CANTUS-CAN

- *Software Develop Environment* -

32bits EISC Microprocessor *CANTUS*

Ver 1.2
April 24, 2013

Advanced Digital Chips Inc.

History

2013-02-19	Released	
2013-03-19	Modified	4.1 Configuration 5.3 Run Application
2013-04-24	Modified	CANTUS-CAN

CANTUS-CAN Application Note : #0000B Software Develop Environment

Office

(Gwanyang-dong, Keumkang Penterium IT Tower) 22F, A-Tower,
282, Hagui-ro, Dongan-gu, Anyang-si, Gyeonggi-do, SEOUL 431-810 Rep. of KOREA

Tel : +82-31-463-7500
Fax : +82-31-463-7588
URL : http://www.adc.co.kr

— Table of Contents —

1 Summary

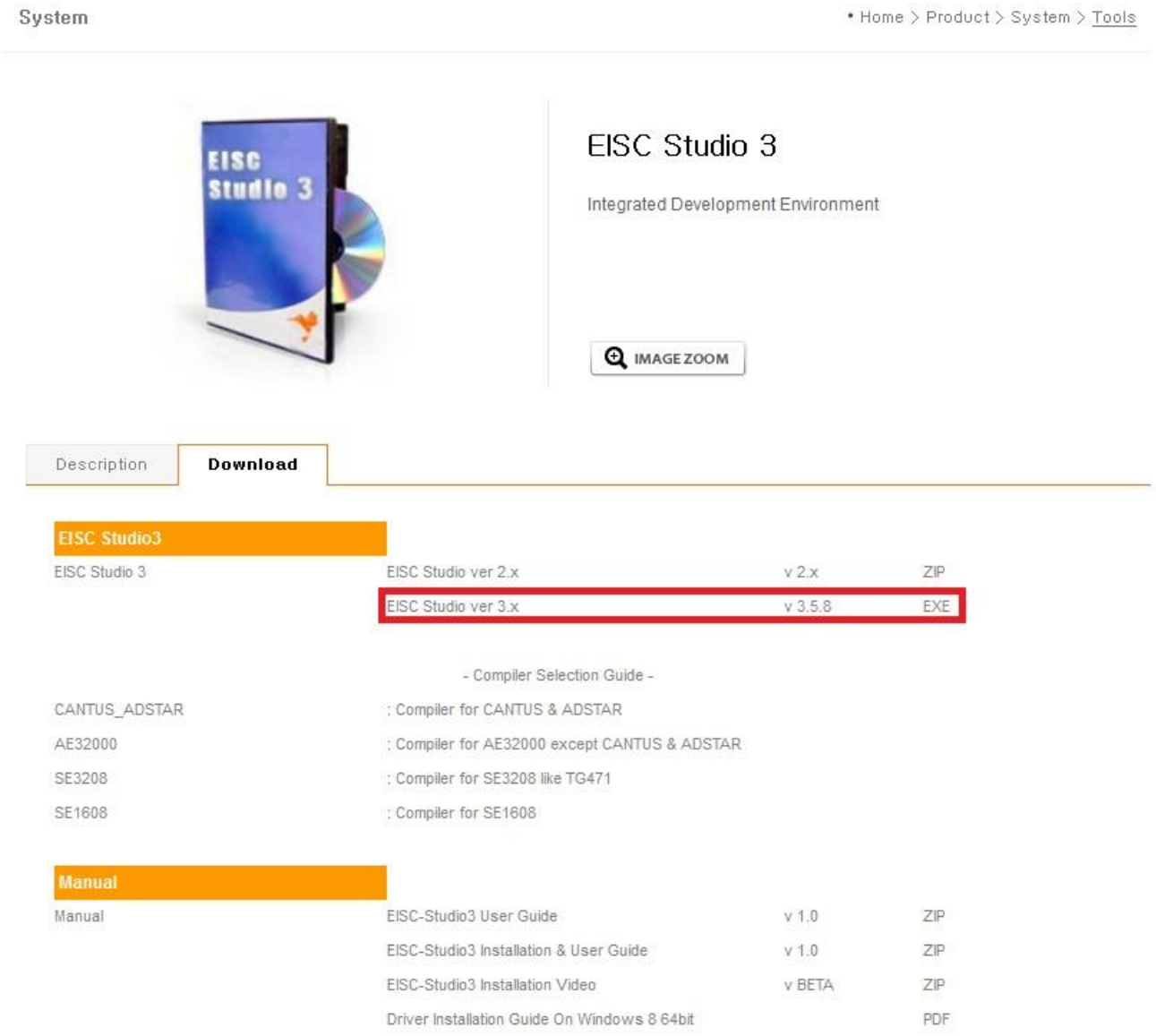
이 문서는 CANTUS-CAN를 사용하기 위한 Application Note로 Software를 개발하기 위한 개발환경 구축과 SDK의 구조를 설명한다.

2 IDE

EISC Studio3는 EISC Core를 위한 Windows[1]상에서 동작하는 통합개발환경으로 Editor, Compiler, Downloader, Debugger를 포함한다.

