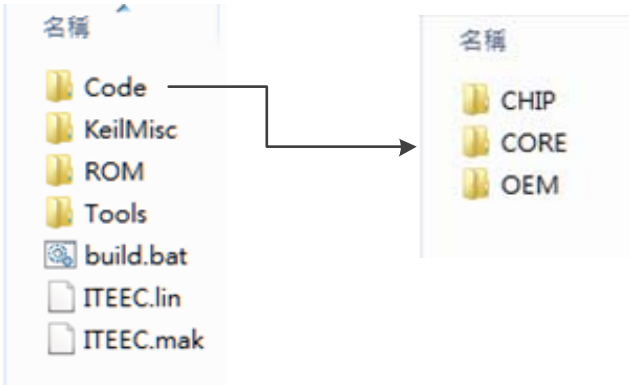


Figure 2-1-2

2.2 Source Code Folder Architecture of Command Line Version

Figure 2-2-1 is folder architecture of command line version. The “Code” folder includes definition of chip registers, kernel code and OEM code. Most of the time, programmer don’t have to modify the contents of kernel code. The “KeilMisc” folder includes LST, OBJ, and MAP files. Programmer can always refer to these folders when necessary. If compiling and linking can be completed, programmer can find out the bin file of EC firmware in “ROM” folder. The “Tools” folder includes “FU.EXE”, “Hex2bin.exe” and “NMake” folder. “FU.EXE” is firmware utility for adjusting bin file. “Hex2bin.exe” is hex to binary file converter utility. The use of NMake here is deciding which *.C file need to re-compile and linking. Please visit the Microsoft Website for more use.

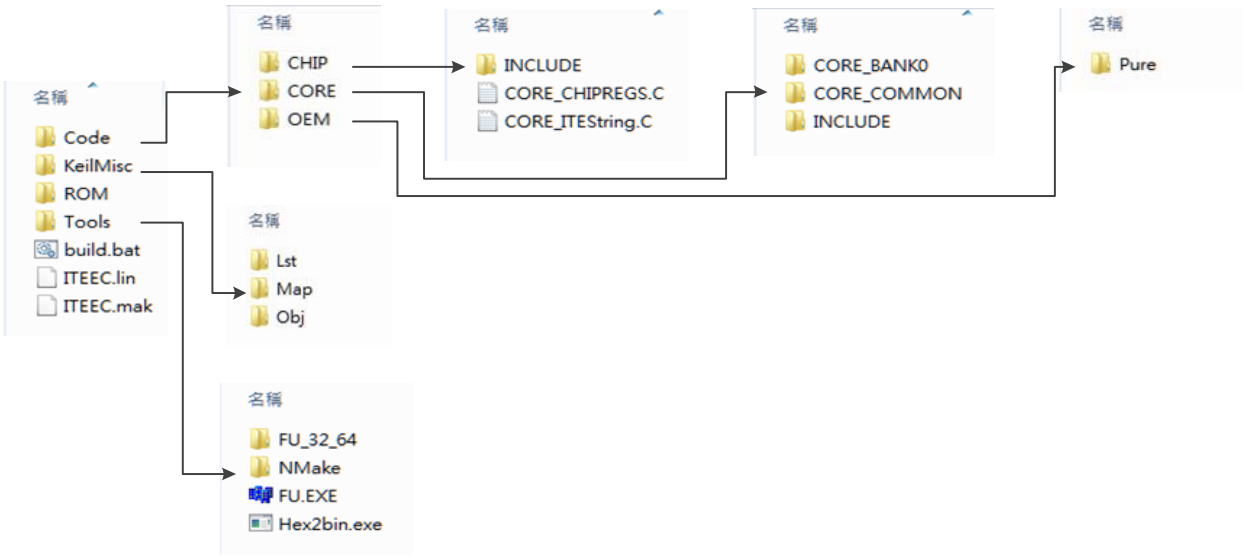


Figure 2-2-1

2.3 Source Code Folder Architecture of IDE Version

The folder architecture of IDE version is very similar to command line version. The difference is that IDE version has uVision folder for project setting and linker control file shown in Figure 2-3-1.

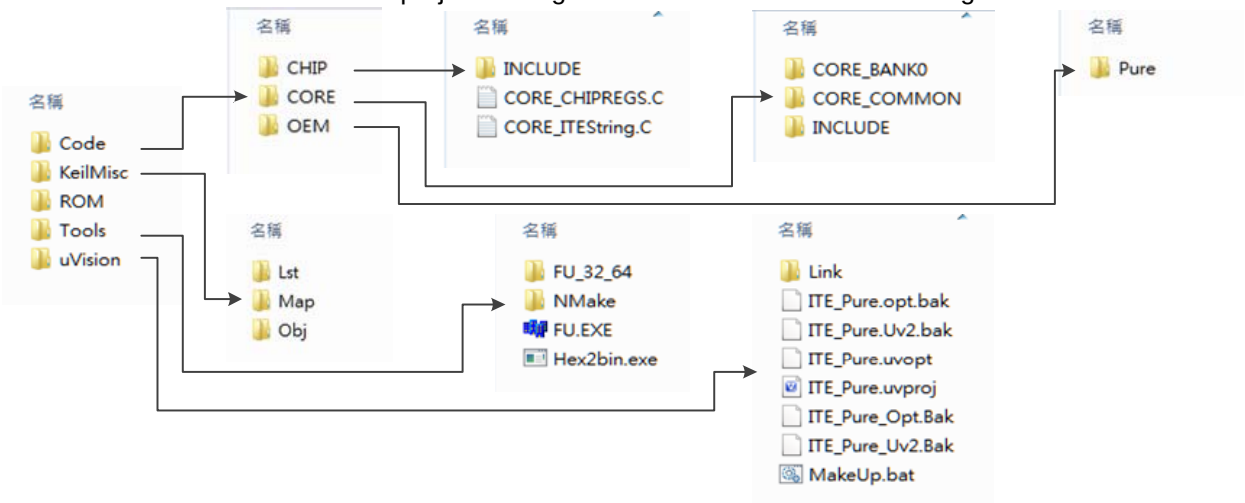


Figure 2-3-1

2.4 Generating a Bin File of Command Line Version

First of all, please modify build.bat if necessary. Only two item need to be modified, shown in Figure 2-4-1. The Item [A] is final size (KB) of EC binary file. In most case, setting “64” is enough. The item [B] is installation path of Keil C. Please make changes according to installation path.

```
REM ************************************************************************************************************************
REM     Setting
REM ************************************************************************************************************************
REM ------------------------------------------------------------------------------------------------------------------------
REM (64, 96, 128, and 160 are valid)
    A   SET EC_ROM_SIZE=64

    B   path=C:\Keil\C51\BIN;.\Tools;.\Tools\NMake;
REM ------------------------------------------------------------------------------------------------------------------------
```

Figure 2-4-1

Next, copying a OEM code in “OEM” folder, shown in Figure 2-4-2. “xxx” can be the project name or others. All of OEM code refer to the same kernel code and chip register definition.

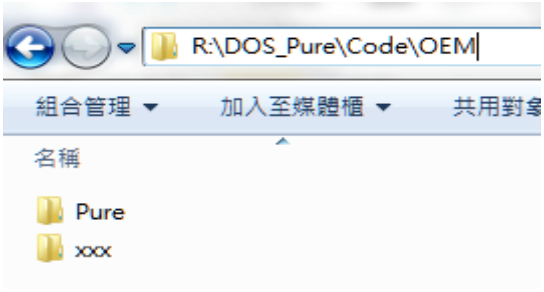


Figure 2-4-2

Then, set directory path of build.bat in the command prompt, shown in Figure 2-4-3. Finally, typing “build xxx” in command prompt. EC firmware binary file “xxx.bin” will be in “ROM” folder if all setting is correct. Please type “build” in command prompt to get more use.

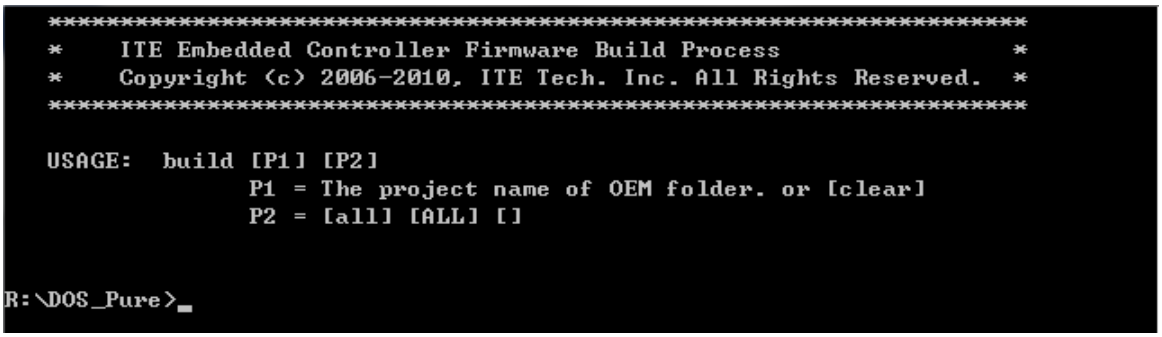


Figure 2-4-3

2.5 Generating a Bin File of IDE Version

We provide a project setting of Keil C in “uVision” folder. Actually, just open the project and pressing build button, as shown in Figure 2-5-1. EC firmware binary file will be in “ROM” folder. Programmer can modify “MakeUp.bat” for EC firmware code size. We use linker control file (“uITEEC.lin” of “Link” folder) for linker control setting. Please visit the Keil Website for more use.

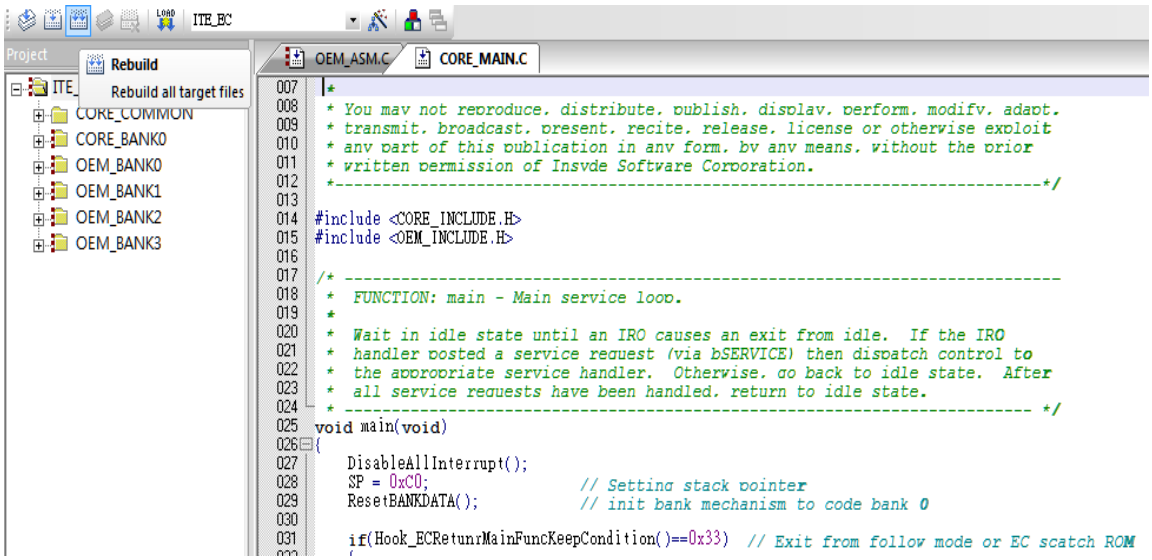


Figure 2-5-1

2.6 Creating a New C Source File of Command Line Version

If the existing C source files aren't enough, programmer can create a new C source file in "OEM" project folder and modify two files ("ITEEC.lin" / "ITEEC.mak"). For example, we want to create a new C source file and name is "OEM_TEST.C". First, we modify "ITEEC.mak" file after create "OEM_TEST.C" file as shown in Figure 2-5-2 and Figure2-5-3.

```
KeilMisc\Obj\OEM_LPC.OBJ:Code\Oem\OEM_LPC.C $(COREInclude) $(OEMInclude) $(CHIPInclude)
    $(CC) Code\Oem\OEM_LPC.C $(CDirectives)
    move Code\Oem\OEM_LPC.OBJ KeilMisc\Obj
    move Code\Oem\OEM_LPC.LST KeilMisc\Lst

KeilMisc\Obj\OEM_TIMER.OBJ:Code\Oem\OEM_TIMER.C $(COREInclude) $(OEMInclude) $(CHIPInclude)
    $(CC) Code\Oem\OEM_TIMER.C $(CDirectives)
    move Code\Oem\OEM_TIMER.OBJ KeilMisc\Obj
    move Code\Oem\OEM_TIMER.LST KeilMisc\Lst

KeilMisc\Obj\OEM_TEST.OBJ:Code\Oem\OEM_TEST.C $(COREInclude) $(OEMInclude) $(CHIPInclude)
    $(CC) Code\Oem\OEM_TEST.C $(CDirectives)
    move Code\Oem\OEM_TEST.OBJ KeilMisc\Obj
    move Code\Oem\OEM_TEST.LST KeilMisc\Lst

KeilMisc\Obj\OEM_HSPI.OBJ:Code\Oem\OEM_HSPI.C $(COREInclude) $(OEMInclude) $(CHIPInclude)
    $(CC) Code\Oem\OEM_HSPI.C $(CDirectives1)
    $(AS) Code\Oem\OEM_HSPI.SRC $(ADirectives)
    move Code\Oem\OEM_HSPI.OBJ KeilMisc\Obj
    move Code\Oem\OEM_HSPI.LST KeilMisc\Lst
    del Code\Oem\OEM_HSPI.SRC
```

