



adLuna

- SDK Reference Manual -

32bits EISC Microprocessor*adStar-L*

Ver 1.0
February 27, 2019

Advanced Digital Chips Inc.

History

2019-02-27 Created Preliminary Specification

adStar-L SDK Reference Manual

©Advanced Digital Chips Inc.

All right reserved.

No part of this document may be reproduced in any form without written permission from Advanced Digital Chips Inc.

Advanced Digital Chips Inc. reserves the right to change in its products or product specification to improve function or design at any time, without notice.

Office

22F, Bldg A, KeumkangPentarium IT Tower,

810, Gwanyang_dong, Dongan-Gu, Anyang-si, Gyeonggi-do, 431-060, Korea

Tel : +82-31-463-7500

Fax : +82-31-463-7588

URL : <http://www.adc.co.kr>

— Table of Contents —

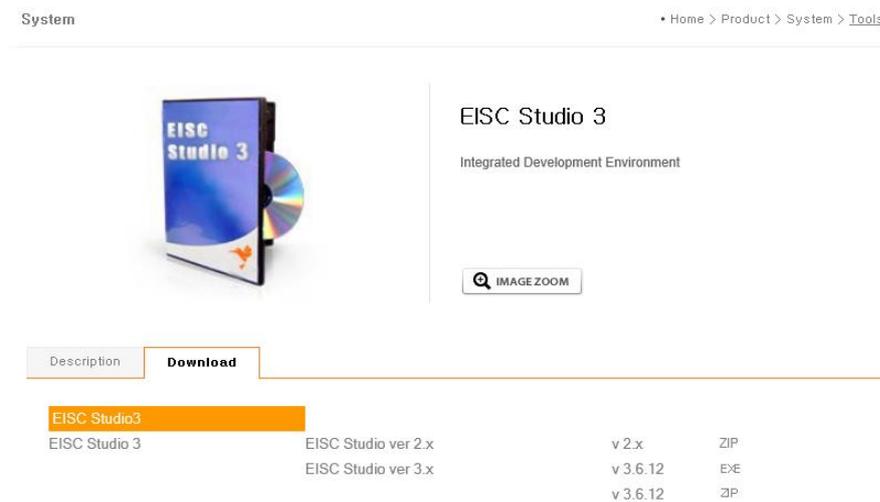
1. SOFTWARE 개발환경	6
2. ADLUNA SDK 활용	12
2.1 adLuna SDK 구성 및 기본 정의.....	12
2.2 adLuna SDK library build.....	14
2.3 Demo Program 실행.....	16
2.3.1 Project build & download.....	16
2.4 adLuna 새프로젝트만들기.....	17
3. LIB_CONFIG.H	23
4. UART	25
4.1 uart_config.....	25
4.2 uart_putch.....	26
4.3 uart_putdata.....	26
4.4 uart_putstring.....	26
4.5 uart_getch.....	27
4.6 uart_getdata.....	27
4.7 uart_rx_flush.....	27
4.8 uart_tx_flush.....	28
4.9 set_debug_channel.....	28
4.10 get_debug_channel.....	28
4.11 debugprintf.....	29
4.12 debugstring.....	29
4.13 PRINTLINE.....	29
4.14 PRINTVAR(A).....	30
4.15 UART Example.....	30
< UART Interrupt >.....	31
5. INTERRUPT	32
5.1 init_interrupt.....	32
5.2 set_interrupt.....	32
5.3 enable_interrupt.....	32
5.4 Interrupt Example.....	34
6. TIMER.....	35
6.1 set_timer.....	35
6.2 stop_timer.....	35
6.3 delays.....	35
6.4 set_pwm.....	36
6.5 TIMER Example.....	36

7. SOUND	37
7.1 <i>sount_init()</i>	37
7.2 <i>sound_loadwav</i>	37
7.3 <i>sound_loadwavp</i>	37
7.4 <i>sound_release</i>	38
7.5 <i>sound_play</i>	38
7.6 <i>sound_stop</i>	38
7.7 <i>sound_vol</i>	39
7.8 <i>sound_vol_wav</i>	39
7.9 <i>sound_pause</i>	39
7.10 <i>sound_resume</i>	39
7.11 <i>sound_isplay</i>	40
7.12 <i>sound_isplay</i>	40
7.13 <i>Sound Example</i>	41
8. TWI	42
8.1 <i>twi_set_freq</i>	42
8.2 <i>twi_write</i>	42
8.3 <i>twi_write_multi</i>	42
8.4 <i>twi_read</i>	43
8.5 <i>TWI Example</i>	43
9. SPI	46
9.1 <i>spi_master_init</i>	46
9.2 <i>spi_set_freq</i>	46
9.3 <i>spi_master_xfer</i>	46
9.4 <i>spi_wait_empty_fifo</i>	47
9.5 <i>SPI Example</i>	47
10. DEBUGGING	56
10.1 <i>Debugging 준비</i>	56
10.2 <i>Debugging</i>	57
11. ETC	59
11.1 <i>여유 메모리(Ram size) 확인 방법</i>	59
11.2 <i>현재 동작 시간 확인</i>	59
11.3 <i>파일 program 에 추가하기</i>	60
11.4 <i>GPIO 사용법</i>	60

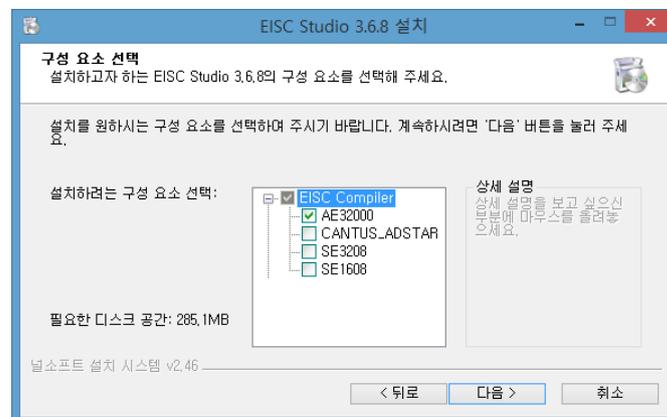
1. Software 개발환경

첫 번째 장으로 adLuna 를 사용하기 위한 개발환경을 구축하는 방법부터 알아보도록 하겠다. 에이디칩스에서는 adLuna Chip 을 위해 프로그램 코드 편집, compile, download, debugging 을 하나의 프로그램에서 수행할 수 있는 통합개발환경(IDE) EISC Studio3 를 제공하기 때문에 자료실에서 EISC Studio3 설치 파일을 내려 받아 설치만 해주면 간단하게 개발환경 구축을 마칠 수 있다. 참고로 EISC Studio3 는 Windows OS 만 지원하면 XP 이상에서 동작한다.

1. 에이디칩스 홈페이지(<http://www.adc.co.kr>)에 접속해서, Product → System → EISC Studio3 → Download 에서 EISC Studio ver 3.x 의 최신버전을 내려 받는다.
(설치 및 사용자 매뉴얼도 있으니 참고 하도록 한다.)