2.1 Download

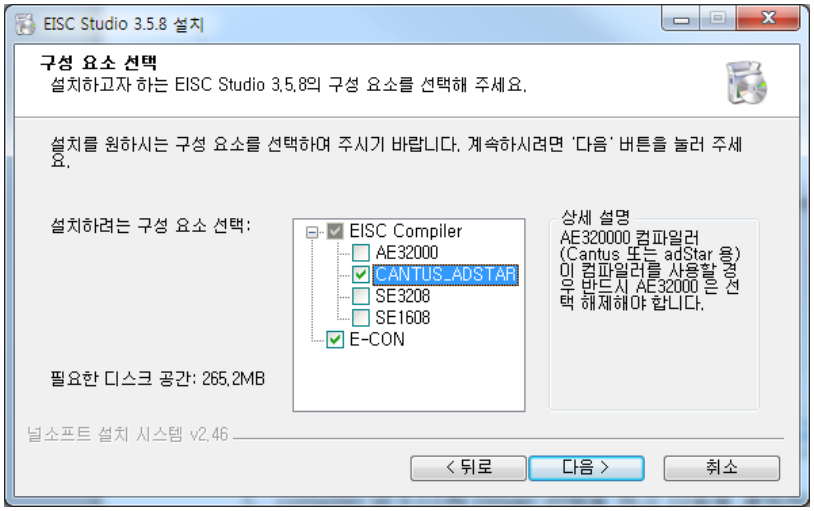
당사 홈페이지[2]에서 "Product → System → EISC Studio3 → Download"에서 Download 할 수 있다.



[1] XP 이상에서 사용할 수 있다.
[2] http://www.adc.co.kr

2.2 Install

Download한 실행파일을 실행하여 설치를 시작한다. 다음과 같이 구성 요소 선택에서 Compiler로 “CANTUS_ADSTAR”를 선택한다. “E-CON”을 선택하지 않으면 Download Tool E-CON을 위한 Windows Driver가 설치 되지 않는다.

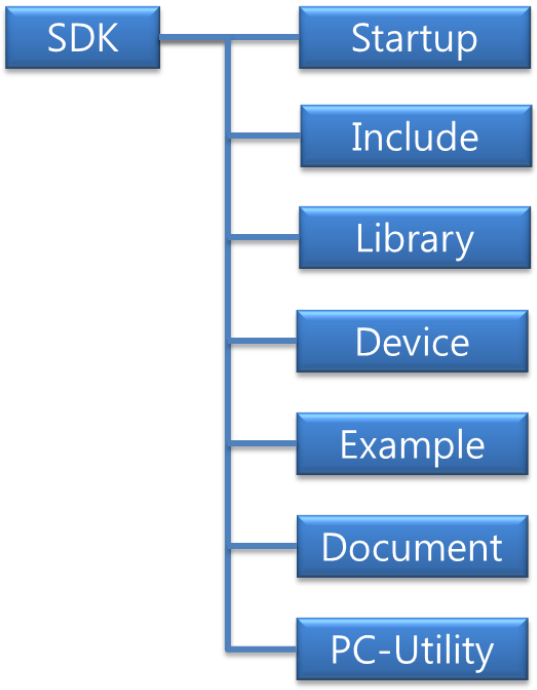


3 SDK

SDK(Software Development Kit)는 CANTUS-CAN에서 CANTUS의 모든 기능을 사용할 수 있도록 구성되어있다.

3.1 Structure

SDK는 다음과 같은 구조를 갖는다.



Startup :

실행 프로그램이 반드시 포함해야 하는 File의 폴더이다.

start.S
Reset이후 가장 먼저 실행되는 Code(C Runtime 0) 이다.

boardinit.c
EVM를 위한 Source File 이다.

cantus.ld
Linking을 위한 linker script 이다.

Include :

Header File의 폴더이다.

Library :

Source File의 폴더로 Source File을 Compile하기 위한 Library Project를 포함하고 있다.

Device :

EVM의 Device를 위한 Header와 Source File의 폴더이다.

Example :

EVM에서 사용할 수 있는 예제 프로그램의 Project 폴더이다.

Document :

Device의 Datasheet및 Manual의 폴더이다.

PC-Utility :

CANTUS_Dev_Tool
BootLoader의 USB-Communication Mode를 사용할 경우에 사용하는 USB Download Tool 이다.

E-CON
E-CON을 위한 Tool 이다.

EVM_Nand_Flash_Data
Example에서 사용하는 Nand Flash의 Data로 bmp, jpg file의 폴더이다.

USB_Driver
Windows용 Bulk, CDC Driver 이다.

3.2 Datatype

SDK에서는 편의성과 가독성을 위해 다음과 같이 자료형을 정의한다.

signed char	S8, s8
signed short	S16, s16
signed int	S32, s32
unsigned char	U8, u8, BYTE
unsigned short	U16, u16, WORD
unsigned int	U32, u32, DWORD
unsigned long long	U64, u64
1	TRUE, true
0	FALSE, false

SDK에서는 다음과 같은 접두사를 사용하여 Register를 정의한다.

R_ : Register Address

F_ : Register Flag

EVM_ : EVM에서만 사용하는 재정의

따라서 다음과 같이 사용할 수 있다.