Figure 2-5-2

```
            KeilMisc\Obj\OEM_PM2.OBJ\
            KeilMisc\Obj\OEM_PS2.OBJ\
            KeilMisc\Obj\OEM_SPI.OBJ\
            KeilMisc\Obj\OEM_LPC.OBJ\
            KeilMisc\Obj\OEM_TIMER.OBJ\
            KeilMisc\Obj\OEM_HSPI.OBJ\
            KeilMisc\Obj\OEM_ASM.OBJ\
            KeilMisc\Obj\OEM_Debug.OBJ\
            KeilMisc\Obj\OEM_TEST.OBJ\
            KeilMisc\Obj\OEM_BANK1_Func.OBJ\
            KeilMisc\Obj\OEM_BANK2_Func.OBJ\
            KeilMisc\Obj\OEM_BANK3_Func.OBJ
    $(Linker) @ITEEC.lin

#-------------------------------------------------------------------
# Compile chip file
#-------------------------------------------------------------------
KeilMisc\Obj\CORE_CHIPREGS.OBJ:Code\CHIP\CORE_CHIPREGS.C $(COREInclude) $(OEMInclude)
    $(CC) Code\CHIP\CORE_CHIPREGS.C $(CDirectives)
    move Code\CHIP\CORE_CHIPREGS.OBJ KeilMisc\Obj
    move Code\CHIP\CORE_CHIPREGS.LST KeilMisc\Lst
```

Figure 2-5-3

Then, modify "ITEEC.lin" file as shown in Figure 2-5-4. The OBJ file can be assigned to any bank In "ITEEC.lin" file. Finally, please refer to section 2.4 to generate new EC bin file.

```
    KeilMisc\Obj\OEM_6064.OBJ,
    KeilMisc\Obj\OEM_MEMORY.OBJ,
    KeilMisc\Obj\OEM_MailBox.OBJ,
    KeilMisc\Obj\OEM_Ver.OBJ,
    KeilMisc\Obj\OEM_LCD.OBJ,
    KeilMisc\Obj\OEM_PORT686C.OBJ,
    KeilMisc\Obj\OEM_PECI.OBJ,
    KeilMisc\Obj\OEM_PM3.OBJ,
    KeilMisc\Obj\OEM_IRQ.OBJ,
    KeilMisc\Obj\OEM_PM2.OBJ,
    KeilMisc\Obj\OEM_PS2.OBJ,
    KeilMisc\Obj\OEM_SPI.OBJ,
    KeilMisc\Obj\OEM_LPC.OBJ,
    KeilMisc\Obj\OEM_TIMER.OBJ,
    KeilMisc\Obj\OEM_HSPI.OBJ,
    KeilMisc\Obj\OEM_ASM.OBJ,
    KeilMisc\Obj\OEM_TEST.OBJ,
    KeilMisc\Obj\OEM_Debug.OBJ},
 b1{KeilMisc\Obj\OEM_BANK1_Func.OBJ},
 b2{KeilMisc\Obj\OEM_BANK2_Func.OBJ},
 b3{KeilMisc\Obj\OEM_BANK3_Func.OBJ} to ITEEC.ABS
```

Figure 2-5-4

2.7 Creating a New C Source File of IDE Version

In IDE environment, the “Manage Components” selection of project can be used to add or remove C source file. Please refer to the Keil C related teaching book or Website for more use.

3. EC Memory Space

Figure 3-1

Internal Ram	
Range	Description
00h ~ 07h	R0 – R7 bank0 for general function.
08h ~ 0Fh	R0 – R7 bank1 reserved / local or global variables of internal ram. Note : [?BANK?DATA] occupy 08h for code bank switch.
10h ~ 17h	R0 – R7 bank2 for interrupt service routine.
18h ~ 1Fh	R0 – R7 bank3 reserved / local or global variables of internal ram.
20h ~ BFh	Kernel code occupy. 20h – 2Fh are bit addressable.
C0h ~ FFh	For stack pointer use.

Table 3-1

External Ram	
Range	Description
000h ~ 0FFh	Kernel code occupy.
100h ~ 1FFh	OEM function use
200h ~ 2FFh	OEM function use
300h ~ 3FFh	OEM function use
400h ~ 4FFh	OEM function use
500h ~ 5FFh	OEM function use
600h ~ 6FFh	Scratch ROM FE00h - FEFFh
700h ~ 7FFh	Scratch ROM FF00h - FFFFh / Function local variables
800h ~ 8FFh	Scratch SRAM 1 / OEM function use
900h ~ 9FFh	Scratch SRAM 1 / OEM function use
A00h ~ AFFh	Scratch SRAM 1 / OEM function use
B00h ~ BFFh	Scratch SRAM 1 / OEM function use
C00h ~ CFFh	Scratch SRAM 2 / OEM function use
D00h ~ DFFh	Scratch SRAM 2 / OEM function use
E00h ~ EFFh	Scratch SRAM 3 / OEM function use
F00h ~ FFFh	Scratch SRAM 4 / OEM function use

Table 3-2

4. Firmware Architecture

The flowchart of EC firmware is shown in Figure 4-1.

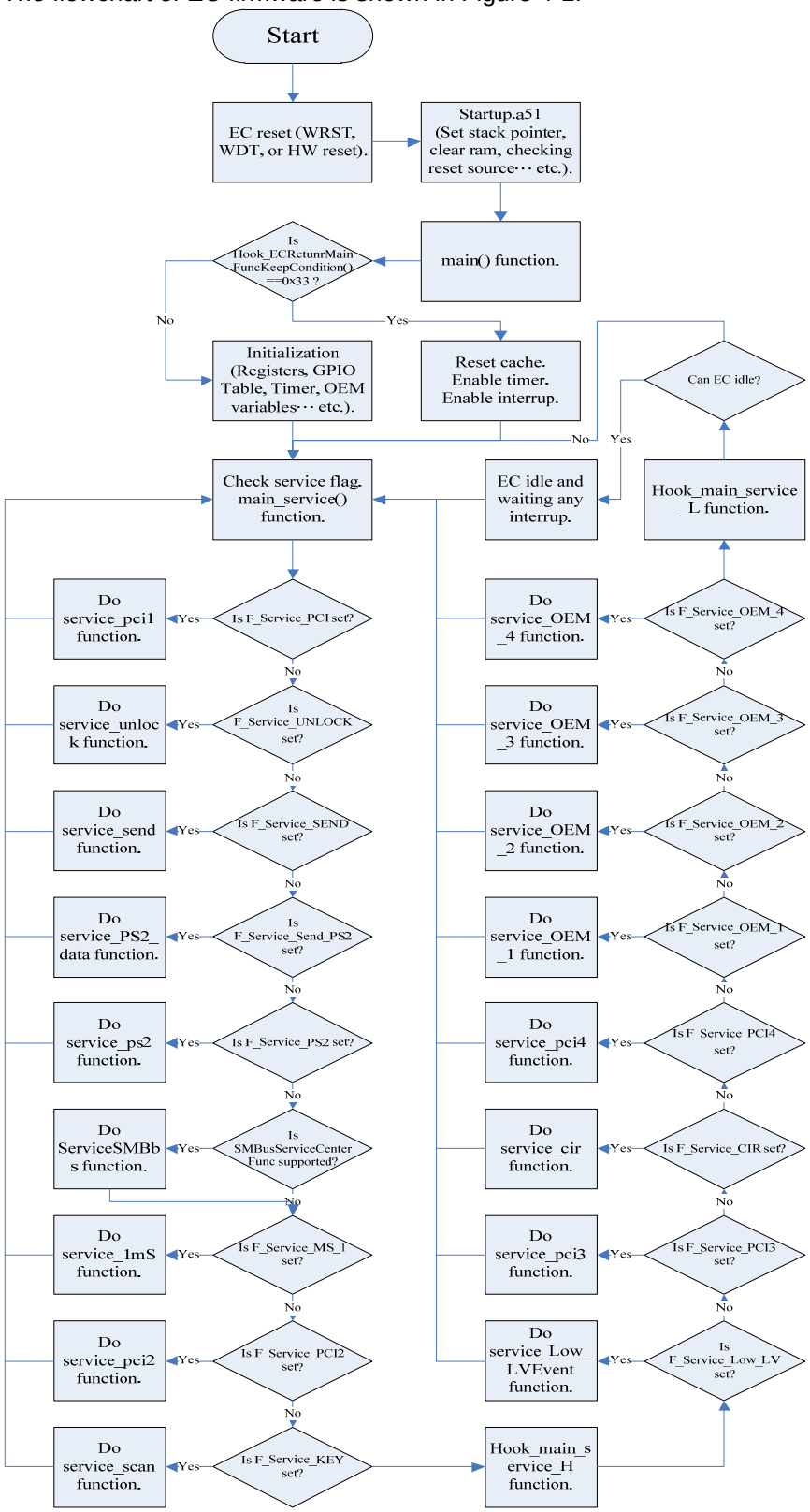


Figure 4-1

4.1 The Start of EC Firmware

The Startup.a51 file is the start of EC firmware. After EC reset, firmware will set stack point to C0h (occupy internal ram C0h ~ FFh) and using SFR P1.0, P1.1 for code bank switch. “Startup.a51” provides three functions for OEM function use before jump to “main” function, shown in Figure 4-1-1. Please refer to Table 4-1-1 for more description of functions.

```
;2008+ITE+Start
;               MOV     SP,#?STACK-1
                MOV     SP,#0C0H    ; Stack Point
                MOV     DPTR,#1001H ; FPCFG Register
                MOV     A,#03FH     ; Use 8032 P1[0] and P1[1] as code banking source
                MOVX    @DPTR,A
;2008+ITE+End

; This code is required if you use L51_BANK.A51 with Banking Mode 4
EXTRN CODE (?B_SWITCH0)
EXTRN CODE (Oem_StartUp)
EXTRN CODE (Core_Init_ClearRam)
EXTRN CODE (Init_ClearRam)
EXTRN CODE (CheckResetSource)
                CALL    ?B_SWITCH0      ; init bank mechanism to code bank 0
                CALL    Oem_StartUp     ; For OME EC start up function          1
                CALL    Core_Init_ClearRam
                CALL    Init_ClearRam                                           2

                PUSH    ACC
                MOV     DPTR,#2006H     ; RSTS Register
                MOVX    A,@DPTR

                MOV     DPTR,#082H      ; RSTStatus
                MOVX    @DPTR,A
                POP     ACC

                CALL    CheckResetSource                                        3

                LJMP    ?C_START                                                4
                END
```

Figure 4-1-1

Startup.a51	
Function / Instruction	Description
Oem_StartUp	This is the first function of OEM firmware feature. Note : No any memory is cleared in this time. [Return] : None [Parameter] : None
Init_ClearRam	In this time, internal ram (20h-BFh) and external ram (000h-0FFh) are cleared. The function can use for clearing other memory if necessary. [Return] : None [Parameter] : None
CheckResetSource	In this function, firmware can check global variable “RSTStatus” to get last reset source. [Return] : None [Parameter] : None
LJMP ?C_START	Jump to main function. [Return] : None [Parameter] : None

Table 4-1-1

4.2 The Main Function EC firmware

The structure of main function is shown in Figure 4-2-1. After initialization, the main function checks the service flags and processing appropriate subroutine. If no any event need to process, EC will enter idle mode. The table 4-2-1 lists general functions of OEM feature.

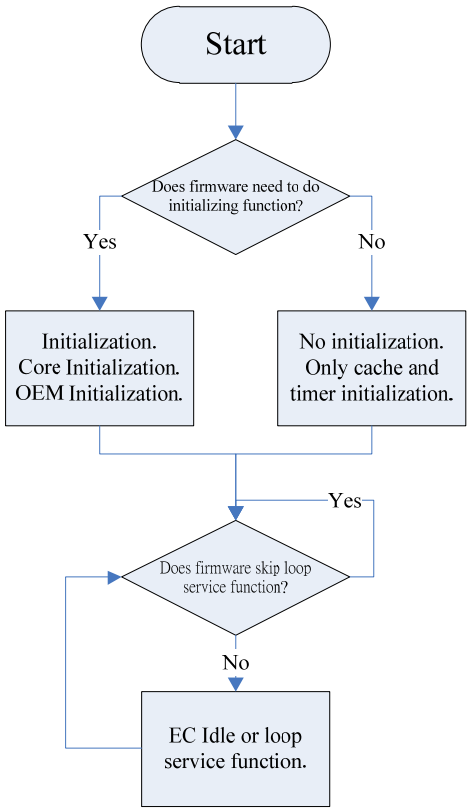


Figure 4-2-1

CORE_MAIN.C	
Function	Description
Hook_ECRetunrMainFuncKeepCondition	Doing initializing function or not. [Return] : [33h] To skip initializing setting function of “main” function. [Parameter] : None
Hook_ECExitFollowMode	If the return value is 33h of “Hook_ECRetunrMainFuncKeepCondition” function, this function will be executed after jump to “main” function. [Return] : None [Parameter] : None
Oem_Initialization	OEM initializing setting function of “main” function. [Return] : None [Parameter] : None
OEM_SkipMainServiceFunc	This function is useful if programmer want to have unique service method of “main” function. [Return] : [FFh] Only “OEM_SkipMainServiceFunc” function will be executed in while(1) loop of main function. [Parameter] : None
Hook_main_service_H	This function will be executed when any service flag “Service” and “Service1” is set. [Return] : None [Parameter] : None

Hook_main_service_L	This function will be executed before leaving “main_service” function. [Return] : None [Parameter] : None
Hook_Only_Timer1msEvent	To perform all timer event function or not. [Return] : [01h] Firmware only executes “Hook_Timer1msEvent” function. Other timer event functions will be ignored. [Parameter] : None
Hook_Timer1msEvent	This function will be executed per 1ms. [Return] : None [Parameter] : 0 ~ 9
Hook_Timer5msEvent	This function will be executed per 5ms. [Return] : None [Parameter] : None
Hook_Timer10msEventA	This function will be executed per 10ms. [Return] : None [Parameter] : None
Hook_Timer10msEventB	This function will be executed per 10ms. [Return] : None [Parameter] : None
Hook_Timer50msEventA	This function will be executed per 50ms. [Return] : None [Parameter] : None
Hook_Timer50msEventB	This function will be executed per 50ms. [Return] : None [Parameter] : None
Hook_Timer50msEventC	This function will be executed per 50ms. [Return] : None [Parameter] : None
Hook_Timer100msEventA	This function will be executed per 100ms. [Return] : None [Parameter] : None
Hook_Timer100msEventB	This function will be executed per 100ms. [Return] : None [Parameter] : None
Hook_Timer100msEventC	This function will be executed per 100ms. [Return] : None [Parameter] : None
Hook_Timer500msEventA	This function will be executed per 500ms. [Return] : None [Parameter] : None
Hook_Timer500msEventB	This function will be executed per 500ms. [Return] : None [Parameter] : None
Hook_Timer500msEventC	This function will be executed per 500ms. [Return] : None [Parameter] : None
Hook_Timer1SecEventA	This function will be executed per second. [Return] : None [Parameter] : None
Hook_Timer1SecEventB	This function will be executed per second. [Return] : None [Parameter] : None
Hook_Timer1SecEventC	This function will be executed per second.