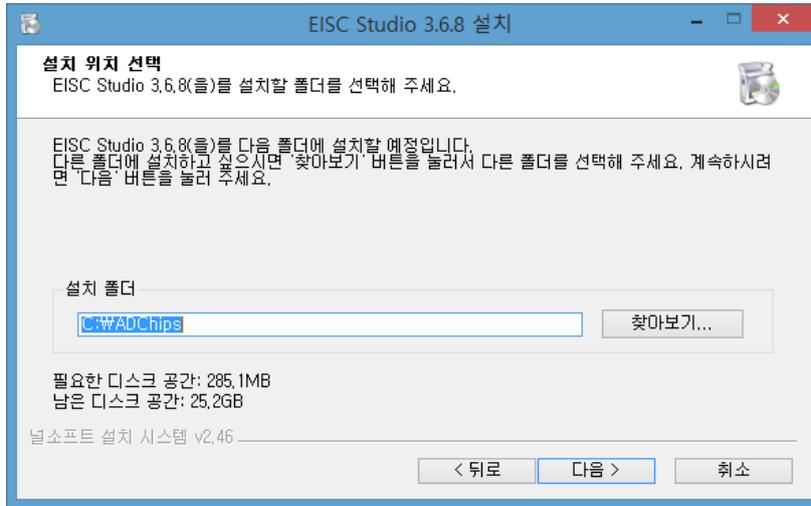
2. 내려 받은 ES3_setup_v3.6.xx.exe 를 실행하여 설치를 시작하면, 설치를 시작한다는 창이 뜨는데 '다음'을 클릭하여 설치를 시작한다. '다음'을 클릭하면 아래와 같이 설치할 Compiler 와 개발장비인 E-CON 의 Driver 설치를 선택하는 창이 뜬다. adStar-L 은 AE32000 processor¹를 내장한 Chip 으로 AE32000 을 선택하면 된다.



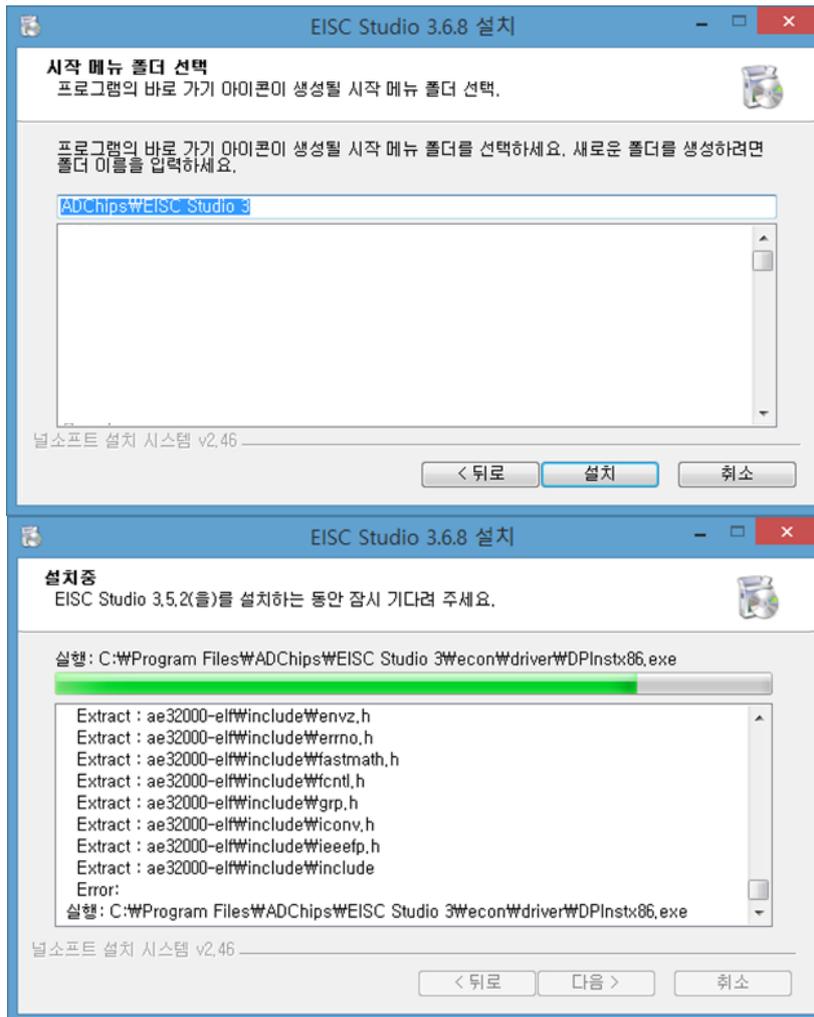
¹에이디칩스에서 개발한 32bit Processor. (EISC Architecture)

- 3. compiler 와 E-CON Driver 선택을 하고 다음을 클릭하면, 설치할 폴더를 선택하는 창이 나온다. 특별한 경우가 아니라면 default 로 되어있는 경로에 설치하면 된다.

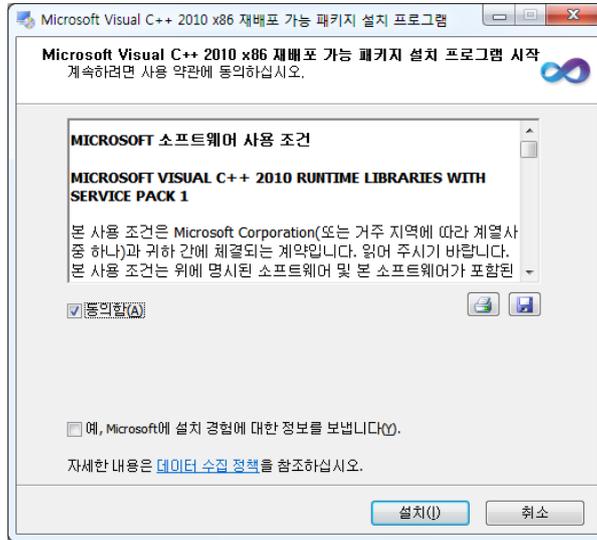
(주의사항으로는 경로상에 괄호 () 가 들어가면 안 된다.)



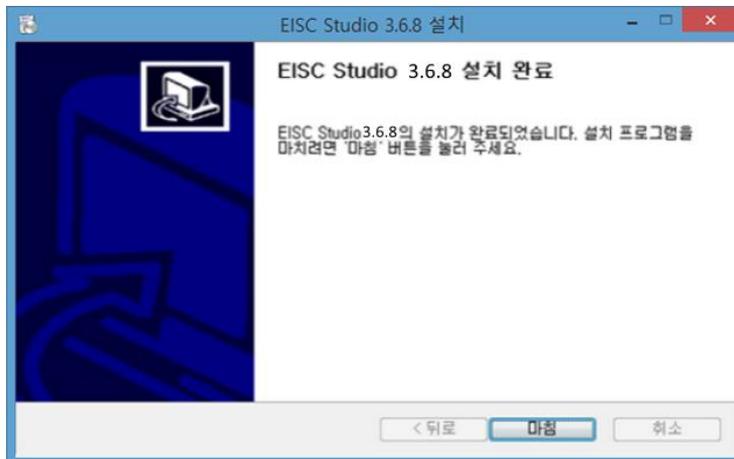
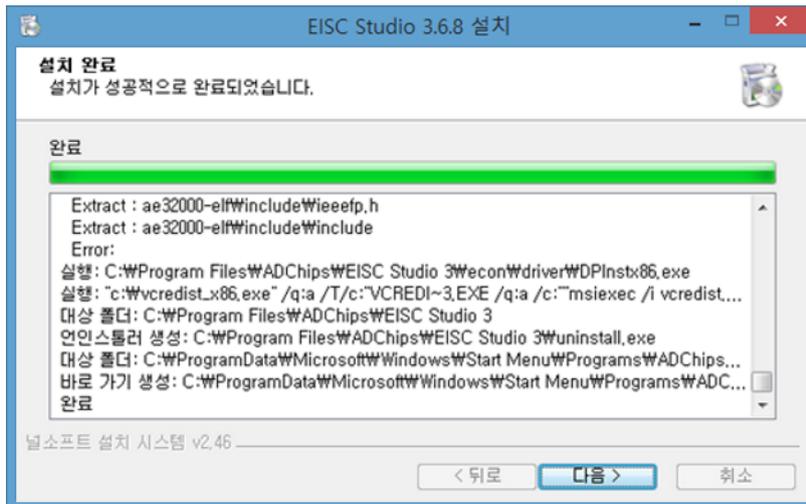
- 4. 다음을 클릭하면 아래와 같은 시작 메뉴 폴더를 선택하는 창이 뜨고, 설치를 누르면 설치가 시작된다.