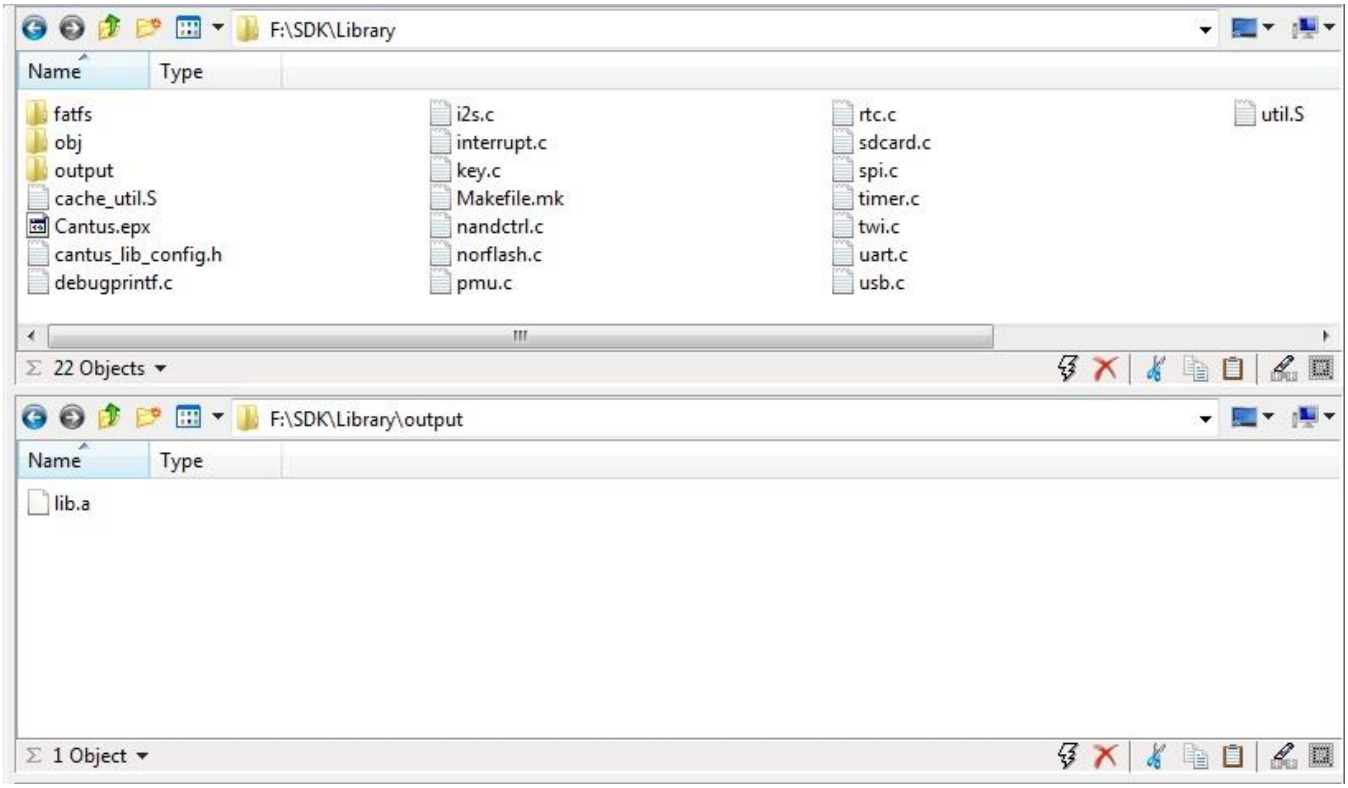
```
*R_PAF2  = F_PAF2_0_GPIO | F_PAF2_1_GPIO | F_PAF2_2_SRAM_NRE | F_PAF2_3_SRAM_NWE
         | F_PAF2_4_GPIO | F_PAF2_5_SRAM_NCS1 | F_PAF2_6_SRAM_NCS2 | F_PAF2_7_SRAM_NCS3;

*R_P2IDIR = EVM_F_WIFI_GS_NINT;
```

3.3 Library

Library Project는 main()함수 없이 “timer.c”나 “uart.c”와 같은 Source File을 Build하여 “.a”와 같은 Archive를 Output File로 생성한다. Example Project는 “timer.c”나 “uart.c”와 같은 Library폴더의 Source File을 Build하지 않고, 생성된 Archive를 사용한다.

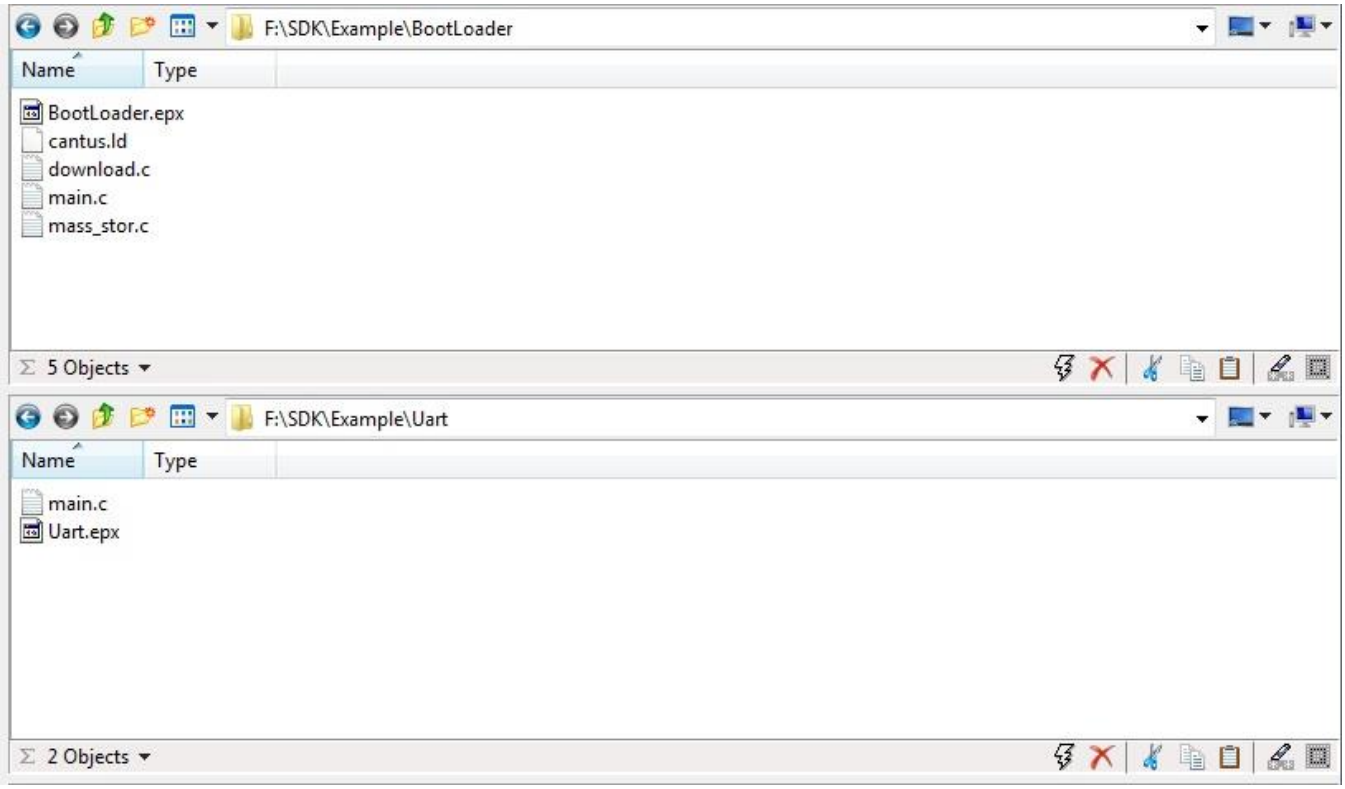
따라서 Example Project Build에 앞서 Library를 Build 해야 한다. Library Project는 “SDK\Library\Cantus.epx” 이며, 이를 실행하여 EISC Studio3에서 Build 한다.