	[Return] : None [Parameter] : None
service_OEM_1	Reserved for OEM feature. If “F_Service_OEM_1” flag is set, this function will be executed. [Return] : None [Parameter] : None
service_OEM_2	Reserved for OEM feature. If “F_Service_OEM_2” flag is set, this function will be executed. [Return] : None [Parameter] : None
service_OEM_3	Reserved for OEM feature. If “F_Service_OEM_3” flag is set, this function will be executed. [Return] : None [Parameter] : None
service_OEM_4	Reserved for OEM feature. If “F_Service_OEM_4” flag is set, this function will be executed. [Return] : None [Parameter] : None

Table 4-2-1

4.3 Interrupt Service Routine

The interrupts of ITE EC is compatible with the interrupts of the original 8032 microcontroller. It has 6 interrupts sources shown in Table 4-3-1.

EC Interrupt List			
Interrupt Source	Interrupt Vector Address	Enable Flag of SFR	The Usage Status of Firmware
External Interrupt 0	0003h	EX0 / IE.0	None
Internal Timer 0	000Bh	ET0 / IE.1	1ms timer for “service_1mS”.
External Interrupt 1	0013h	EX1 / IE.2	INTC interrupt. (Table 4-4)
Internal Timer 1	001Bh	ET1 / IE.3	2ms timer for “service_send”.
Internal Serial Port	0023h	ES0 / IE.4	None
Internal Timer 2	002Bh	ET2 / IE.5	None

Table 4-3-1

INTC mainly collects several interrupts from modules. External interrupt 1 to 8032 is generated by INTC. External interrupts can wakeup 8032 from idle/sleep mode, but internal interrupts can wakeup 8032 from idle mode only. Table 4-3-2 is the list of OEM hook functions of INTC.

INTC OEM Hook Function		
Function	Source	Description
Hook_IRQ_INT1_WKO20	External	WKO[20] [Return] : None [Parameter] : None
Hook_IRQ_INT2_KBCOBE	Internal	KBC Output Buffer Empty Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT3_PMCPMC1OBE	Internal	PMC Output Buffer Empty Interrupt PMC1 Output Buffer Empty Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT4_SMBusD	Internal	SMBus D Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT5_WKINTAD	External	WKINTAD (WKINTA or WKINTD) [Return] : None [Parameter] : None
Hook_IRQ_INT6_WKO23	External	WKO[23] [Return] : None [Parameter] : None
Hook_IRQ_INT7_PWM	Internal	PWM Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT8_ADC	Internal	ADC Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT9_SMBusA	Internal	SMBus A Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT10_SMBusB	Internal	SMBus B Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT11_KBMatrixScan	Internal	KB Matrix Scan Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT12_WKO26	External	WKO[26]

		[Return] : None [Parameter] : None
Hook_IRQ_INT13_WKINTC	External	WKINTC [Return] : None [Parameter] : None
Hook_IRQ_INT14_WKO25	External	WKO[25] [Return] : None [Parameter] : None
Hook_IRQ_INT15_CIR	Internal	CIR Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT16_SMBusC	Internal	SMBus C Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT17_WKO24	External	WKO[24] [Return] : None [Parameter] : None
Hook_IRQ_INT18_PS2Interrupt2	Internal	PS/2 Interrupt 2 [Return] : None [Parameter] : None
Hook_IRQ_INT19_PS2Interrupt1	Internal	PS/2 Interrupt 1 [Return] : None [Parameter] : None
Hook_IRQ_INT20_PS2Interrupt0	Internal	PS/2 Interrupt 0 [Return] : None [Parameter] : None
Hook_IRQ_INT21_WKO22	External	WKO[22] [Return] : None [Parameter] : None
Hook_IRQ_INT22_SMFISemaphore	Internal	SMFI Semaphore Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT23_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT24_KBCIBF	Internal	KBC Input Buffer Full Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT25_PMCPMC1IBF	Internal	PMC Input Buffer Full Interrupt PMC1 Input Buffer Full Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT26_PMC2OBE	Internal	PMC2 Output Buffer Empty Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT27_PMC2IBF	Internal	PMC2 Input Buffer Full Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT28_GINTofGPD5	External	GINT from function 1 of GPD5 [Return] : None [Parameter] : None
Hook_IRQ_INT29_EGPC	Internal	EGPC Interrupt

		[Return] : None [Parameter] : None
Hook_IRQ_INT30_ET1	Internal	External Timer 1 Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT31_WKO21	External	WKO[21] [Return] : None [Parameter] : None
Hook_IRQ_INT32_GPINT0	Internal r	GPINT0 [Return] : None [Parameter] : None
Hook_IRQ_INT33_GPINT1	Internal	GPINT1 [Return] : None [Parameter] : None
Hook_IRQ_INT34_GPINT2	Internal	GPINT2 [Return] : None [Parameter] : None
Hook_IRQ_INT35_GPINT3	Internal	GPINT3 [Return] : None [Parameter] : None
Hook_IRQ_INT36_CIRGPINT	Internal	CIR GPINT [Return] : None [Parameter] : None
Hook_IRQ_INT37_SSPI	Internal	SSPI Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT38_UART1	Internal	UART1 Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT39_UART2	Internal	UART2 Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT40_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT41_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT42_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT43_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT44_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT45_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT46_Null	--	Reserved [Return] : None [Parameter] : None

Hook_IRQ_INT47_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT48_WKO60	External	WKO[60] [Return] : None [Parameter] : None
Hook_IRQ_INT49_WKO61	External	WKO[61] [Return] : None [Parameter] : None
Hook_IRQ_INT50_WKO62	External	WKO[62] [Return] : None [Parameter] : None
Hook_IRQ_INT51_WKO63	External	WKO[63] [Return] : None [Parameter] : None
Hook_IRQ_INT52_WKO64	External	WKO[64] [Return] : None [Parameter] : None
Hook_IRQ_INT53_WKO65	External	WKO[65] [Return] : None [Parameter] : None
Hook_IRQ_INT54_WKO66	External	WKO[66] [Return] : None [Parameter] : None
Hook_IRQ_INT55_WKO67	External	WKO[67] [Return] : None [Parameter] : None
Hook_IRQ_INT56_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT57_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT58_ET2	Internal	External Timer 2 Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT59_DeferredSPIInstruction	Internal	Deferred SPI Instruction Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT60_TMRINTA0	Internal	TMRINTA0 [Return] : None [Parameter] : None
Hook_IRQ_INT61_TMRINTA1	Internal	TMRINTA1 [Return] : None [Parameter] : None
Hook_IRQ_INT62_TMRINTB0	Internal	TMRINTB0 [Return] : None [Parameter] : None
Hook_IRQ_INT63_TMRINTB1	Internal	TMRINTB1 [Return] : None [Parameter] : None
Hook_IRQ_INT64_PMC2EXOBE	Internal	PMC2EX Output Buffer Empty Interrupt

		[Return] : None [Parameter] : None
Hook_IRQ_INT65_PMC2EXIBF	Internal	PMC2EX Input Buffer Full Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT66_PMC3OBE	Internal	PMC3 Output Buffer Empty Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT67_PMC3IBF	Internal	PMC3 Input Buffer Full Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT68_PMC4OBE	Internal	PMC4 Output Buffer Empty Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT69_PMC4IBF	Internal	PMC4 Input Buffer Full Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT70_Null	--	Reserved [Return] : None [Parameter] : None
Hook_IRQ_INT71_I2BRAM	Internal	I2BRAM Interrupt [Return] : None [Parameter] : None
Hook_IRQ_INT72_WKO70	External	WKO[70] [Return] : None [Parameter] : None
Hook_IRQ_INT73_WKO71	External	WKO[71] [Return] : None [Parameter] : None
Hook_IRQ_INT74_WKO72	External	WKO[72] [Return] : None [Parameter] : None
Hook_IRQ_INT75_WKO73	External	WKO[73] [Return] : None [Parameter] : None
Hook_IRQ_INT76_WKO74	External	WKO[74] [Return] : None [Parameter] : None
Hook_IRQ_INT77_WKO75	External	WKO[75] [Return] : None [Parameter] : None
Hook_IRQ_INT78_WKO76	External	WKO[76] [Return] : None [Parameter] : None
Hook_IRQ_INT79_WKO77	External	WKO[77] [Return] : None [Parameter] : None

Table 4-3-2

INTC OEM Control Hook Function	
Function	Description
Hook_EnableInterrupt	To enable interrupt of OEM feature. [Return] : None.

	[Parameter] : None
Hook_OEM_Isr_Int1	This function is useful if programmer want to have unique service method of " INTC ISR. (External interrupt 1) Note : Flag "OEM_Isr_Int1_Request ==1" is necessary. [Return] : None. [Parameter] : None

Table 4-3-3

4.4 LPC I/O 60h/64h Interface

The subroutine “service_pci1” handles input data of 60h/64h port. The appropriate subroutine functions for OEM feature described in Table 4-4-1 and the processing flowchart of interface is shown in Figure 4-4-1.

60h/64h Interface OEM Hook Function	
Function	Description
Hook_60Port	Hook function of system writes data to port 60h. [Return] : None. [Parameter] : [KBHIData]
Hook_64Port	Hook function of system writes command to port 64h. [Return] : None. [Parameter] : [KBHICmd]
Hook_Keyboard_Cmd	Hook function for keyboard command. [Return] : None. [Parameter] : [kbcmd]
Hook_Keyboard_Cmd_Parameter	Hook function for system data of keyboard command. [Return] : None. [Parameter] : [kbcmdp]
Hook_Mouse_D4Cmd	Hook function for KBC command D4h. [Return] : None. [Parameter] : [mscmd]
Hook_Mouse_90Cmd	Hook function for KBC command 90h. [Return] : None. [Parameter] : [mscmd]
Hook_Mouse_91Cmd	Hook function for KBC command 91h. [Return] : None. [Parameter] : [mscmd]
Hook_Mouse_92Cmd	Hook function for KBC command 92h. [Return] : None. [Parameter] : [mscmd]
Hook_Mouse_93Cmd	Hook function for KBC command 93h. [Return] : None. [Parameter] : [mscmd]
Hook_A20ON	Hook function for GateA20 high. [Return] : None. [Parameter] : None
Hook_A20OFF	Hook function for GateA20 low. [Return] : None. [Parameter] : None
Hook_KBRSTON	Hook function for RC_IN (reset line) low. [Return] : None. [Parameter] : None
Hook_KBRSTOFF	Hook function for RC_IN (reset line) high. [Return] : None. [Parameter] : None
Hook_NUMLED_ON	Hook function for number lock LED on. [Return] : None. [Parameter] : None
Hook_NUMLED_OFF	Hook function for number lock LED off. [Return] : None. [Parameter] : None

Hook_CAPLED_ON	Hook function for Caps lock LED on. [Return] : None. [Parameter] : None
Hook_CAPLED_OFF	Hook function for Caps lock LED off. [Return] : None. [Parameter] : None

Table 4-4-1

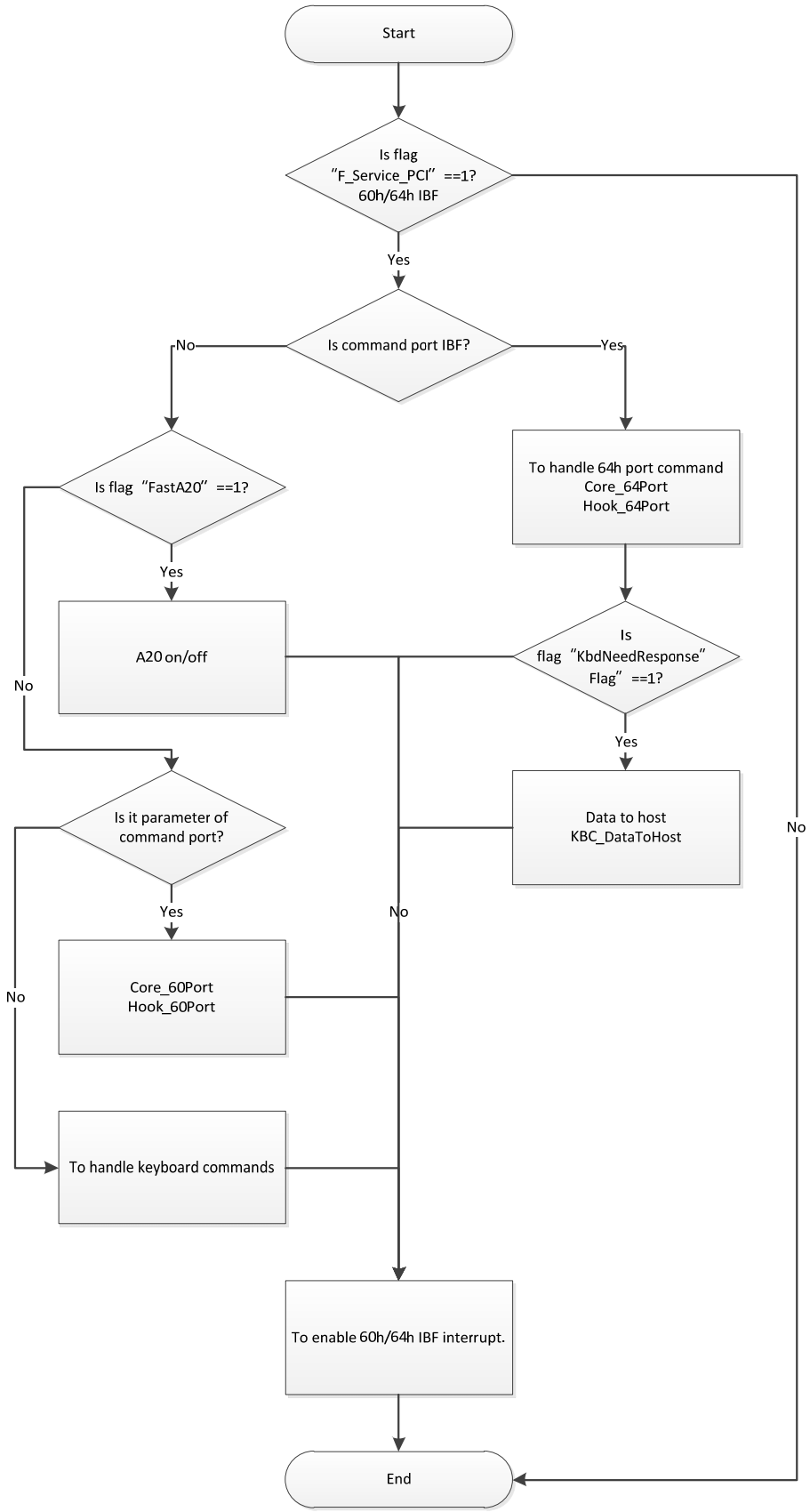

Start
Is flag
"F_Service_PCI" ==1?
60h/64h IBF
Yes
No
Is command port IBF?
Yes
Is flag "FastA20" ==1?
To handle 64h port command
Core_64Port
Hook_64Port
Yes
A20 on/off
Is
flag "KbdNeedResponse"
Flag" ==1?
No
Yes
Is it parameter of
command port?
Data to host
KBC_DataToHost
No
Yes
No
Core_60Port
Hook_60Port
No
To handle keyboard commands
To enable 60h/64h IBF interrupt.
End

Figure 4-4-1

We support KBC commands described in Table 4-4-2.