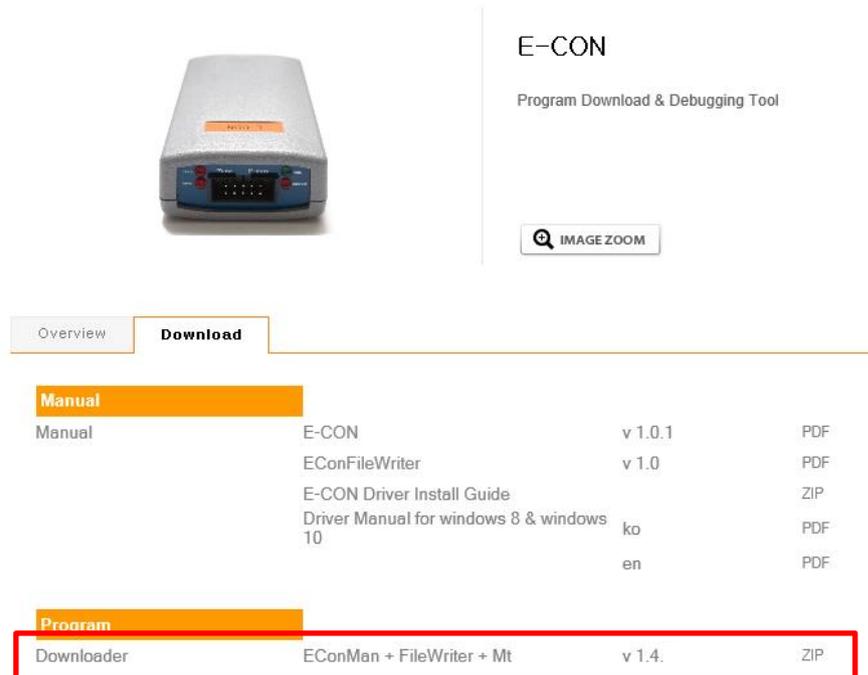
- 5. 다음과 같이 Microsoft Visual C++ 2010 재배포 가능 패키지를 설치하라는 창이 뜨면 동의함에 체크한 후 설치하면 된다. EISC Studio3 를 실행하는데 필요하므로 설치하고 넘어가도록 한다.



- 6. 설치가 완료되면 다음과 같은 창이 출력되고, 다음을 누르면 설치가 완료된다.



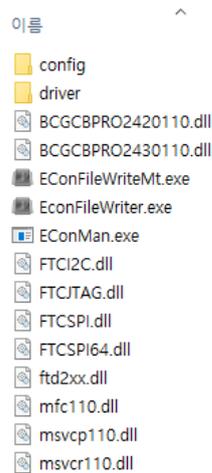
7. EISC Studio 3 설치가 완료 되면 다음으로 Download 장비인 E-Con driver 를 설치하여야 한다. E-Con 드라이버 및 프로그램은, 홈페이지(www.adc.co.kr)의 Product->System->E-Con ->Download 에서 EConMan+FileWriter+Mt 를 다운로드 받으면 된다. (v1.4.32 이상 사용)



The screenshot shows the E-CON website interface. On the left is an image of the E-CON hardware device. On the right, the text reads 'E-CON Program Download & Debugging Tool'. Below this is a navigation bar with 'Overview' and 'Download' tabs. Under the 'Download' tab, there are two sections: 'Manual' and 'Program'. The 'Program' section contains a table with the following data:

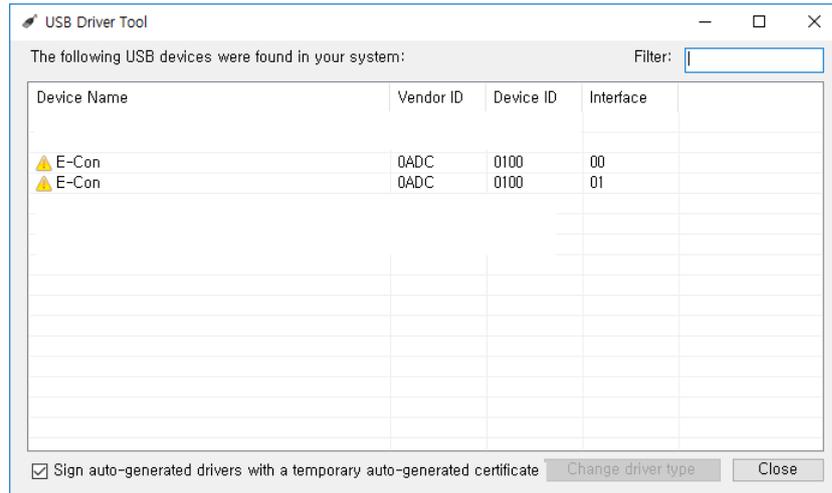
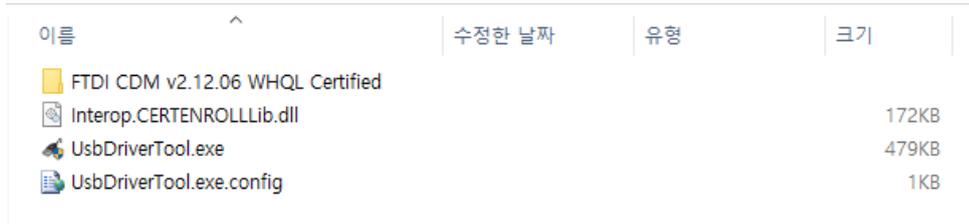
Manual			
Manual	E-CON	v 1.0.1	PDF
	EConFileWriter	v 1.0	PDF
	E-CON Driver Install Guide		ZIP
	Driver Manual for windows 8 & windows 10	ko	PDF
		en	PDF
Program			
Downloader	EConMan + FileWriter + Mt	v 1.4.	ZIP