Build의 Output File인 Archive(SDK\Library\output\lib.a)가 생성 된다.

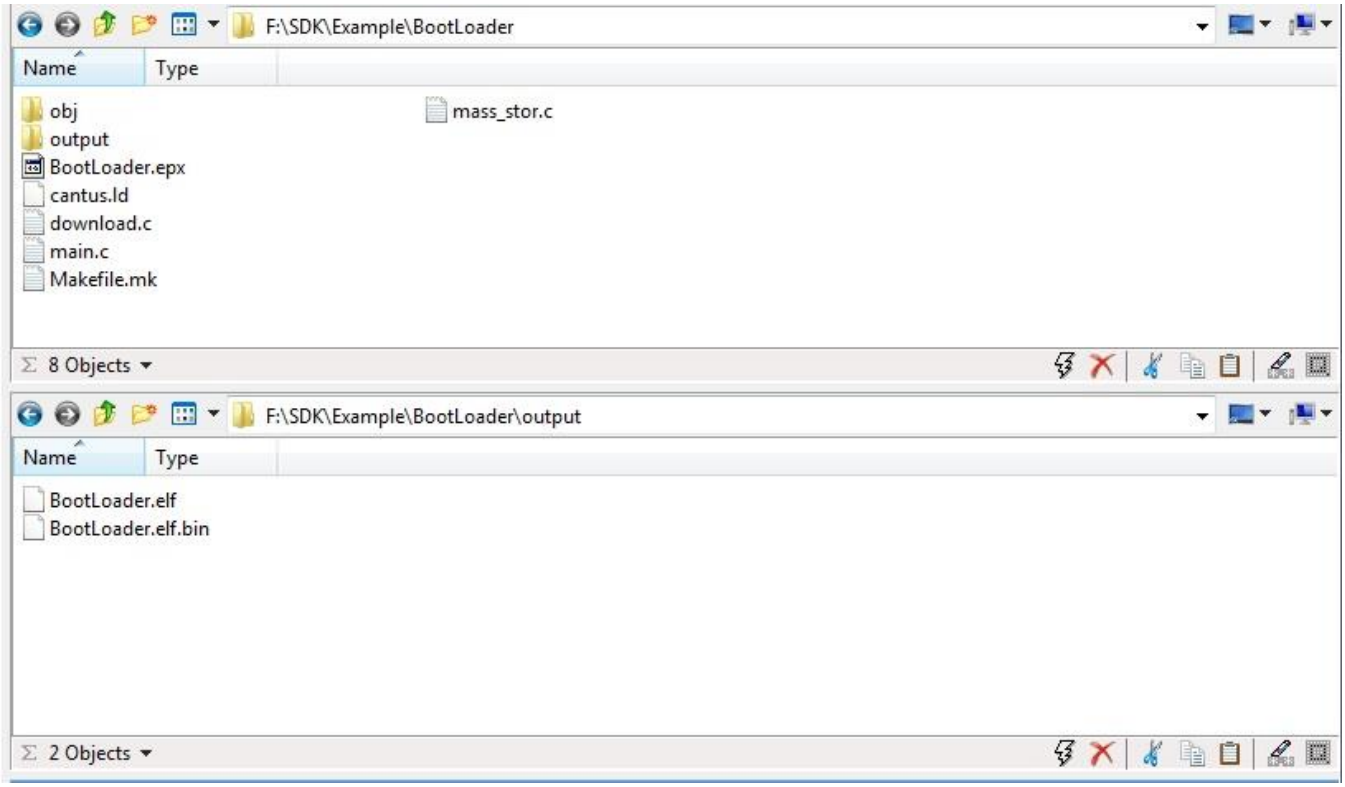


3.4 Example

Example Project는 기본적으로 “main.c”와 “SDK\Startup”의 Source File과 “SDK\Library\Output\lib.a”만으로 Build 된다. Example에 따라 “SDK\Device”의 Source File이나 이외의 Source File을 포함할 수 있다.



Example Project가 Build되면 Output File로 “.elf”와 “.elf.bin”이 Project File의 하위 “output” 폴더에 생성된다.



4 Library

Library는 main()함수 없이 “timer.c”나 “uart.c”와 같은 Source File을 Build하여 “.a”와 같은 Archive를 Output File로 생성한다.