KBC Commands	
Command	Description
20h	Read command byte. (Ccb42) bit7 - Reserved bit6 - Convert Scan Codes bit5 - Auxiliary Disabled bit4 - Keyboard Disabled bit3 - Reserved bit2 - System Flag bit1 - Auxiliary Interrupt Enabled bit0 - Keyboard Interrupt Enabled
60h	Write command byte. (Ccb42) bit7 - Reserved bit6 - Convert Scan Codes bit5 - Auxiliary Disabled bit4 - Keyboard Disabled bit3 - Reserved bit2 - System Flag bit1 - Auxiliary Interrupt Enabled bit0 - Keyboard Interrupt Enabled
90h	PS2 multiplexing mode AUX port0. Source bit : 00b Reserved.
91h	PS2 multiplexing mode AUX port1. Source bit : 01b EC GPIOF.0, GPIOF.1
92h	PS2 multiplexing mode AUX port2. Source bit : 10b EC GPIOF.2, GPIOF.3
93h	PS2 multiplexing mode AUX port3. Source bit : 11b EC GPIOF.4, GPIOF.5
A7h	Disable auxiliary device. Bit5 of command byte.
A8h	Enable auxiliary device. Bit5 of command byte.
A9h	Test AUX device interface. Note : Always return no error to system. (00h)
AAh	EC self test Note : Always return pass to system. (55h)
ABh	Test keyboard interface. Note : Always return no error to system. (00h)
ADh	Disable keyboard interface. Bit4 of command byte.
AEh	Enable keyboard interface. Bit4 of command byte.
C0h	Emulate reading the 8042 Input port and send data to the system. Just return the compatibility value for now.
D0h	Send 8042 Output port value to the system. Emulates data since there's no real Output port.
D1h	On/Off GateA20 line based on the system data (Port 60h) bit1.

	Bit1 ==1 : hook function “Hook_A20ON” Bit1 ==0 : hook function “Hook_A20OFF”
D2h	Send data to the system as if it came from the keyboard. (Port 60h)
D3h	Send data to the system as if it came from the auxiliary device. (Port 60h)
D4h	Send next received byte of data from system (Port 60h) to auxiliary device.
E0h	Reports the state of the test inputs. Always return 00h to the system.
F0~FFh	Even command : Pulse RC_IN (the reset line). Hook function : “Hook_KBRSTON” and “Hook_KBRSTOFF”. Odd command : Do no thing.

Table 4-4-2

The table 4-4-3 and 4-4-4 list keyboard commands processed by the EC.

One Byte Command for Internal Keyboard.	
Command	Description
ECh	Return acknowledge only. Return to the system : FAh
EEh	ECHO command. Return to the system : EEh
F2h	Read ID. Return to the system : FAh, ABh, 83h or 41h
F4h	Enable. Return to the system : FAh
F5h	Disable. Return to the system : FAh
F6h	Set default. Return to the system : FAh
F7h	Return acknowledge only. Return to the system : FAh
F8h	Return acknowledge only. Return to the system : FAh
F9h	Return acknowledge only. Return to the system : FAh
FAh	Return acknowledge only. Return to the system : FAh
FBh	Return acknowledge only. Return to the system : FAh
FFh	Reset keyboard. Return to the system : FAh, AAh
Other	To request system resend. Return to the system : FEh

Table 4-4-3

Two Byte Command for Internal Keyboard.	
Command	Description
EDh	To update LEDs command.

	Return to the system : FAh
F0h	Select alternate scan code set. Return to the system : FAh
F3h	Set typematic rate/delay. Return to the system : FAh

Table 4-4-4

4.5 LPC I/O 62h/66h Interface

The subroutine “service_pci2” handles input data of 62h/66h port. The appropriate subroutine functions for OEM feature described in Table 4-5-1 and the processing flowchart of interface is shown in Figure 4-5-1. Never use internal timer 1 in “Hook_66Port” and “Hook_62Port” function because internal timer1 is used for total burst time.

62h/66h Interface OEM Hook Function	
Function	Description
Hook_66Port	Hook function of system writes command to port 66h. Never use internal timer 1 in this function and appropriate subroutine. [Return] : None [Parameter] : PM1Cmd
Hook_62Port	Hook function of system writes data to port 62h. Never use internal timer 1 in this function and appropriate subroutine. [Return] : None [Parameter] : PM1Data
Hook_ACPICommand	This function will be executed before leaving “service_pci2” function. [Return] : None [Parameter] : None
OEM_ACPI_Gen_Int	Pulse SCI in this function. [Return] : None [Parameter] : None
ResetSCIEvent	Reset SCI event flag and clearing SCI buffer. [Return] : None [Parameter] : None
ECQEvent	Setup SCI number with SCI interrupt. The Host uses this number to determine the cause of the SCI. [Return] : None [Parameter] : sci_number sci_mode
Hook_ReadMapECSpace	According to parameter “MapIndex” and return appropriate data of EC RAM to host. [Return] : Return appropriate data of EC RAM to host. [Parameter] : MapIndex
Hook_WriteMapECSpace	According to parameter “MapIndex” and “data1”. Writing “data1” to offset “MapIndex” of EC ram. [Return] : None [Parameter] : MapIndex data1
Hook_SCION	Hook function for SCI set. [Return] : None [Parameter] : None
Hook_SCIOFF	Hook function for SCI release. [Return] : None [Parameter] : None

Table 4-5-1

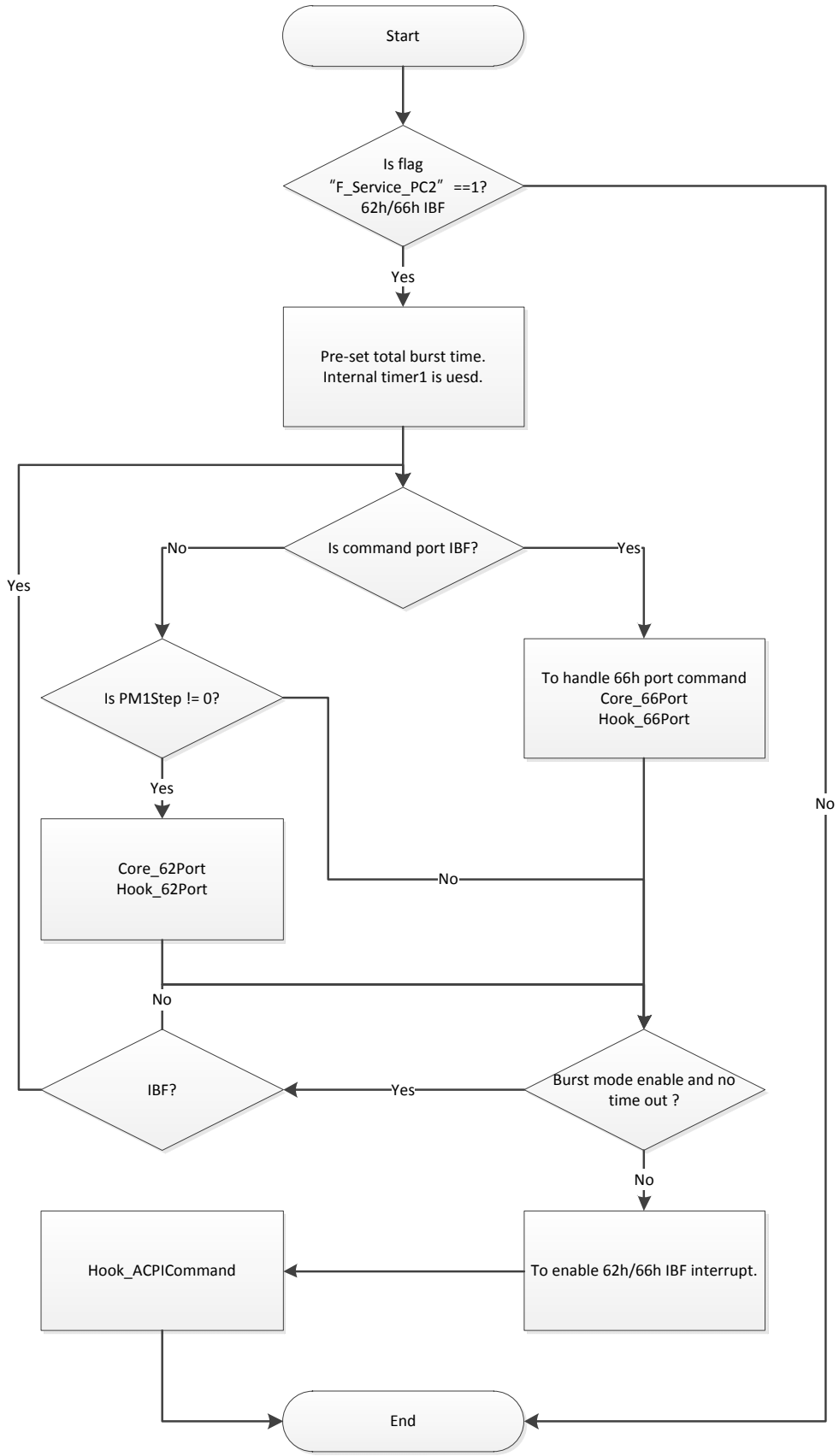

Start
Is flag
"F_Service_PC2" ==1?
62h/66h IBF
Yes
No
Pre-set total burst time.
Internal timer1 is uesd.
Is command port IBF?
No
Yes
Yes
Is PM1Step != 0?
To handle 66h port command
Core_66Port
Hook_66Port
Yes
No
Core_62Port
Hook_62Port
No
No
IBF?
Yes
Burst mode enable and no
time out ?
No
Hook_ACPICommand
To enable 62h/66h IBF interrupt.
End

Figure 4-5-1

ACPI embedded controller interface is supported and command set lists in Table 4-5-2.

ACPI Embedded Controller Commands	
Command	Description
80h	Read EC RAM. To modify “Hook_ReadMapECSpace” is necessary.
81h	Write EC RAM. To modify “Hook_ReadMapECSpace” is necessary.
82h	Burst enable.
83h	Burst disable.
84h	Query EC event.

Table 4-5-2

We support an extended command set described in Table 4-5-3 for other purposes.

62h/66h Extended Commands	
Command	Description
92h	Read external RAM and registers via 62h/66h interface.
93h	Write external RAM and registers via 62h/66h interface.
DCh	ITE flash utility (IU.EXE IU)
F0h	ITE keyboard matrix generation utility (IU.EXE KU)

Table 4-5-3

4.6 PS2 Interface

We provide two selection of PS2 interface in the “OEM_Project.H”. If only one device is used, please modify selection “TouchPad_only” to true and disabling multiplexing mode. If two or more PS2 devices are used, please disable “TouchPad_only” selection and enabling multiplexing mode. EC firmware supports device hot plug and hot swap if disable “TouchPad_only” selection. If “TouchPad_only” selection is true, several suggestions as follow. The related functions are list in Table 4-6-1.

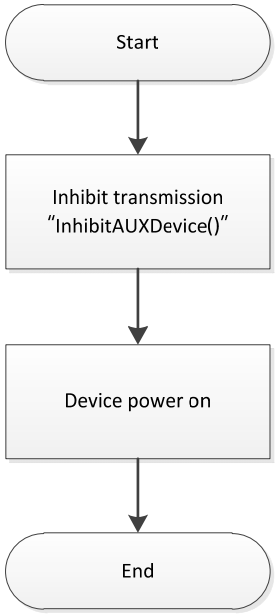


Figure 4-6-1 Recommend sequence when device power on.

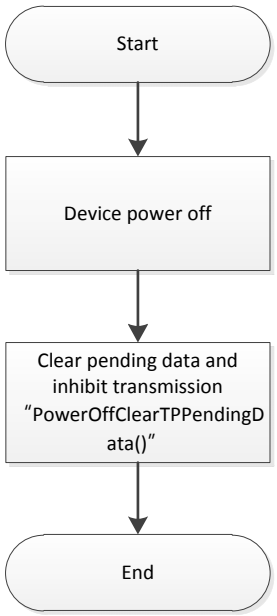


Figure 4-6-2 Recommend sequence when device power off.

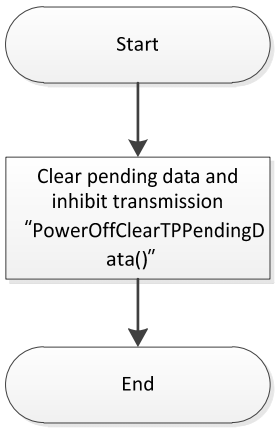


Figure 4-6-3 Recommend sequence when system warm boot.