8. Download 를 받은 후 압축을 풀고, EISC Studio 3 가 설치된 폴더의 econ 폴더 안에 복사를 하면 된다. (기존 파일이 존재한다면, 덮어쓰면 된다.)

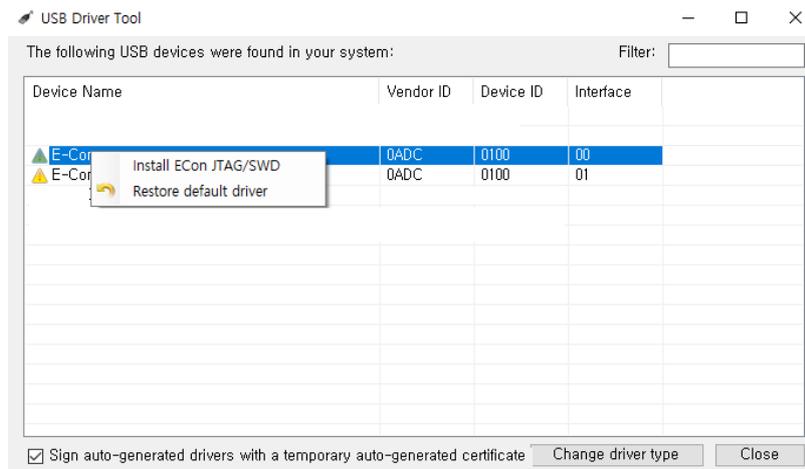


< EISC Studio 3 / econ 폴더 >

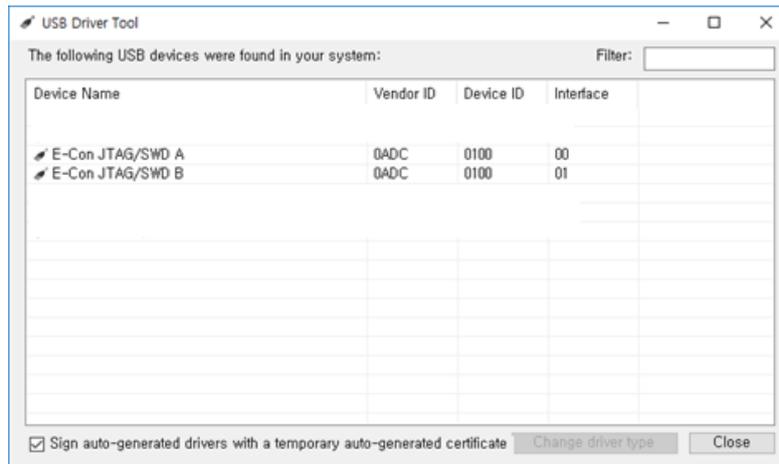
Driver 설치 파일은 driver 폴더 안에 있다. Driver 폴더 파일 중 UsbDriverTool.exe 를 실행한다. 그리고 E-Con 을 PC 와 연결하면 다음과 같이 E-Con 이라고 표시되는 것을 확인할 수 있다.



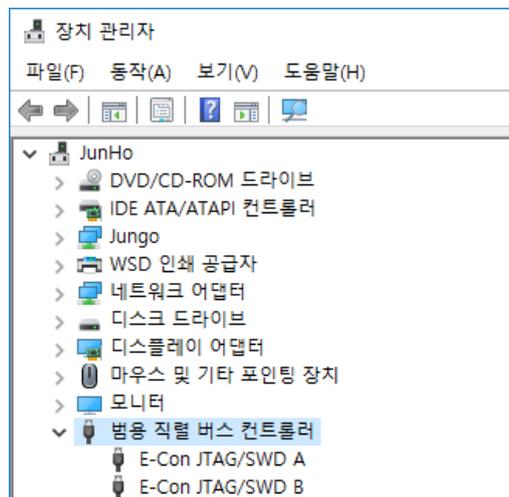
9. E-Con 항목에서 마우스 오른쪽 버튼을 눌러 다음의 그림처럼 나오면, Install ECon JTAG/SWD 를 클릭하여 driver 설치를 진행하면 된다. 남은 다른 하나의 E-Con 항목도 동일하게 진행을 하면 된다.



10. 위 그림처럼 driver 설치를 완료하면, 다음 그림과 같이, E-Con JTAG/SWD A, E-Con JTAG/SWD B 로 변경된 것을 확인할 수 있다. 이렇게 표시가 되면 E-Con driver 설치를 완료한 것이다.



11. E-Con driver 설치가 완료 된 후 장치 관리자를 확인하여 드라이버가 정상적으로 설치 되었는지를 확인한다.



12. 위처럼 정상적으로 driver 가 설치 되었는데도, 다음과 같은 메시지가 출력되면서 정상 동작하지 않는다면, PC 에 연결된 USB To Serial cable 을 모두 제거한 후 다시 확인을 해보기 바란다.

이렇게 시도하여 동작을 한다면, USB To Serial driver 와 E-Con driver 를 모두 제거한 후 다시 설치해서 사용하길 바란다.

```

Target is not connected
Target 을 검색 합니다.
에러 : E-CON 을 찾을 수 없습니다. 연결 상태를 확인 해 주세요.
에러 : Target 에 연결하지 못 했습니다.
"target" : 에러 : 명령어 수행을 완료하지 못했습니다.
EConMan('q' to exit) >

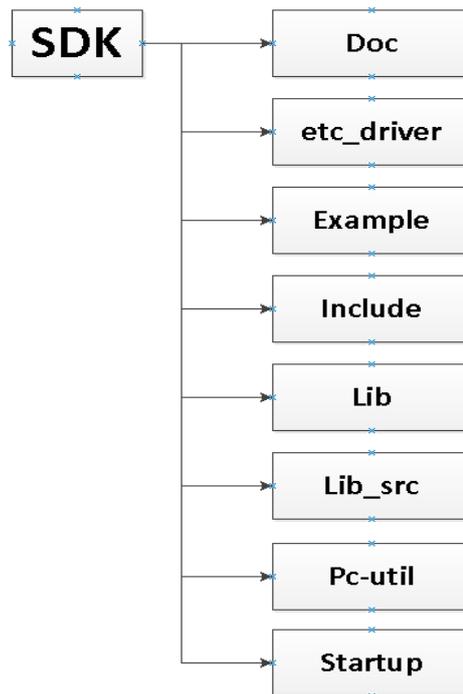
```

2. ADLUNA SDK 활용

개발에 편의를 위해 adchips 에서는 adStar-L SDK 를 제공하고 있다. adLuna SDK 는 adchips 의 홈페이지(www.adc.co.kr)의 Product →SoC→ ADLUNA → Tools & Software 에서 내려 받을 수 있다.

2.1 adLuna SDK 구성 및 기본 정의

adLuna SDK 는 다음과 같이 구성되어 있다.



Doc → 현재 보고 있는 매뉴얼을 포함, adLuna 와 관련된 문서가 들어있는 폴더.

Example →adLuna STK Board 에서 동작하는 예제 프로그램 폴더.

(Example 폴더의 flash_data 에는 SDK 예제 프로그램이 동작하는데 있어 필요한 image, sound 파일이 들어있다.)

Include →adLuna SDK 의 header file 폴더.

lib →adStar-L SDK 의 library file 폴더.

lib_src→adStar-L SDK 의 library source file 폴더.

Pc-util→adStar-L 을 사용하는데 있어 필요한 Utility 폴더. adStar-L USB Driver 가 들어있다.

Startup → adLuna 개발에 필요한 startup 코드와 link script 코드가 들어있는 폴더. 이 외에 STK board 기본 설정 코드도 들어있다.

adLuna SDK 에는 사용하기 편하고 구분이 쉽게 변수형을 아래와 같이 정의해놓았다. SDK Source 를 참조하는데 아래 표를 참고하기 바란다.

Signed char	S8, s8
Signed short	S16, s16
Signed int	S32, s32
Unsigned char	U8, u8, __u8, BYTE, uchar
Unsigned short	U16, u16, __u16, WORD, ushort
Unsigned int	U32, u32, __u32
Unsigned long	DWORD, ulong
Unsigned long long	U64, u64, __u64
Volatile unsigned char	vU8
Volatile unsigned short	vU16
Volatile unsigned int	vU32
Volatile unsigned long long	vU64
1	TRUE, true
0	FALSE, false

그리고 Register 의 경우 'R_' 접두어를 붙여 정의해놓았으므로 SDK Source 에서 R_이 붙은 명칭은 Register 로 생각하면 된다.

예를 들어 R_TM0CON 은 채널 0 번 timer control register 이다.