4.1 Configuration

Library는 각 Source File을 Build할 때 “SDK\Library\lib_config.h”를 참조 한다. “lib_config.h”의 정의를 사용하여 다음과 같이 해당 Source File의 구성을 변경할 수 있다.

```
#define OSC_CLOCK 11289600
```
pmu.c

EVM의 Xtal Clock 값(Hz단위)을 설정한다.

```
#define CONFIG_TIMER_PRESCALE 2
```
timer.c

기본적으로 사용할 Timer Control Register(TMxCTRL)의 Pre-scale Factor 값을 설정한다. 이 값은 `settimer()`, `delayms()`에서 TMxCTRL을 설정할 때 사용 된다.

```
#define UART0_ENABLE
#define UART1_ENABLE
#define UART2_ENABLE
#define UART3_ENABLE
#define UART4_ENABLE
#define UART5_ENABLE
#define UART6_ENABLE
#define UART7_ENABLE
```
uart.c

Example에서 UART를 사용하기 위해서는 사용할 각 채널이 정의되어 있어야 한다. 기본적으로 채널 0, 4, 5, 6, 7을 사용하도록 설정한다.

```
#define UART_BUF_SIZE 128
```
uart.c

UART 각 채널에서 사용하도록 설정한 Ringbuffer의 Byte 크기를 설정한다.

```
#define CONFIG_UART_RX_INTERRUPT
```
uart.c

UART 각 채널에서 Rx Interrupt를 사용하도록 설정한다.

```
#define CONFIG_UART_TX_INTERRUPT
```
uart.c

UART 각 채널에서 Tx Interrupt를 사용하도록 설정한다.

```
#define DEBUG_CHANNEL EVM_UART_0
```
uart.c

`debugstring()` 과 `debugprintf()`와 같은 함수를 사용하였을 때 출력할 UART 채널을 설정한다.

```
#define CONFIG_DEBUGPRINTF_ENABLE
```
uart.c

`debugprintf()`를 사용할 수 있도록 설정한다.

`#define CONFIG_KEY_SUSTAIN_SUPPORT`
key.c
Key의 Sustain 상태를 확인하도록 설정한다.

`#define CONFIG_KEY_TIMER_CH_SUSTAIN 6`
key.c
Key의 Sustain 상태를 확인할 때 사용할 TIMER 채널을 설정한다.

`#define CONFIG_KEY_SUSTAIN_CHECK_TIME 300`
key.c
Key를 Sustain 상태로 판단할 시간(ms)을 설정한다.

`#define CONFIG_KEY_TIMER_CH_SCAN 7`
key.c
Key를 Scan할 Timer 채널을 설정한다.

`#define CONFIG_KEY_BUF_SIZE 4`
key.c
Key를 Scan한 결과를 저장할 Buffer의 크기를 설정한다.

`#define CONFIG_I2S_BUF_LIST_MAX 3`
i2s.c
WAVE DATA LIST의 최대 개수를 설정한다.

`#define CONFIG_I2S_USE_DMA`
i2s.c
I2S의 Transmit Access Mode를 DMA로 설정한다.

`#define NAND_FLASH_K9F8G08U`
nandctrl.c
8192 Block의 NAND Flash K9F8G08U를 사용하도록 설정한다.

`#define CONFIG_FAT_NAND_SUPPORT`
fatfs\diskio.c
FAT File System에서 NAND Flash를 사용하도록 설정한다.

`#define CONFIG_FAT_SDCARD_SUPPORT`
fatfs\diskio.c
FAT File System에서 SDCARD를 사용하도록 설정한다.

4.2 Build

Library Project를 Build 하면 다음과 같이 Output File이 생성된다.

- SDK\Library\output\lib.a

5 BootLoader

BootLoader는 CANTUS Flash Sector 0에서 동작하는 Program으로 다음과 같은 기능을 갖는다.

- Run Application
- USB Mass Storage
- USB Communication

USB Mass Storage나 USB Communication의 기능이 필요하지 않을 경우에는 BootLoader없이 Application을 CANTUS Flash Sector 0에서 동작하도록 할 수 있다.

자세한 내용은 AN_0001B_BOOTLOADER를 참조하라.

5.1 Build

Library Project를 Build 한 후에 BootLoader Project를 Build할 수 있다. 다음과 같이 Output File이 생성된다.

- SDK\Example\BootLoader\output\ BootLoader.elf
- SDK\Example\BootLoader\output\ BootLoader.elf.bin

5.2 Download

EVM의 SW2를 JTAG Mode로 두고, 전원을 키고 “SDK\Example\BootLoader\output\BootLoader.elf.bin “을 E-CON을 이용하여 Download 한다.