Function	Description
Hook_ECRespondtoMouseDriver	The firmware can response port 64h D4h command by this function if no any device exist. [Return] : None [Parameter] : mscmd
Hook_service_ps2	Hook function of service_ps2. [Return] : None [Parameter] : ps2_channel 0, 1, or 2 ps2_data receive data.
Hook_DisablePS2Port_0	Disable port 0. Firmware sends F5h command to device. Note : Two condition must be true. 1. "MouseDriverIn" flag was set. 2. Device was powered on. [Return] : None [Parameter] : None
Hook_DisablePS2Port_1	Disable port 1. Firmware sends F5h command to device. Note : Two condition must be true. 1. "MouseDriverIn" flag was set. 2. Device was powered on. [Return] : None [Parameter] : None
Hook_DisablePS2Port_2	Disable port 2. Firmware sends F5h command to device. Note : Two condition must be true. 1. "MouseDriverIn" flag was set. 2. Device was powered on. [Return] : None [Parameter] : None
Hook_EnablePS2Port_0	Enable port 0. Firmware sends F4h command to device. Note : Two condition must be true. 1. "MouseDriverIn" flag was set. 2. Device was powered on. [Return] : None [Parameter] : None
Hook_EnablePS2Port_1	Enable port 1. Firmware sends F4h command to device. Note : Two condition must be true. 1. "MouseDriverIn" flag was set. 2. Device was powered on.

	[Return] : None [Parameter] : None
Hook_EnablePS2Port_2	Enable port 2. Firmware sends F4h command to device. Note : Two condition must be true. 1. "MouseDriverIn" flag was set. 2. Device was powered on. [Return] : None [Parameter] : None
Hook_TPOnlyLowLevelFunc	Hook function of “TPOnlyLowLevelFunc”. [Return] : None [Parameter] : None
PowerOffClearTPPendingData	To clear pending data and inhibiting interface. [Return] : None [Parameter] : None
Send2PortNWait	Sending command to device and waiting transaction done. (30 ms time-out) The global variable “PS2IFAck” saves the last response data of device. Please carefully consider when to use. The function isn’t used in most cases. [Return] : 0 : Transaction done 1 : Transaction fail [Parameter] : PortNum 0, 1, 2 cmd device command. bytecunt

Table 4-6-1

The processing flow chart of PS2 interface is shown in Figure 4-6-4 and the processing flow chart of mouse pending data is shown in Figure 4-6-5.

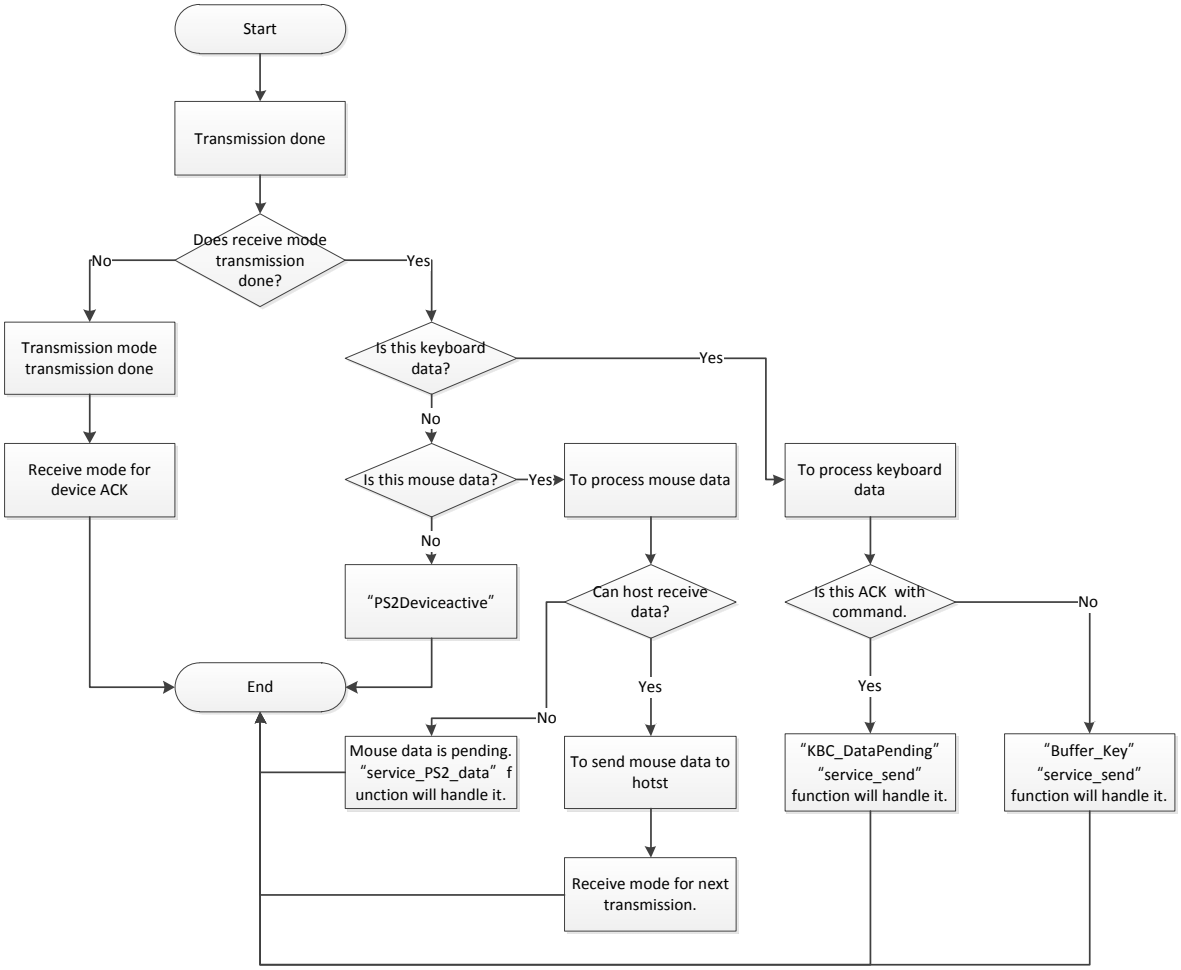


Figure 4-6-4

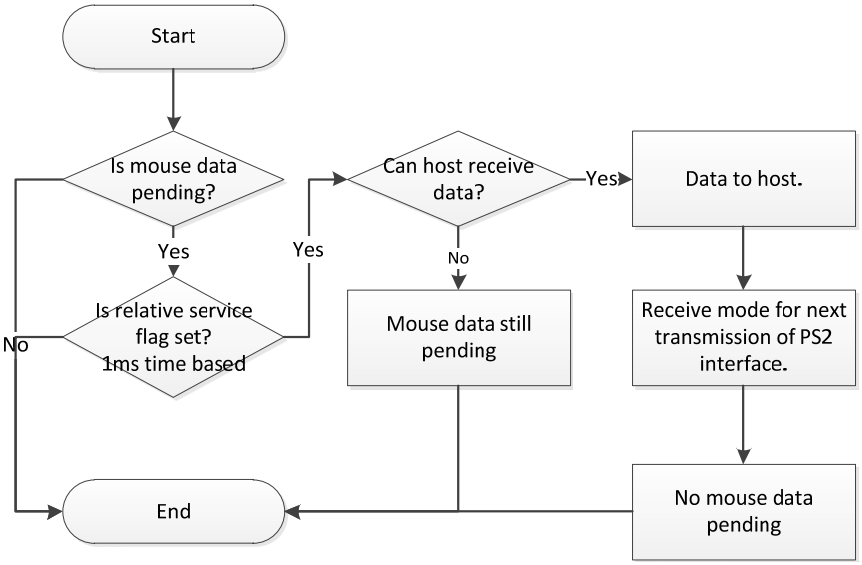


Figure 4-6-5

4.7 Internal Keyboard Scanner

ITE provide a useful utility to generate keyboard matrix table of internal keyboard. Just a USB keyboard and connecting internal keyboard then follow the prompts to get new keyboard matrix table finally replace original keyboard matrix table. If you want to create keyboard matrix table by yourself, you can refer to section 4.7.1 ~ 4.7.4. There are some application hook functions in Table 4-7-1, you can refer to it.

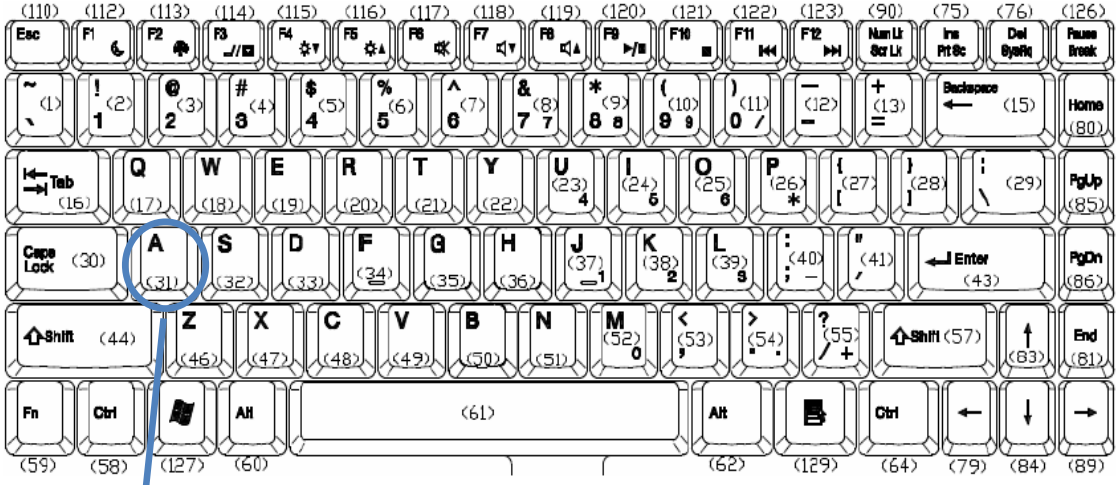
Function	Description
OEM_Hook_Send_Key	Hook function for “Send_Key” function. [Return] : None [Parameter] : table_entry : value of keyboard matrix table. event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
OEM_Hook_Skip_Send_Key	If this function returns FFh, the “Send_Key” function will be break. (no any scan code to host) [Return] : 00h FFh [Parameter] : None
Hook_keyboard	Keyboard hook function (KSO0 - KSO15) [Return] : None [Parameter] : KSIValus keyboard KSI KSOValus keyboard KSO
Et_Hook_keyboard	Keyboard hook function (KSO16 - KSO17 and GPIO KSO) [Return] : None [Parameter] : KSIValus keyboard KSI KSOValus keyboard KSO
Hook_Setup_Scanner_Pntr	To define the possible scanner tables. [Return] : None [Parameter] : None
Hook_SetGPIOScanPinH	Hook function of setting GPIO scan pin to high. [Return] : None [Parameter] : None
Hook_SetGPIOScanPinL	Hook function of setting GPIO scan pin to low. [Return] : None [Parameter] : None
Hook_SetGPIOScanPinCtrl	Hook function of setting GPIO scan pin output. [Return] : None [Parameter] : None
Hook_Fn_Key_Make	Hook function for Fn key make. [Return] : None [Parameter] : None
Hook_Fn_Key_Break	Hook function for Fn key break. [Return] : None [Parameter] : None
Hook_calc_index_comb_BIT6	Return 0xFF --> sskey2_A2_table contains bit6 [Return] : 00h FFh [Parameter] : None
Hook_calc_index_comb_BIT7	Return 0xFF --> sskey2_A2_table contains bit7 [Return] : 00h FFh [Parameter] : None
HotKey_Fn_F1	Hook function for Fn + F1

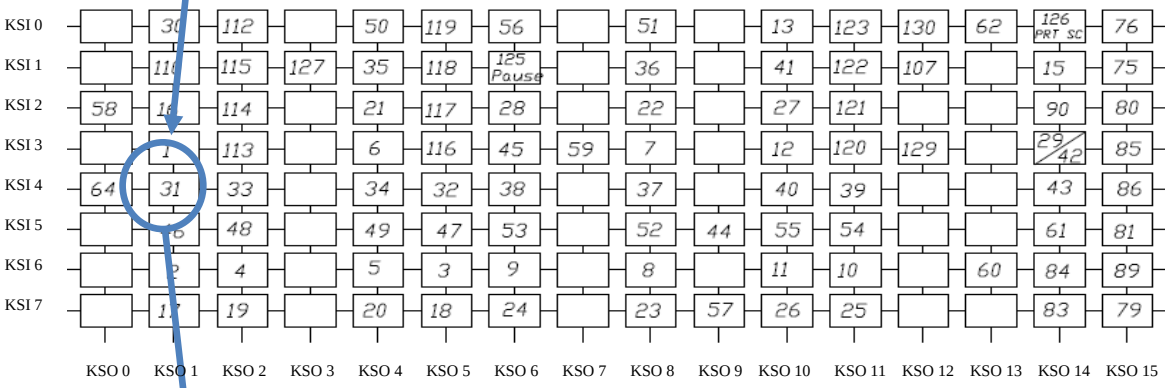
	[Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F2	Hook function for Fn + F2 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F3	Hook function for Fn + F3 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F4	Hook function for Fn + F4 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F5	Hook function for Fn + F5 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F6	Hook function for Fn + F6 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F7	Hook function for Fn + F7 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F8	Hook function for Fn + F8 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F9	Hook function for Fn + F9 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F10	Hook function for Fn + F10 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F11	Hook function for Fn + F11 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_F12	Hook function for Fn + F12 [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_ESC	Hook function for Fn + ESC [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_UP	Hook function for Fn + Up Arrow [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_DOWN	Hook function for Fn + Down Arrow [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_LEFT	Hook function for Fn + Left Arrow [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_RIGHT	Hook function for Fn + Right Arrow [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”
HotKey_Fn_Ins	Hook function for Fn + Ins [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”

HotKey_Fn_Del	Hook function for Fn + Delete [Return] : None [Parameter] : event “MAKE_EVENT”, “BREAK_EVENT”, or “REPEAT_EVENT”