2.2 adLuna SDK library build

adLuna SDK 를 홈페이지에서 내려 받았으면, 제일 먼저 adLuna SDK Library 를 build 해주어야 한다. adLuna SDK Library source 는 lib_src 폴더에 있으므로, lib_src 폴더의 adLuna.epx 파일을 열어서 build menu 의 build project 를 실행하거나, F7 을 눌러서 project build 를 진행하면 된다. Build 가정상적으로 완료 되면, lib 폴더안에 libadLuna.a 파일이 생성 된다.

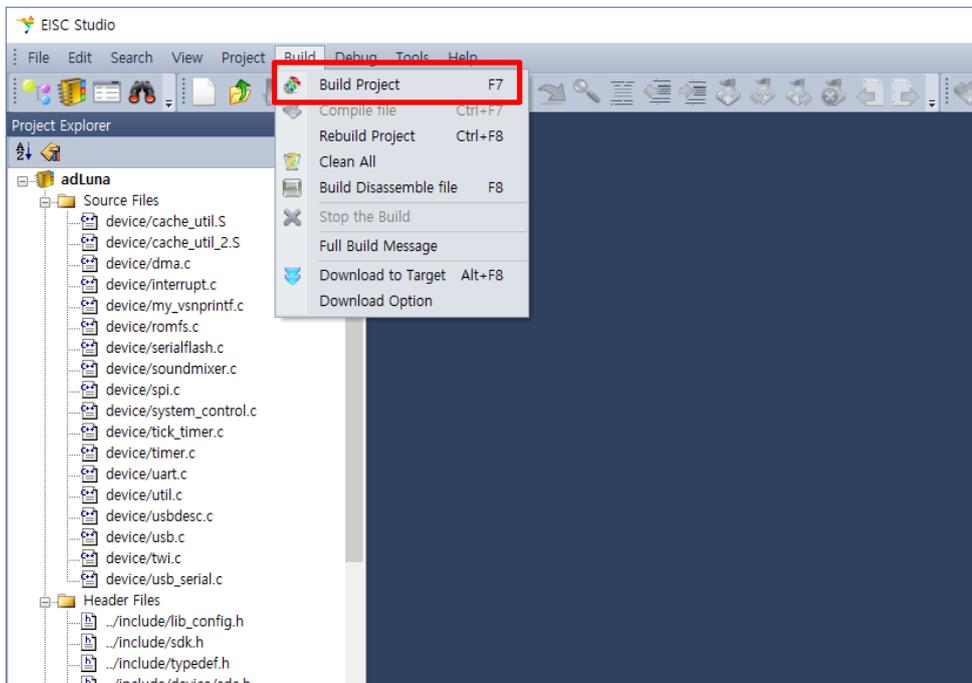
(adLuna SDK 의 모든 예제들이 library 를 참조하므로 제일 먼저 해주어야 한다.)

이름	수정한 날짜	유형
etc_driver	2019-02-22 오후...	파일 폴더
example	2019-02-22 오후...	파일 폴더
include	2019-02-22 오후...	파일 폴더
lib	2019-02-22 오후...	파일 폴더
lib_src	2019-02-22 오후...	파일 폴더
pc-util	2019-02-22 오후...	파일 폴더
startup	2019-02-22 오후...	파일 폴더

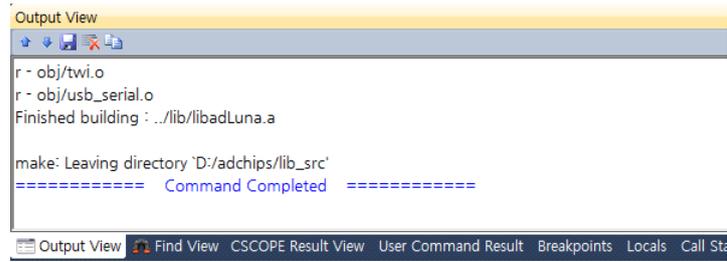
[SDK library]

이름	수정한 날짜	유형
device	2019-02-22 오후...	파일 폴더
adLuna.epx	2019-02-19 오후...	EISC Studio 3 Pro...

[lib_src 폴더의 adLuna.epx 파일]



[Build Project]



[Build Project 정상 완료]

이름	수정한 날짜	유형
libadLuna.a	2019-02-22 오후...	A 파일

[libadLuna.a 파일 생성]

만약에 adLuna library 를 수정하게 되면 위처럼 다시 library 를 build 해주어야 한다.
그리고 adLuna library 를 사용하는 program 도 다시 build 해주어야 수정 된 library 가 적용된다.

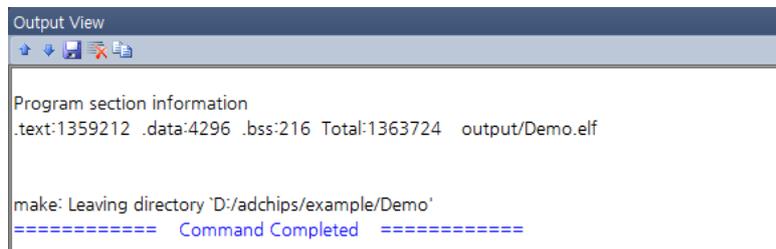
2.3 Demo Program 실행

adLuna SDK example 폴더에는 adLuna STK Board 에서 동작하는 예제들이 준비 되어 있어서, STK Board 에서 adLuna 동작을 쉽게 확인 해 볼 수 있다.

이번 장에서는 SDK 의 example 중 Demo Project 를 build, download 하여 실행하는 방법에 대해 알아보도록 하겠다.

2.3.1 Project build & download

example 폴더의 Demo 폴더를 열어 Demo.epx 파일을 마우스로 더블클릭하여 project 를 open 한다. Project 를 open 한 후 build → Build Project 를 실행하거나 F7 키를 눌러서 project 를 build 한다. build 가 완료되면 Output View 에 다음과 같이 출력된다.



```

Output View
Program section information
.text:1359212 .data:4296 .bss:216 Total:1363724  output/Demo.elf

make: Leaving directory 'D:/adchips/example/Demo'
==== Command Completed =====

```

[Output View]

Build 가 정상적으로 완료 되면, output 폴더에 Demo.elf.bin 파일이 생성 된 것을 확인할 수 있다.

다음으로 build 로 생성된 Demo.elf.bin 파일을 STK board 로 download 하는 방법에 대해 알아보겠다. Download 는 download 할 project 를 EISC Studio3 에 열어 놓은 상태에서 E-CON 을 사용하여 다음의 순서대로 진행하면 된다.

1. Download 를 하기 위해 adLuna board 와 E-CON 장비를 준비한다.
(EISC Studio 3 가 설치되어 있어야 하고, E-CON Driver 가 설치되어 있어야 한다. 설치 방법은 1 장의 Software 개발환경을 참고하기 바란다.)
2. adLuna board 와 E-CON 장비를 연결한 후 board 의 전원을 켜다.
3. Build 메뉴의 Download Option 을 클릭한다. 그러면 다음과 같이 Download Manager 창이 뜨는데, Set Command 에는 EISC Studio3 를 설치한 폴더에서 econ 이라는 폴더 안의 EConMan.exe 라는 E-CON 용 다운로드 프로그램을 선택해준다. (default 값인 EconMan.exe 로 하여도 된다.)