5.3 Run Application

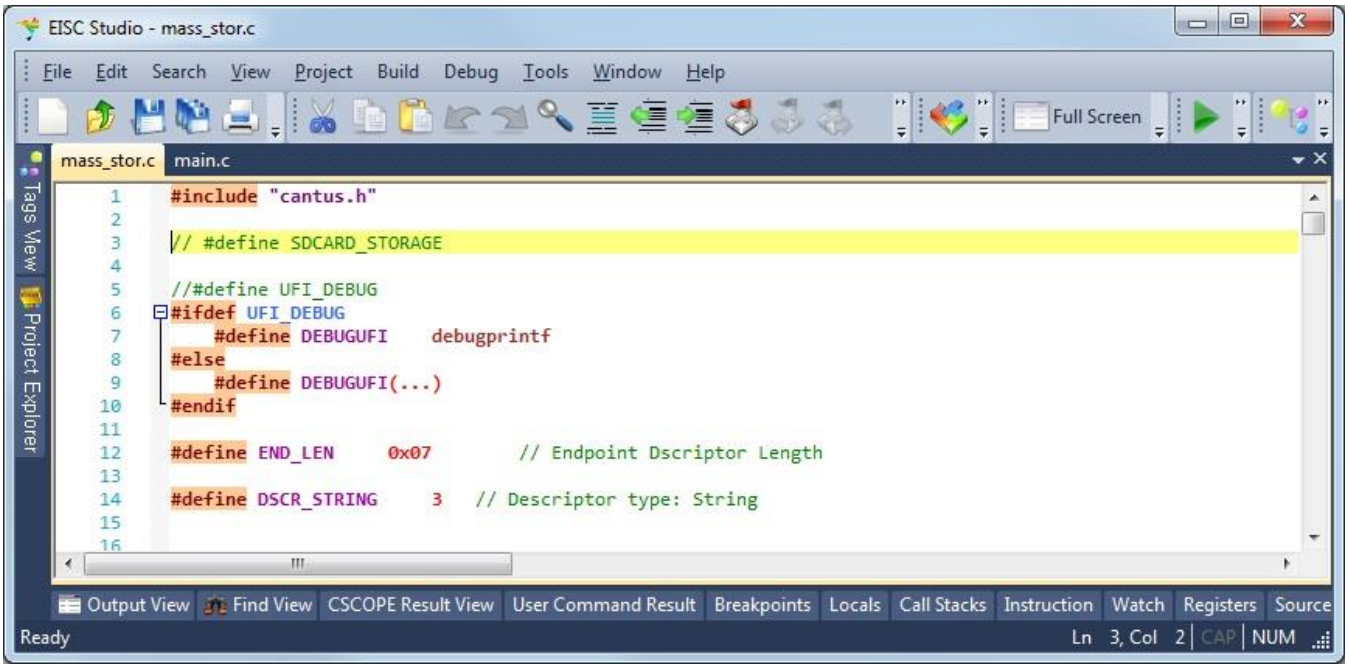
BootLoader가 실행되면 1초 이내에 Key입력이 없을 경우 특정 Sector에 Download한 Program을 실행한다. CANTUS Type에 따라 실행하는 Sector가 다르다.

CANTUS 128A :
Sector Address 0x8000
Sector Number 8

CANTUS 512 :
Sector Address 0x10000
Sector Number 1

5.4 USB Mass Storage

BootLoader 실행 후 1초 이내에 SW6이 입력되면 USB Mass Storage Mode가 실행된다. “mass_stor.c”의 “`#define SDCARD_STORAGE`”에 따라 EVM의 NAND 또는 SD Card를 Disk로 사용한다.



5.5 USB Communication

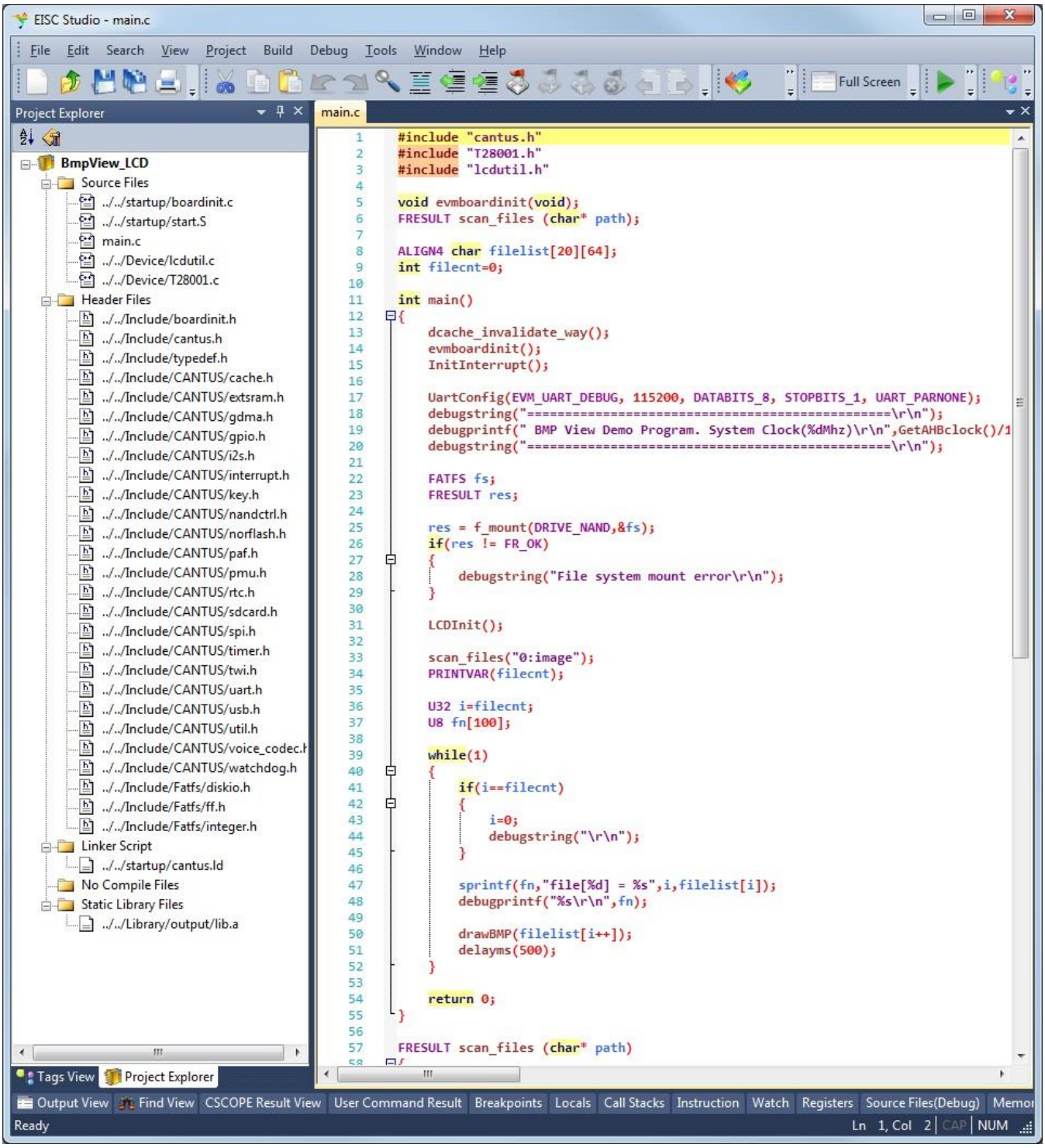
BootLoader 실행 후 1초 이내에 SW5가 입력되면 USB Communication Mode가 실행된다. PC와 처음 연결할 경우에는 “SDK\PC-Utility\USB_Driver\BULK”의 Driver를 설치한다.