Table 4-7-1

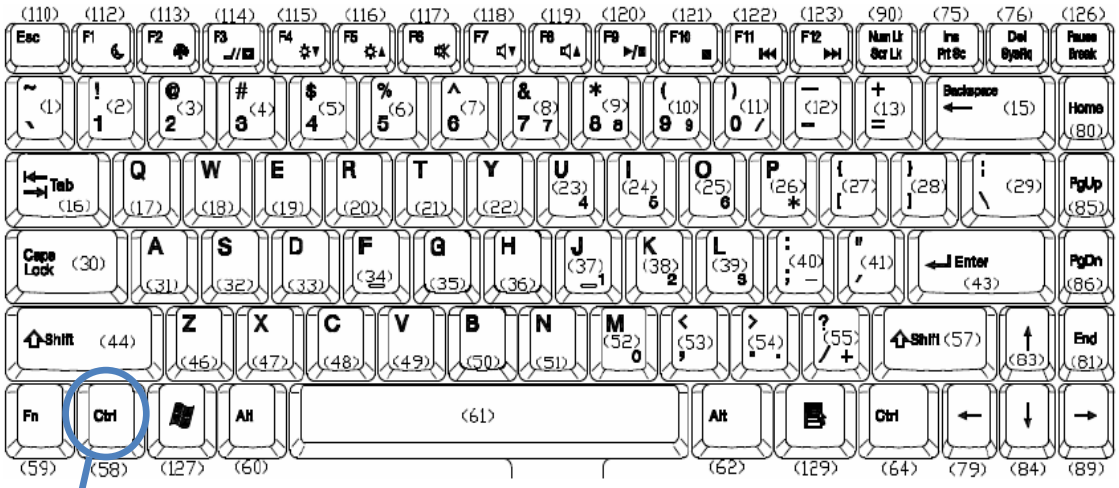
4.7.1 Simple Code (Matrix Index 1 ~ 7Fh)

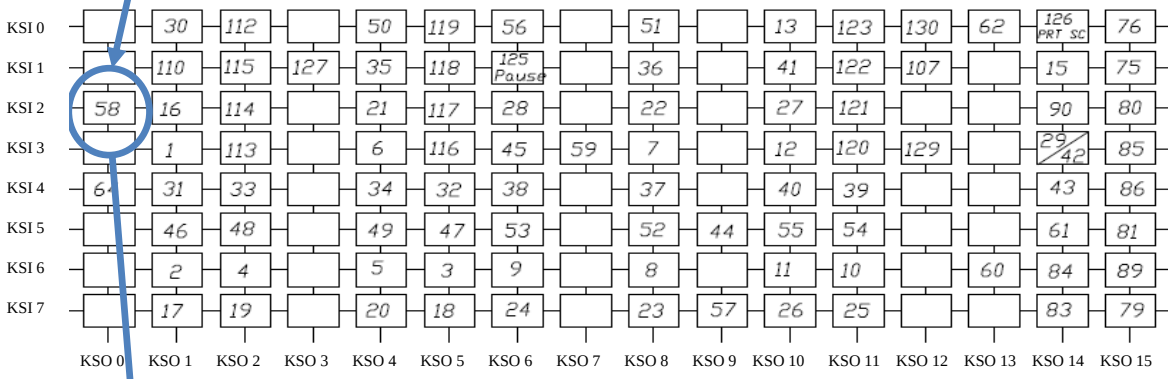





```
const unsigned char code Keyboard_Tables[] =
{
//  0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
    0x00,0x58,0xAD,0x00,0x32,0xB4,0x51,0x00,0x31,0x00,0x55,0xB8,0x00,0x8B,0xF2,0xE2,
    0x00,0xCC,0xB0,0x82,0x34,0xB3,0x00,0x00,0x33,0x00,0x52,0xB7,0x00,0x00,0x66,0xE3,
    0x8C,0x0D,0xAF,0x00,0x2C,0xB2,0x5B,0x00,0x35,0x00,0x54,0xB6,0x00,0x00,0x77,0xE8,
    0x00,0x0E,0xAE,0x00,0x2E,0xB1,0x61,0x8E,0x36,0x00,0x4E,0xB5,0x84,0x00,0x5D,0xE9,
    0x8D,0x1C,0x23,0x00,0x2B,0x1B,0x42,0x00,0x3B,0x00,0x4C,0x4B,0x00,0x00,0x5A,0xEA,
    0x00,0x1A,0x21,0x00,0x2A,0x22,0x41,0x00,0x3A,0x88,0x4A,0x49,0x00,0x00,0x29,0xEB,
    0x00,0x16,0x26,0x00,0x25,0x1E,0x3E,0x00,0x3D,0x00,0x45,0x46,0x00,0x8A,0x99,0x9B,
    0x00,0x15,0x24,0x00,0x2D,0x1D,0x43,0x00,0x3C,0xB9,0x4D,0x44,0x00,0x00,0x98,0x9A,
};
```

4.7.2 (Matrix Index 80h ~ 9Bh)

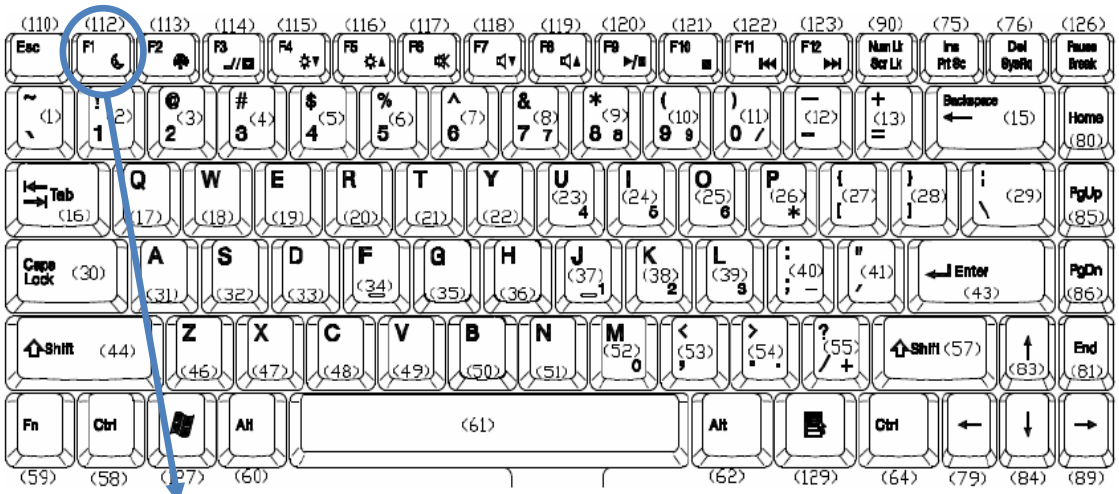


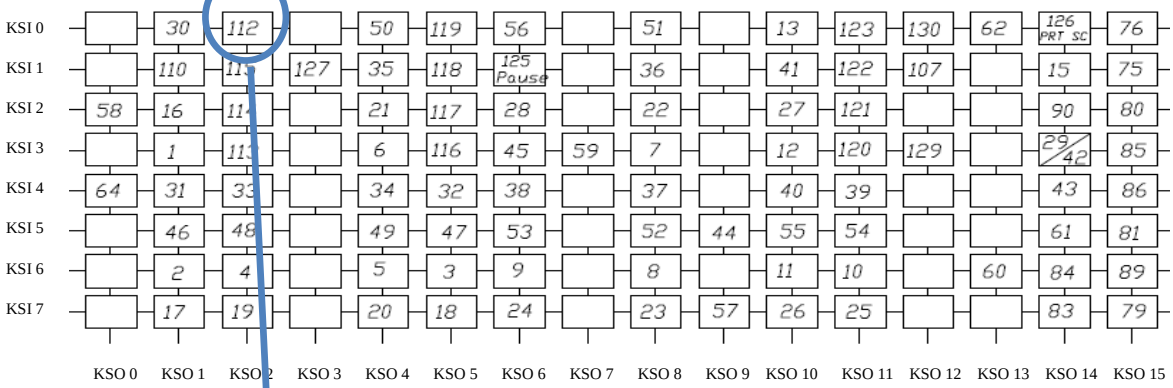



```
const unsigned char code Keyboard_Tables[] =
{
//   0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
    0x00,0x58,0xAD,0x00,0x32,0xB4,0x51,0x00,0x31,0x00,0x55,0xB8,0x00,0x8B,0xF2,0xE2,
    0x00,0xEC,0xB0,0x82,0x34,0xB3,0x00,0x00,0x33,0x00,0x52,0xB7,0x00,0x00,0x66,0xE3,
    0x8C,0x0D,0xAF,0x00,0x2C,0xB2,0x5B,0x00,0x35,0x00,0x54,0xB6,0x00,0x00,0x77,0xE8,
    0x00,0x0E,0xAE,0x00,0x2E,0xB1,0x61,0x8E,0x36,0x00,0x4E,0xB5,0x84,0x00,0x5D,0xE9,
    0x8D,0x1C,0x23,0x00,0x2B,0x1B,0x42,0x00,0x3B,0x00,0x4C,0x4B,0x00,0x00,0x5A,0xEA,
    0x00,0x1A,0x21,0x00,0x2A,0x22,0x41,0x00,0x3A,0x88,0x4A,0x49,0x00,0x00,0x29,0xEB,
    0x00,0x16,0x26,0x00,0x25,0x1E,0x3E,0x00,0x3D,0x00,0x45,0x46,0x00,0x8A,0x99,0x9B,
    0x00,0x15,0x24,0x00,0x2D,0x1D,0x43,0x00,0x3C,0xB9,0x4D,0x44,0x00,0x00,0x98,0x9A,
};
```

sskey3_80_table[]

4.7.3 (Matrix Index 9Ch ~ DFh)

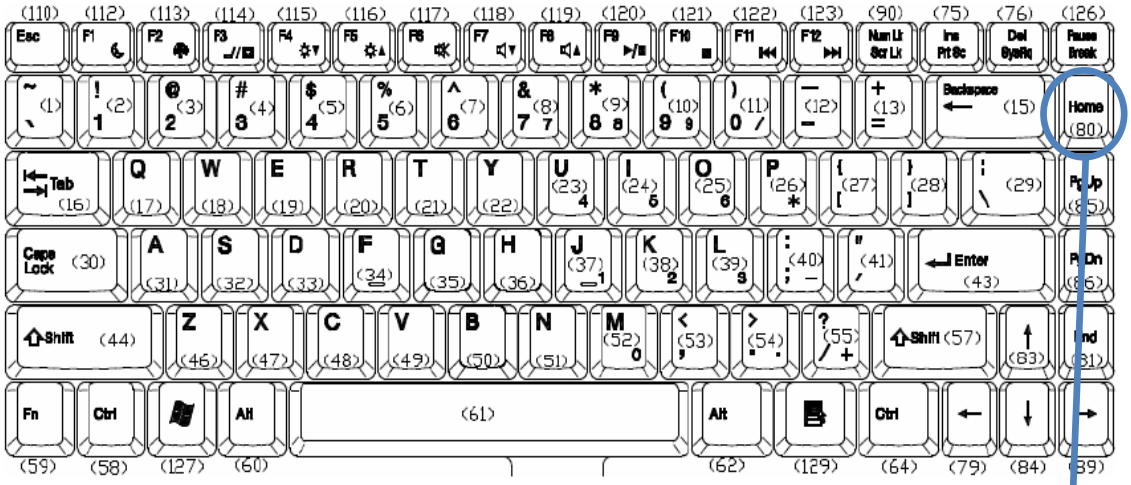





```
const unsigned char code Keyboard_Tables[] =
{
//   0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
    0x00,0x58,0xAD,0x00,0x32,0xB4,0x51,0x00,0x31,0x00,0x55,0xB8,0x00,0x8B,0xF2,0xE2,
    0x00,0xEC,0xB0,0x82,0x34,0xB3,0x00,0x00,0x33,0x00,0x52,0xB7,0x00,0x00,0x66,0xE3,
    0x8C,0x0D,0xAF,0x00,0x2C,0xB2,0x5B,0x00,0x35,0x00,0x54,0xB6,0x00,0x00,0x77,0xE8,
    0x00,0x0E,0xAE,0x00,0x2E,0xB1,0x61,0x8E,0x36,0x00,0x4E,0xB5,0x84,0x00,0x5D,0xE9,
    0x8D,0x1C,0x23,0x00,0x2B,0x1B,0x42,0x00,0x3B,0x00,0x4C,0x4B,0x00,0x00,0x5A,0xEA,
    0x00,0x1A,0x21,0x00,0x2A,0x22,0x41,0x00,0x3A,0x88,0x4A,0x49,0x00,0x00,0x29,0xEB,
    0x00,0x1C,0x26,0x00,0x25,0x1E,0x3E,0x00,0x3D,0x00,0x45,0x46,0x00,0x8A,0x99,0x9B,
    0x00,0x15,0x24,0x00,0x2D,0x1D,0x43,0x00,0x3C,0xB9,0x4D,0x44,0x00,0x00,0x98,0x9A,
};
```

sskey2_A2_table[]

4.7.4 (Matrix Index E0h ~ FFh)



	KSO 0	KSO 1	KSO 2	KSO 3	KSO 4	KSO 5	KSO 6	KSO 7	KSO 8	KSO 9	KSO 10	KSO 11	KSO 12	KSO 13	KSO 14	KSO 15
KSI 0		30	112		50	119	56		51		13	123	130	62	126 PRT SC	76
KSI 1		110	115	127	35	118	125 Pause		36		41	122	107		15	
KSI 2	58	16	114		21	117	28		22		27	121			90	80
KSI 3		1	113		6	116	45	59	7		12	120	129		29/42	
KSI 4	64	31	33		34	32	38		37		40	39			43	
KSI 5		46	48		49	47	53		52	44	55	54			61	
KSI 6		2	4		5	3	9		8		11	10		60	84	
KSI 7		17	19		20	18	24		23	57	26	25			83	

```
const unsigned char code Keyboard_Tables[] =
{
//   0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
    0x00,0x58,0xAD,0x00,0x32,0xB4,0x51,0x00,0x31,0x00,0x55,0xB8,0x00,0x8B,0xF2,0xE2,
    0x00,0xEC,0xB0,0x82,0x34,0xB3,0x00,0x00,0x33,0x00,0x52,0xB7,0x00,0x00,0x66,0xE3,
    0x8C,0x0D,0xAF,0x00,0x2C,0xB2,0x5B,0x00,0x35,0x00,0x54,0xB6,0x00,0x00,0x77,0xE8,
    0x00,0x0E,0xAE,0x00,0x2E,0xB1,0x61,0x8E,0x36,0x00,0x4E,0xB5,0x84,0x00,0x5D,0xE9,
    0x8D,0x1C,0x23,0x00,0x2B,0x1B,0x42,0x00,0x3B,0x00,0x4C,0x4B,0x00,0x00,0x5A,0xEA,
    0x00,0x1A,0x21,0x00,0x2A,0x22,0x41,0x00,0x3A,0x88,0x4A,0x49,0x00,0x00,0x29,0xEB,
    0x00,0x16,0x26,0x00,0x25,0x1E,0x3E,0x00,0x3D,0x00,0x45,0x46,0x00,0x8A,0x99,0x9B,
    0x00,0x15,0x24,0x00,0x2D,0x1D,0x43,0x00,0x3C,0xB9,0x4D,0x44,0x00,0x00,0x98,0x9A,
};
```

sskey2_overlay_table

4.8 SMBus Interface

The SMBus related functions are listed in Table 4-8-1 and controlling flow chart are shown in Figure 4-8-1 ~ Figure 4-8-6. The firmware will wait a result of SMBus transmission. If you don't want the firmware waiting for the result of SMBuc transmission, please refer to "example" version of EC firmware.