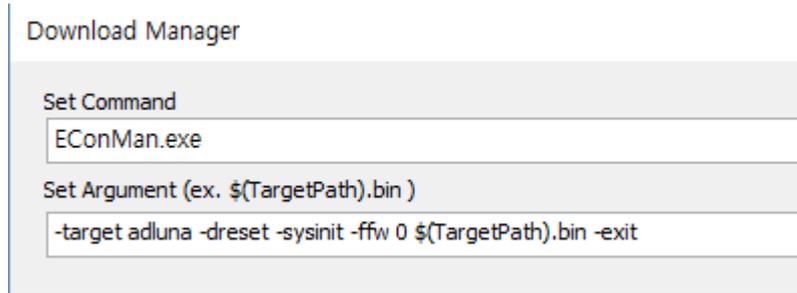
그리고 Set Argument 에는

```

-target adluna -dreset -sysinit -ffw 0x0 $(TargetPath).bin -exit"

```

이라고 입력하고 OK 를 클릭한다.



Argument 에 대해 설명하면,

- target** 은 말 그대로 download 할 target 을 말하며 adStar1 로 적어주면 된다.
 - sysinit** 는 download 를 위한 초기화를 해주는 command 이다.
 - ffw** 는 build 결과물로 나온 bin 파일을 다운로드 하는 command 로 0x0 은 download 할 주소이고, \$(TargetPath).bin 은 download 할 파일명으로 \$(TargetPath).bin 은 현재 열려있는 project 의 bin 파일을 뜻한다. Bootloader 를 사용하지 않고 프로그램을 실행시킬 때는, linker script 를 adstar-L.ld 로 사용하여 build 하고, 0 번지에 download 하면 된다.
 - exit** 는 download 가 완료되면 자동으로 종료하라는 command 이다.
- (Set Argument 에 대한 자세한 내용은 www.adc.co.kr → Product → System → E-CON → Download → E-CON.pdf 파일을 참고하기 바란다.)

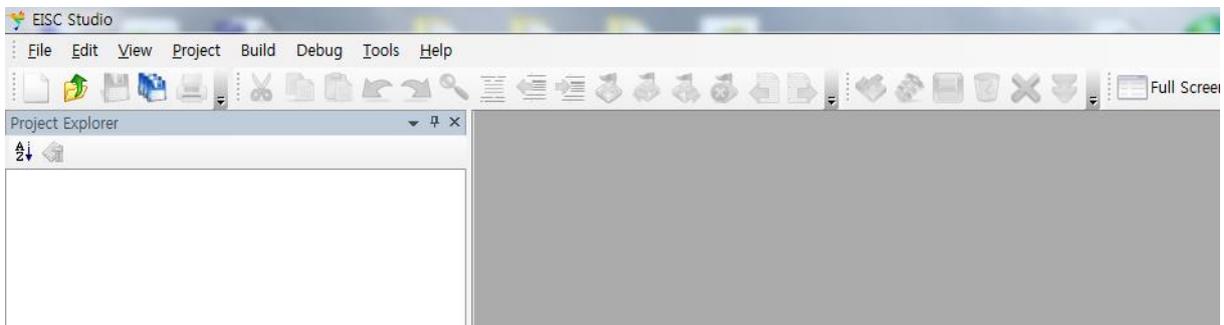
4. Build 메뉴의 Download to Target 을 클릭하면 download 가 진행이 된다.
5. Download 가 정상 완료 된 후 전원을 off 했다가 on 하면 동작을 확인할 수 있다.

2.4 adLuna 새프로젝트만들기

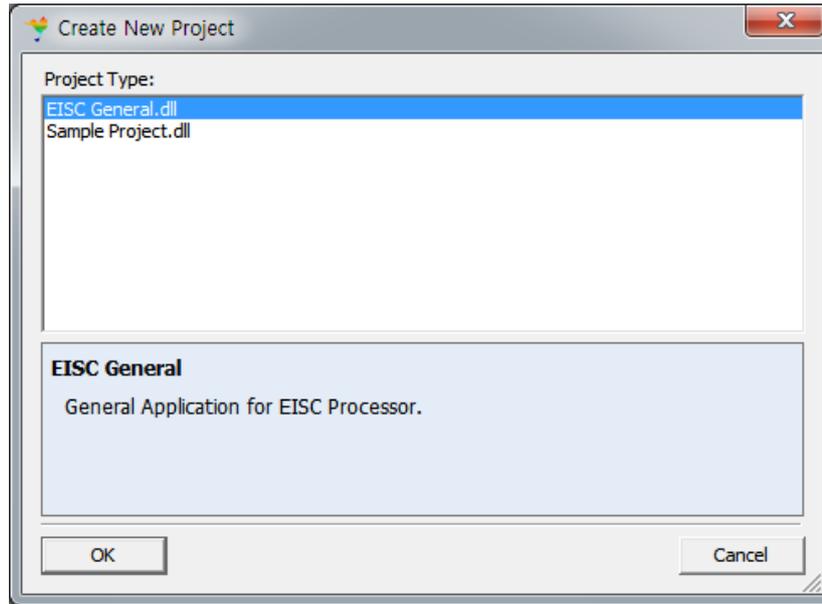
이번 장에서는“Hello adchips!”라는 문장을

UART 를통해출력하는프로그램을작성하면서새프로젝트를만드는법에대해알아보겠다.

1. 먼저 EISC Studio3 를실행한다.

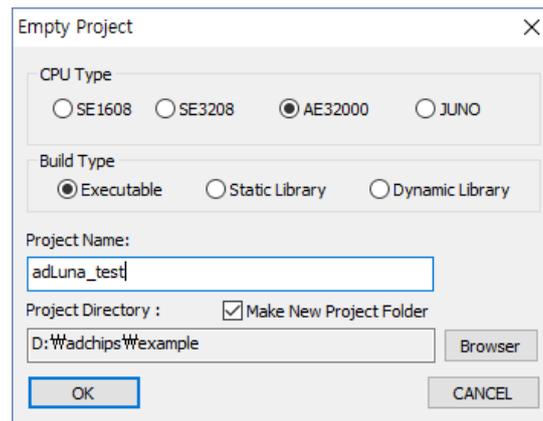


2. 메뉴의 File → New → Project 를 클릭하면 다음과 같은 창이 나오는데, 아래 그림과 같이 EISC General.dll 을선택한후 OK 버튼을 누른다.

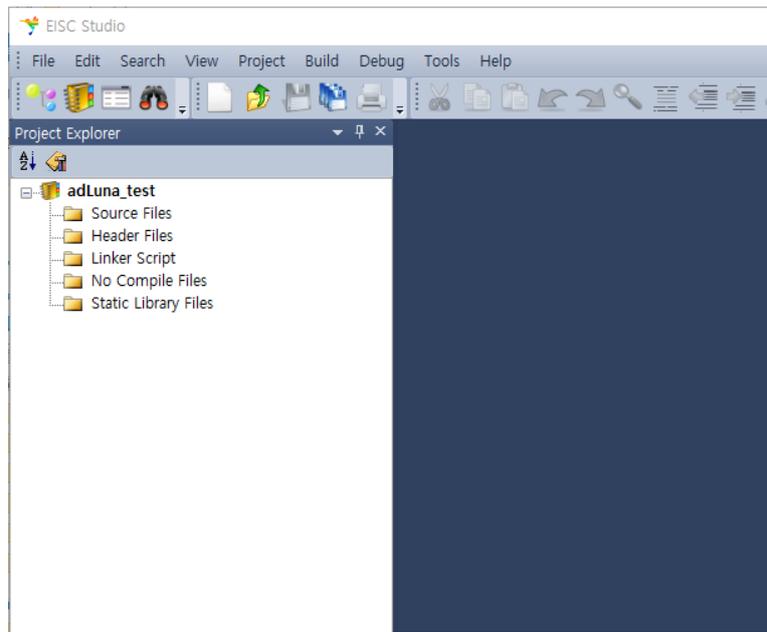


3. 다음과 같이, project 설정창이 출력되는데, CPU Type, Build Type, Project Name, Project Directory 를 결정해주어야 한다.