USB Communication Mode에서는 “SDK\PC-Utility\CANTUS_Dev_Tool\CANTUS_Dev_Tool.exe”를 실행하여 CANTUS Flash Sector 1 이후 Sector에 File Download가 가능하다.

6 Application

Appliaction[3]은 기본적으로 “main.c”와 “SDK\Startup”의 Source File과 “SDK\Library\Output\lib.a”만으로 Build 된다. Example에 따라 “SDK\Device”의 Source File이나 이외의 Source File을 포함할 수 있다.

다음과 같이 Project Explorer에서 BmpView_LCD가 참조하는 File을 확인 할 수 있다.



[3] Example의 Bootloader이외

6.1 Build

Library Project를 Build 한 후에 Application Project를 Build할 수 있다. 다음과 같이 ".elf"와 ".elf.bin"이 Project File의 하위 "output" 폴더에 생성된다.

- SDK\Example\BmpView_LCD\output\BmpView_LCD.elf
- SDK\Example\BmpView_LCD\output\BmpView_LCD.elf.bin

6.2 Download

EVM의 SW2를 JTAG Mode로 두고, 전원을 키고 "SDK\Example\BmpView_LCD\output\BmpView_LCD.elf.bin"을 E-CON을 이용하여 Download 한다.

또는 BootLoader의 USB Communication Mode를 사용하여
"SDK\PC-Utility\CANTUS_Dev_Tool\CANTUS_Dev_Tool.exe"로 Download 한다.

6.3 Run Application

BootLoader가 실행되면 1초 이내에 Key입력이 없을 경우 Download한 Program을 실행한다.

7 BASICAL API

7.1 UART

7.1.1 UartConfig

```
BOOL UartConfig(int ch, int baud, UART_DATABITS databits, UART_STOPBITS stopbit,
                UART_PARITY parity)
```

UART 사용을 위한 초기 설정을 하는 함수이다.

lib_config.h 파일에 `#define CONFIG_UART_RX_INTERRUPT`가 정의 되어 있으면, UART Rx Interrupt 설정도 같이 한다. (기본적으로 `CONFIG_UART_RX_INTERRUPT`가 정의 되어 있다.)

Parameter

`ch` UART channel값. 초기 설정할 channel을 입력한다.

`baud` UART baud rate 값. Baud rate값을 설정한다.

SDK는 기본적으로 115200을 사용한다.

`databits` UART 전송 data bit을 설정한다. SDK에서는 data bit를 다음과 같이 정의하고 있다.

```
typedefenum{
    DATABITS_5 = 5,
    DATABITS_6 = 6,
    DATABITS_7 = 7,
    DATABITS_8 = 8
}UART_DATABITS;
```

`stopbit` UART의 stop bit를 설정한다. SDK에서는 stop bit를 다음과 같이 정의하고 있다.

```
Typedefenum{
    STOPBITS_1 = 1,
    STOPBITS_2 = 2
}UART_STOPBITS;
```

`parity` UART의 parity bit를 설정한다. SDK에서는 parity bit를 다음과 같이 정의하고 있다.

```
Typedefenum{
    UART_PARNONE = 0,
    UART_PARODD,
    UART_PAREVEN
}UART_PARITY;
```

Return Value

`TRUE(1) or FALSE(0)`

7.1.2 UartPutch

```
BOOL UartPutCh(int n, char ch)
```

UART n 채널을 통해 ch 값을 출력한다.

Parameter

n	값을 출력할 UART channel값.
ch	UART를 통해 출력할 값. SDK는 기본적으로 115200을 사용한다.

Return Value

TRUE(1) or FALSE(0)

7.1.3 UartPutString

```
BOOL UartPutString(int n, U8* str)
```

UART n 채널을 통해 str의 문자열을 출력한다.

Parameter

n	값을 출력할 UART channel값.
str	출력할 문자열의 pointer.

Return Value

TRUE(1) or FALSE(0)

7.1.4 UartGetCh

```
BOOL UartGetCh(int n, U8* ch)
```

UART n 채널을 통해 1 byte(1 character) 값을 받아 ch에 저장한다.

Parameter

n	값을 입력 받을 UART channel값.
ch	UART를 통해 값을 입력 받을 변수의 pointer.

Return Value

TRUE(1) or FALSE(0)

7.1.5 UartGetData

```
int UartGetData(int n, U8* buf, int bufmax)
```

UART n 채널을 통해 최대 bufmax 만큼 값(character)을 읽어서 buf에 저장한다.