Function	Description
bRWSMBus	[Return] : 0 : protocol fail 1 : protocol OK [Parameter] : BYTE Channel 0, 1, 2, 3 BYTE Protocol "SMbusWB", "SMbusRB", "SMbusWW" , "SMbusRW" BYTE Addr device address BYTE Comd command XBYTE *Var external ram pointer for read/write data BYTE PECSupport PEC byte support, only write protocol.
bRSMBusBlock	[Return] : 0 : protocol fail 1 : protocol OK [Parameter] : BYTE Channel 0, 1, 2, 3 BYTE Protocol SMbusRBK BYTE Addr device address BYTE Comd command XBYTE *Var external ram pointer for read data
bWSMBusBlock	[Return] : 0 : protocol fail 1 : protocol OK [Parameter] : BYTE Channel 0, 1, 2, 3 BYTE Protocol SMbusWBK BYTE Addr device address BYTE Comd command XBYTE *Var external ram pointer for write data BYTE ByteCont byte count BYTE PECsupport PEC byte support
bSMBusSendByte	[Return] : 0 : protocol fail 1 : protocol OK [Parameter] : BYTE Channel 0, 1, 2, 3 BYTE Addr device address BYTE SData write data
bSMBusReceiveByte	[Return] : 0 : protocol fail 1 : protocol OK [Parameter] : BYTE Channel 0, 1, 2, 3 BYTE Addr device address XBYTE *Var external ram pointer for read data