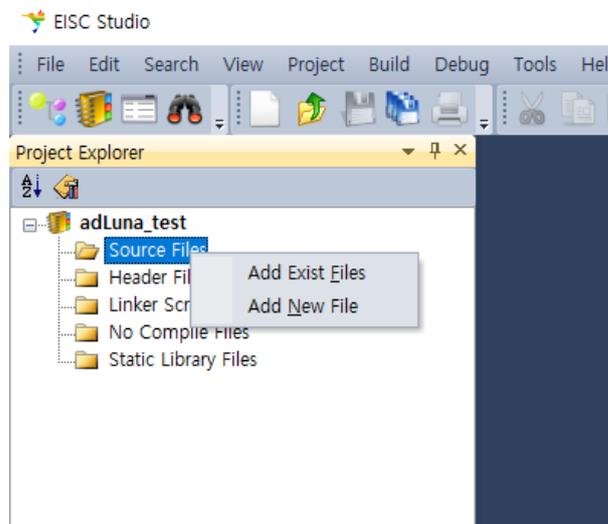
adStar-L 는 CPU Type 으로 AE32000 을 선택해주면 되고, 동작 프로그램을 작성할 것이기에 Build Type 으로 Executable 을 선택해준다. 다음으로 Project Name 과 Project 를 생성할 위치를 정해주면 된다. Project Directory 옆에 위치한 체크박스(Make New Project Folder)는 Project Name 과 같은 이름의 폴더를 생성유무를 결정하는 것으로, 체크하게 되면 Project Directory 로 지정해놓은 폴더에 Project Name 과 같은 이름의 폴더를 생성한 후 그 안에 Project file 을 생성해준다. 체크하지 않을 경우에는 Project Directory 로 지정해놓은 폴더에 Project file 을 생성한다. 모든 설정이 끝나면 OK 를 클릭하여 새 프로젝트 생성을 완료한다.



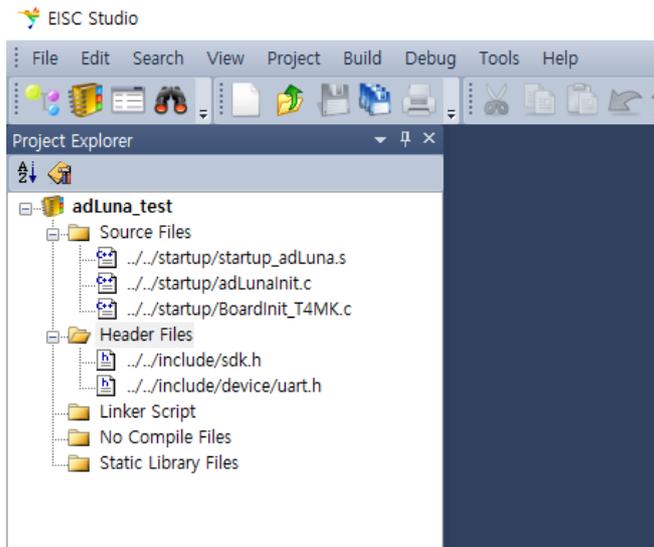
4. 새로운 프로젝트가 생성되면 다음과 같이 Project Explorer 창에 Project 명과 Project tree 를 확인할 수 있는데, 이곳에 adLuna 용 project 에 기본적으로 필요한 파일들을 추가해 주어야 한다.



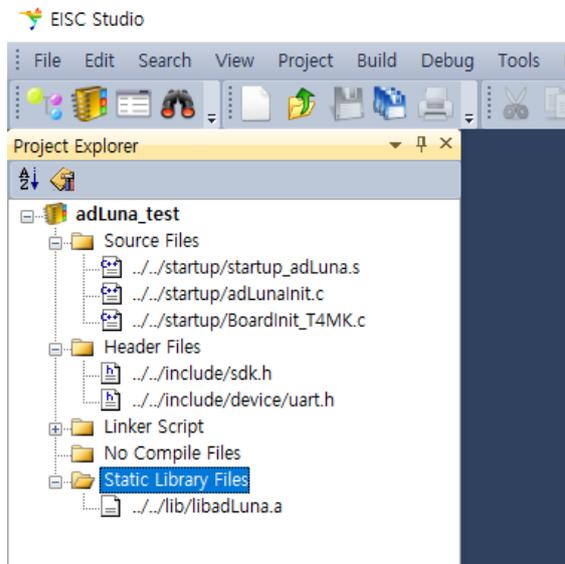
5. 먼저 Source Files 에서 마우스 오른쪽 키를 클릭하여 Add Exist Files 를 선택하여 파일을 추가한다. 추가할 파일은 SDK 의 startup 폴더에 있는 startup_adLuna.s, adLunaInit.c, BoardInit_T4MK.c 이다. (BoardInit_T4MK.c 파일은 adLuna stk board 를 사용할 경우 추가해주면 된다.)



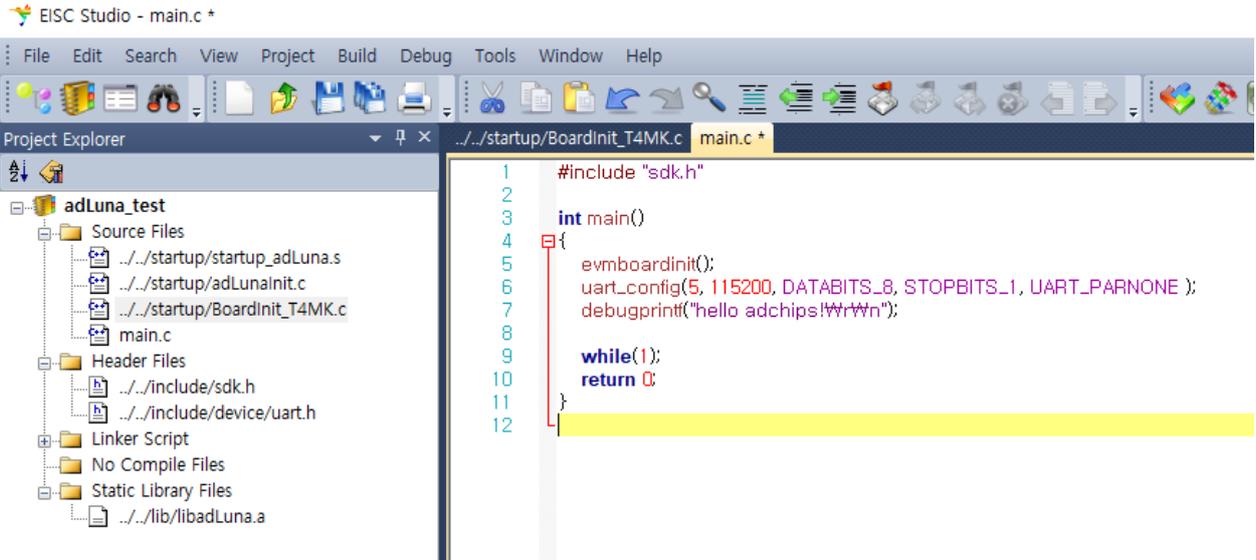
6. 다음으로 Source Files 와 마찬가지로 Header Files 에서도 마우스 오른쪽 키를 클릭하여 include 폴더에 있는 .h 파일을 추가해준다. header 파일을 추가하지 않아도 되지만, 프로그램 작성 시 함수 사용에 도움을 받기 위해서는 include 폴더와 include/device 폴더에 있는 관련 header 파일도 같이 추가해주면 좋다. 여기서는 UART 를 사용하는 프로그램을 작성해 볼 것이므로, UART.h 파일을 추가하도록 하겠다.