Parameter

n	값을 입력 받을 UART channel값.
buf	UART를 통해 값을 입력 받을 변수의 pointer.
bufmax	저장할 최대 수

Return Value

buf에 저장한 수.

7.1.6 Uart_rx_flush

```
void Uart_rx_flush(int ch)
```

UART n 채널의 Rx FIFO를 초기화 한다.

Parameter

ch	초기화할 UART channel값.

Return Value

None

7.1.7 Uart_tx_flush

```
void Uart_tx_flush(int ch)
```

UART n 채널의 Tx FIFO를 초기화 한다.

Parameter

ch	초기화할 UART channel값.

Return Value

None

7.1.8 setdebugchannel

```
void setdebugchannel(int ch)
```

debugprintf(), debugsting() 함수가 출력할 UART 채널을 설정한다.

Parameter

ch 설정할 UART channel값.

Return Value

None

7.1.9 getdebugchannel

```
int setdebugchannel(void)
```

debugprintf(), debugsting() 함수가 출력할 UART 채널을 읽는다.

Parameter

None

Return Value

설정된 UART channel값.

7.1.10 debugprintf

```
void debugprintf(const char* const format, …)
```

printf()와 같은 함수로 설정된 UART Channel로 출력한다.

Usage

```
debugprintf("result : 0x%x, %d \r\n", result, result);
```

Return Value

None

7.1.11 debugstring

```
void debugstring(const char* str)
```

문자열을 설정된 UART Channel로 출력한다.

Usage

```
debugstring("CANTUS \r\n");
```

Return Value

None

7.1.12 PRINTLINE

```
#define PRINTLINE       debugprintf("%s, %d \r\n", __FILE__, __LINE__)
```

현재 Code의 Line Number를 설정된 UART Channel로 출력한다.

Usage

```
PRINTLINE;
```

7.1.13 PRINTVAR

```
#define PRINTVAR(A)     debugprintf("%s, %d : %s = 0x%x(%d) \r\n", __FILE__, __LINE__,
                                    #A, A, A)
```

현재 Code의 Line Number를 설정된 UART Channel로 출력한다.

Usage

```
int a = 10;
PRINTVAR(a);
```

7.2 INTERRUPT

7.2.1 InitInterrupt

```
void InitInterrupt(void)
```

Interrupt 초기화.

Parameter

None

Return Value

None

7.2.2 INTERRUPT_TYPE

CANTUS는 다음과 같이 32개의 User Vector가 존재 한다.

```
typedef enum
{
    INTNUM_EIRQ0              =0,
    INTNUM_TIMER0               ,
    INTNUM_I2S                  ,
    INTNUM_UART0                ,
    INTNUM_EIRQ1              =4,
    INTNUM_TIMER1               ,
    INTNUM_DMA0                 ,
    INTNUM_UART1                ,
    INTNUM_GPIO0              =8,
    INTNUM_TIMER2               ,
    INTNUM_USB                  ,
    INTNUM_UART2                ,
    INTNUM_GPIO1             =12,
    INTNUM_TIMER3               ,
    INTNUM_SPI                  ,
    INTNUM_UART3                ,
    INTNUM_GPIO2             =16,
    INTNUM_TIMER4               ,
    INTNUM_TWI_RTC              ,
    INTNUM_UART4_OCR2A          ,
    INTNUM_GPIO3             =20,
    INTNUM_TIMER5               ,
    INTNUM_VOICE                ,
    INTNUM_UART5_OCR2B          ,
    INTNUM_GPIO4             =24,
    INTNUM_TIMER6               ,
    INTNUM_NFCTRL_SDHC          ,
    INTNUM_UART6_OCR3A          ,
    INTNUM_GPIO5             =28,
    INTNUM_TIMER7_KEYSCAN       ,
    INTNUM_DMA1                 ,
    INTNUM_UART7_OCR3B          ,

    INNUM_MAX
}INTERRUPT_TYPE;
```

7.2.3 setInterrupt

```
BOOL setInterrupt(INTERRUPT_TYPE intnum, void (*fp)())
```

intnum의 Interrupt가 발생하였을 때 호출할 함수를 설정한다.

Parameter

intnum	INTERRUPT_TYPE
(*fp)()	함수 Pointer.

Return Value

TRUE(1) or FALSE(0)

7.2.4 EnableInterrupt

```
BOOL EnableInterrupt(INTERRUPT_TYPE intnum, BOOL b)
```

intnum의 Interrupt Enable.

Parameter

intnum	INTERRUPT_TYPE
b	TRUE(1) : Enable FALSE(0) : Disable

Return Value

TRUE(1) or FALSE(0)

7.3 TIMER

7.3.1 settimer

```
BOOL settimer(int nCh, U32 ms)
```

Timer 채널 nCh를 주기 ms로 설정, Enable 하여 Timer Interrupt를 발생한다.
내부에서 EnableInterrupt(intnum, TRUE)를 호출 하므로, settimer() 호출 전에 setInterrupt()로 intnum이 호출할 함수를 설정해야 한다. Prescale 값에 따라 FALSE를 반환할 수 있다.

Parameter

nCh	Timer 채널
ms	Timer Interrupt 주기(ms 단위)

Return Value

TRUE(1) or FALSE(0)

7.3.2 stoptimer

```
BOOL stoptimer(int nCh)
```

Timer 채널 nCh를 Disable한다.

Parameter

nCh	Timer 채널

Return Value

TRUE(1) or FALSE(0)

7.3.3 delayms

```
void delayms(U32 ms)
```

ms 만큼 지연한다.

Parameter

ms	지연시간(ms 단위)

Return Value

TRUE(1) or FALSE(0)