Table 4-8-1

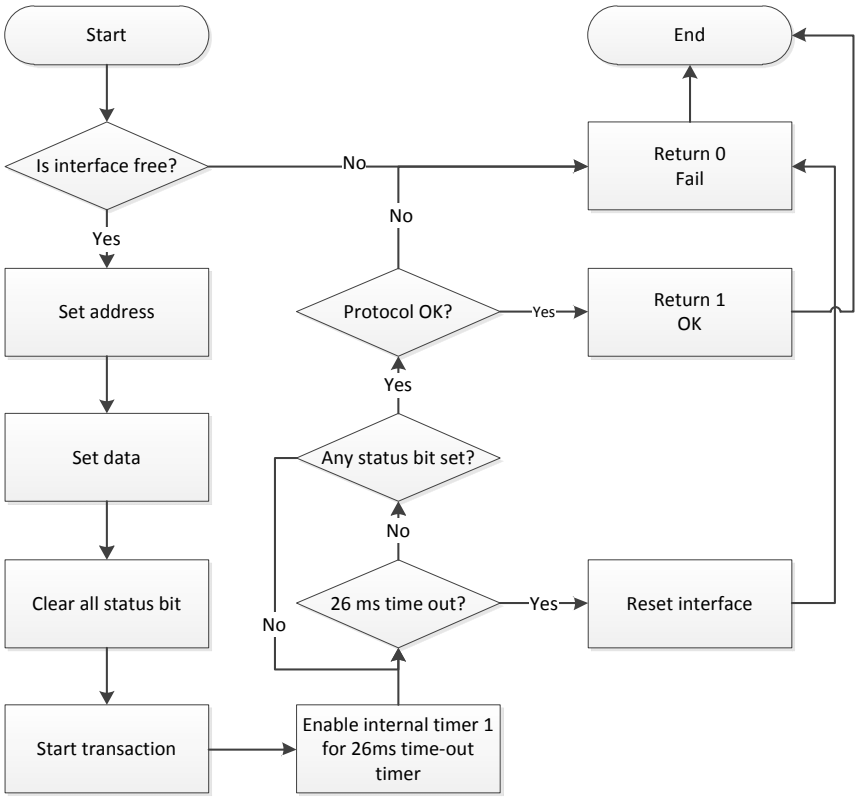


Figure 4-8-1 Send byte flow chart

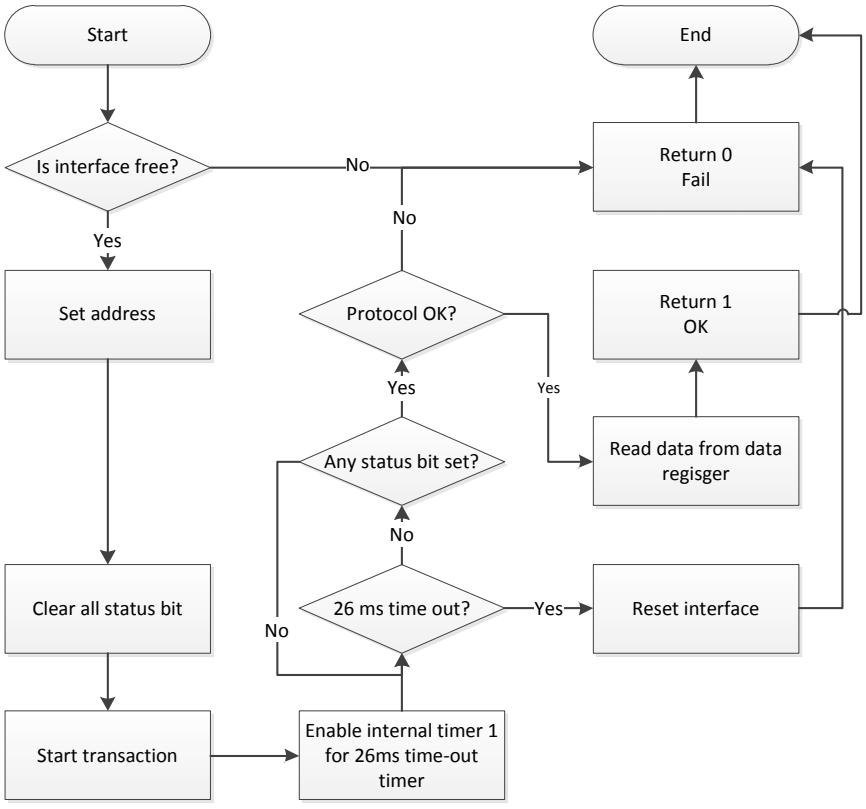


Figure 4-8-2 Receive byte flow chart

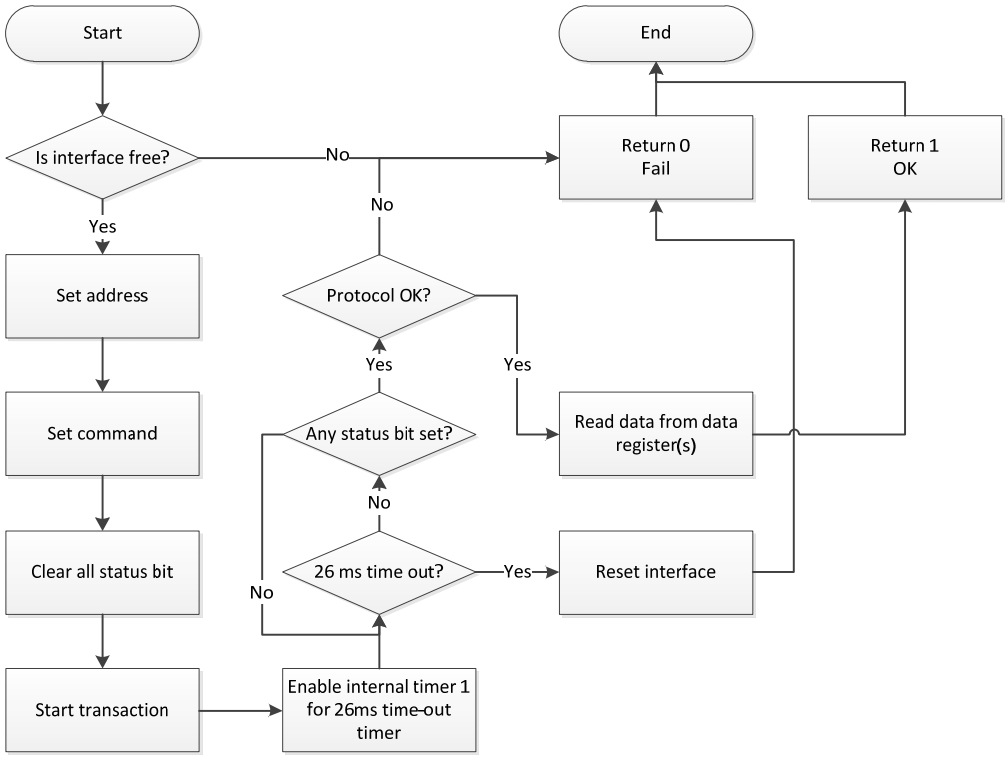


Figure 4-8-3 Read byt /word flow chart

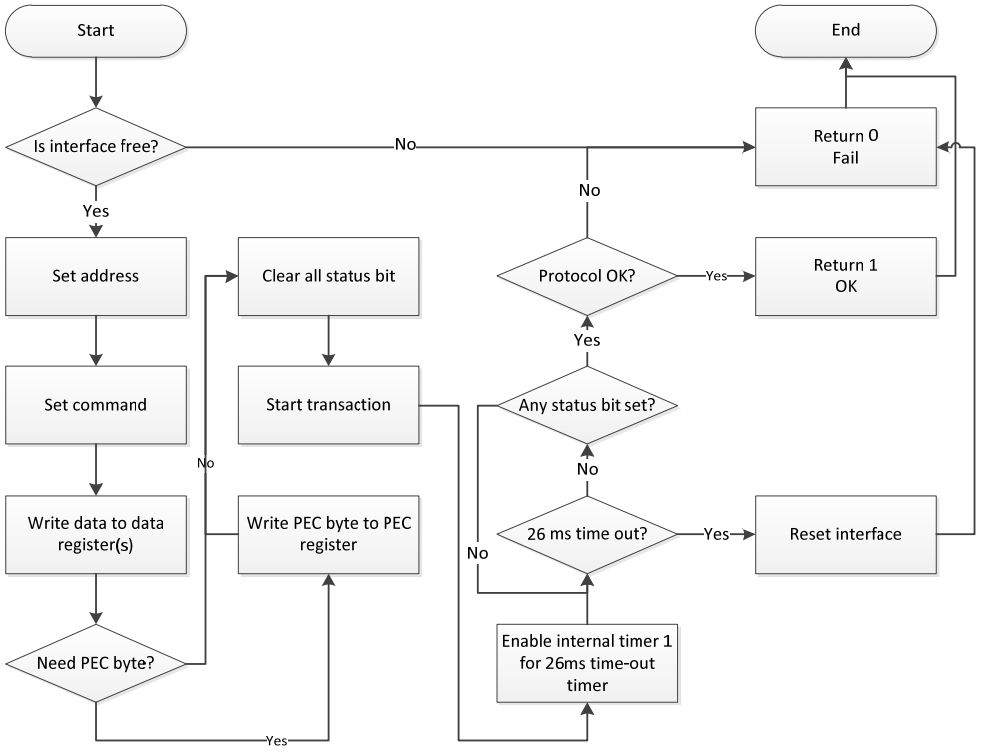


Figure 4-8-4 Write byte/word flow chart

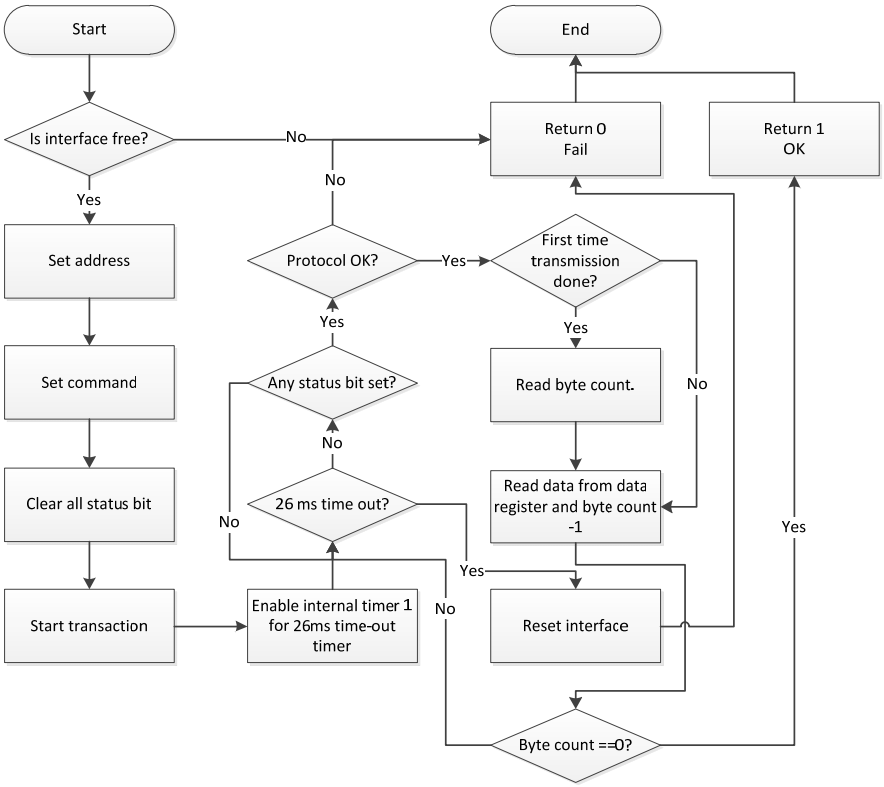


Figure 4-8-5 Block read

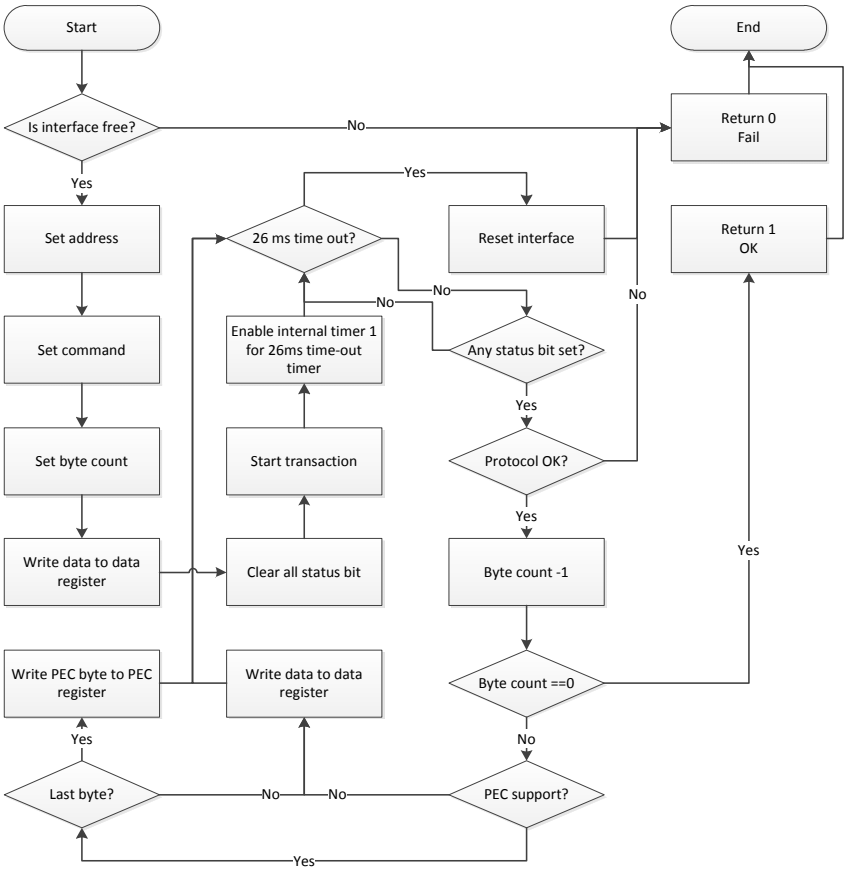


Figure 4-8-6 Block Write

4.9 SPI Read/Write Interface

The firmware can read/write all space of SPI by EC indirect follow mode. The Table 4-9-1 lists some application of EC indirect follow mode. Reading SPI by EC indirect read mode is better choice than EC indirect follow mode.

Function	Description
Do_SPI_Write_Status	Sending 01h (write status) instruction to SPI with parameter “statusvalue”. [Return] : None [Parameter] : BYTE statusvalue
Do_SPI_Read_Status	Reading status register of SPI. Global variable “SPIReadStatus” stores return value of SPI. [Return] : None [Parameter] : None
Do_SPI_Read_ID	Reading SPI ID by 9Fh instruction. Array “SPIIDBuf” stores ID of SPI. [Return] : None [Parameter] : None
Do_SPI_Read_ID_CmdAB	Reading SPI ID by ABh instruction. Array “SPIIDBuf” stores ID of SPI. [Return] : None [Parameter] : None
Do_SPI_Erase	Sending erase command to SPI. Only support “SPICmd_Erase4Kbyte”, “SPICmd_Erase32Kbyte”, and “SPICmd_Erase64Kbyte”. [Return] : None [Parameter] : BYTE EraseCmd BYTE Addr2 A23 ~ A16 BYTE Addr1 A15 ~ A8 BYTE Addr0 A7~ A10
Do_SPI_Write_256Bytes	SPI write 256 bytes. [Return] : None [Parameter] : XBYTE *DataPointer BYTE Addr2 A23 ~ A16 BYTE Addr1 A15 ~ A8 BYTE Addr0 A7~ A10
Do_SPI_Read_256Bytes	SPI read 256 bytes. [Return] : None [Parameter] : XBYTE *DataPointer BYTE Addr2 A23 ~ A16 BYTE Addr1 A15 ~ A8 BYTE Addr0 A7~ A10
Do_SPI_Write_Enable	Sending 06h (write enable) instruction to SPI. [Return] : None [Parameter] : None
Do_SPI_Write_Disable	Sending 04h (write disable) instruction to SPI. [Return] : None [Parameter] : None

Table 4-9-1

5. Debugging Interface

The user can update EC bin file immediate by KBS download board and ITE “WinFlash” utility and monitor all registers and memory of EC by ITE “D2EC” utility. Firmware also provides serial port debug interface please refer to section 5.2.

5.1 KBS Download Board

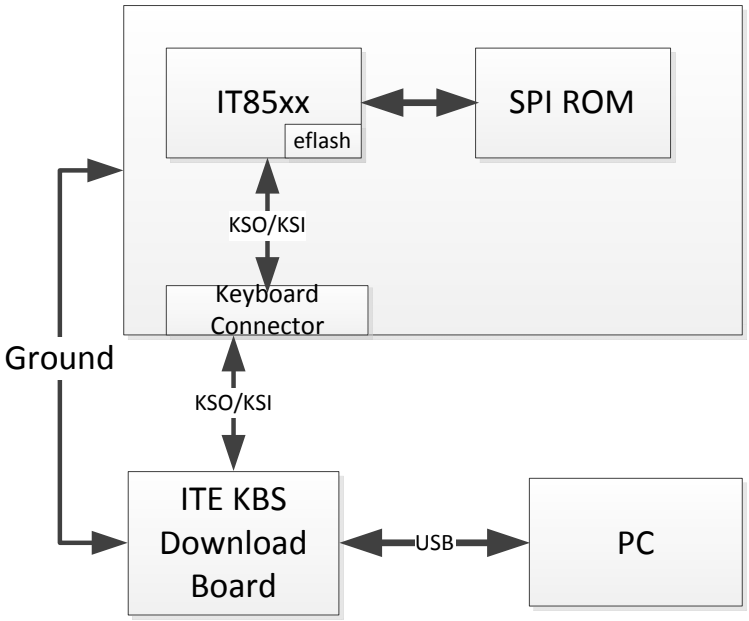


Figure 5-1-1 Schematic diagram of develop tool

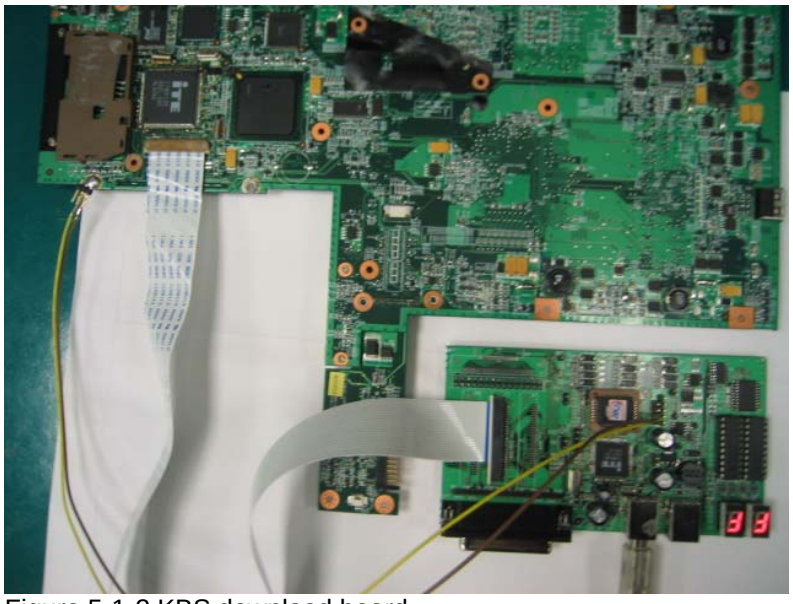


Figure 5-1-2 KBS download board

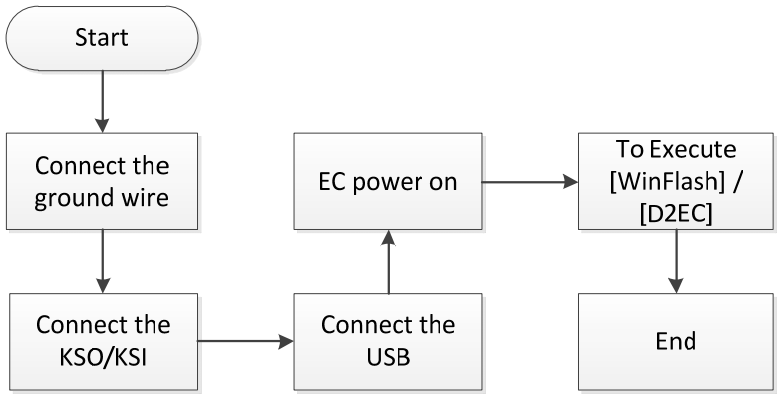


Figure 5-1-3 flowchart of connecting KBS

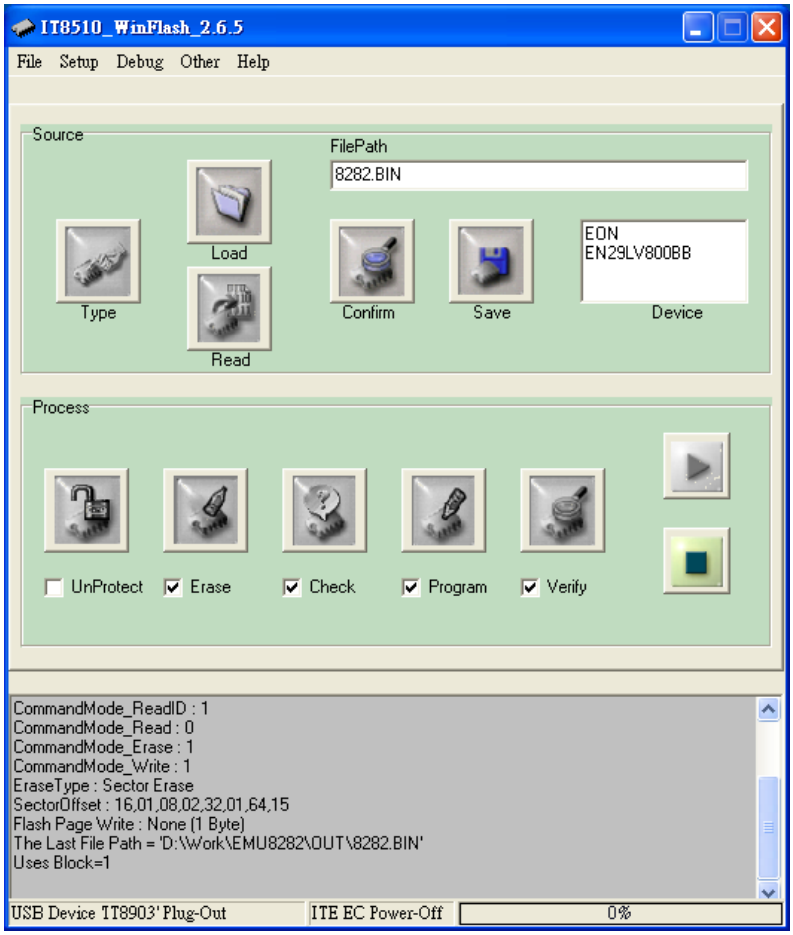


Figure 5-1-4 WinFlash utility

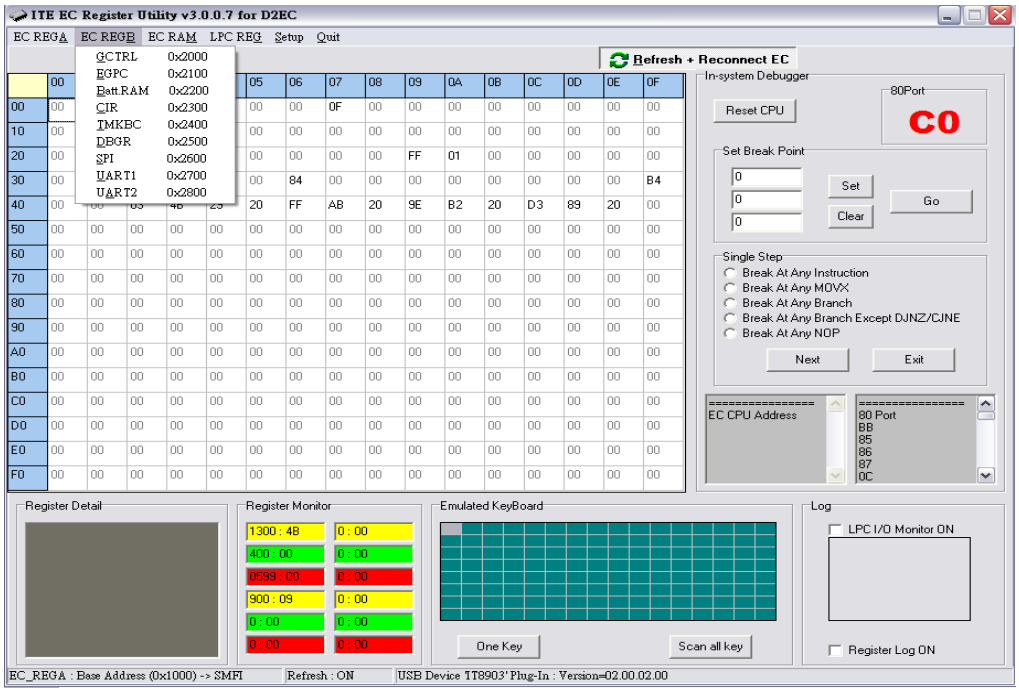


Figure 5-1-5 D2EC utiliy

5.2 Serial Port of 8032

The 8032 serial port debug interface is easy to use. Just define “UART_Debug” selection and choosing communication baud rate in “OEM_Project.H”. You can connect 8032 serial port with your PC via a serial port and using I/O functions for debugging. For example “printf”、“putchar”… etc.

Note : 1、Never use internal timer 2 because the firmware using timer 2 as the baud rate generator for serial port.
2、The GPIOB.1 pin is used as Tx function of serial port.