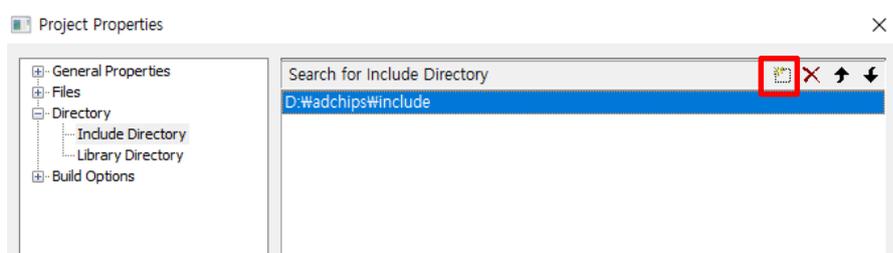
7. 다음으로 Linker Script 에 startup 폴더에 있는 adLuna.ld 파일을 추가하고, Static Library Files 에 lib 폴더에 있는 libadLuna.a 파일을 추가한다.



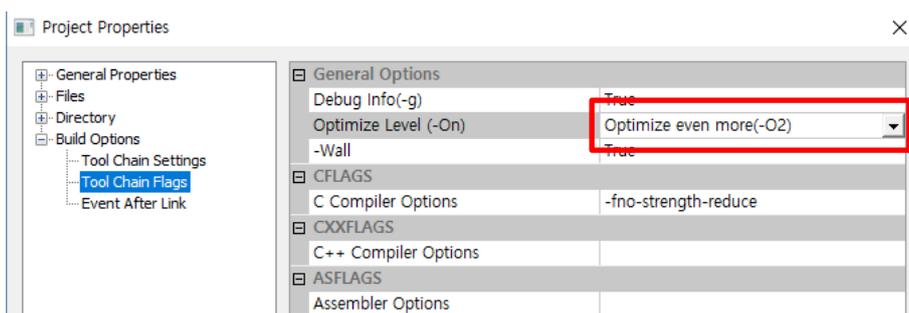
8. 위처럼 추가를 다 했으면 Source Files 에서 마우스 오른쪽 키를 누르고 Add New File 을 클릭 main 프로그램을 작성할 main.c 를 생성해주면 된다. 다음은 main.c 를 생성하고, UART 를 통해 Hello adchips!라는 문자열을 출력하는 코드를 작성 한 것이다.



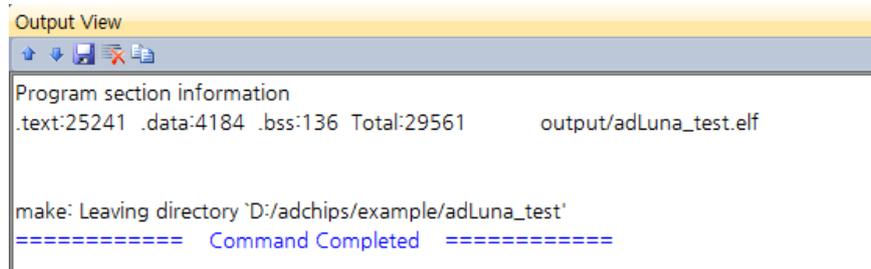
9. 코드를 살펴보면, 제일 먼저 evmboardinit() 함수를 호출하여 사용하는 board 에 맞게 pin 설정을 한다. (evmboardinit()함수는 사용하는 board 마다 다르므로 board 에 맞게 pin 설정을 해주어야 한다.) 다음으로 uart_config()함수를 호출하여 UART 를 설정한 값대로 초기화 해준다. 그리고 debugprintf()함수를 사용하여 "hello adchips!"를 출력한다.
10. 코드작성을 마쳤으면, project 를 build 해야하는데, 그 전에 먼저 project 설정을 해주어야 한다. Project Explorer 창에서 project 명에서마우스오른쪽을클릭하여 Properties 를클릭한다. 그러면 다음과 같은 창이 나오는데, Directory 에서 Include Directory 에 SDK 의 include 폴더를 추가해준다. (우측상단의 점선박스를 클릭하면 폴더를 추가할 수 있다.)



11. Project Properties → Build Options → Tool Chain Flags 에 Optimize Level (-Om)을 Optimize even more(-O2)로 변경하여 compile 시 최저과 옵션이 설정되도록 한다.
(이 과정은 필수는 아니지만, 동작 성능 향상을 위한다면 최적화를 하는 것이 좋다.
Debugging 을 할 경우에는 정확한 debugging 을 위해 이 값을 None (-O0)를 사용해야 한다.)



12. include 폴더까지 추가를 했으면 Build 메뉴에서 Build Project 를 클릭하거나, F7 을 누르면 Project Build 가 이루어진다. Build 가 정상적으로 완료 되었다면, Output View 창에 다음과 같이 출력되고, output 폴더에 project name.elf.bin 파일이 생성된다.



```
Output View
Program section information
.text:25241 .data:4184 .bss:136 Total:29561 output/adLuna_test.elf

make: Leaving directory `D:/adchips/example/adLuna_test`
==== Command Completed =====
```

13. 생성된 bin 파일을 E-Con 을 사용하여 STK board 에 download 하면 Uart 5 번 채널을 통해 hello adchips!가 출력되는 것을 확인 할 수 있다. (STK board 의 con3)

3. lib_config.h

adLuna SDK 의 include 폴더를 보면 lib_config.h 파일이 있는데, 파일명에서도 알 수 있듯이 lib_config.h 파일은 adLuna SDK library 의 설정 파일이라고 보면 된다. 이 장에서는 lib_config.h 파일에 대해 설명하도록 하겠다.

```

24  #pragma once
25
26  // #define NESTED_INTERRUPT_ENABLE
27
28  /*
29  |-----
30  | INTERNAL TIME OUT
31  |-----
32  */
33  /*
34  | Max count for TWI Function
35  |-----
36  */
37  #define TWI_RESP_TIME_OUT_COUNT (7200*100) // About 100ms @ AHB 101MHz

```

제일 처음 나오는 것은 nested_interrupt 를 허용할지를 결정하는 부분이 있다. Nested interrupt 를 허용하려는 경우 해당 주석을 해제한 후 library 를 rebuild 하면 된다. 다음은 TWI Timeout count 값을 설정 하는 부분이다.

```

47  /*
48  |-----
49  | UART Config
50  |-----
51  */
52  /*
53  | 인터럽트 모드일 경우  uart 송/수신 data 를 임시로 저장 할 버퍼의 사이즈이다.
54  */
55  #define UART_BUF_SIZE 128
56  #define CONFIG_UART_RX_INTERRUPT
57  // #define CONFIG_UART_TX_INTERRUPT
58
59  /*
60  | debugstring, debugprintf 함수의 최종 출력 채널
61  |-----
62  */
63  #define DEBUG_CHANNEL 5
64
65  /*
66  |-----
67  | SYSTEM CLOCK
68  |-----
69  */
70  /*
71  | external OSC
72  |-----
73  */
74  #define OSC_CLOCK 12000000

```

다음은 UART 의 ring buffer size 와 Rx/Tx interrupt 를 사용할지 여부를 결정하는 부분으로, 기본값은 buf size 128byte 에 UART Rx interrupt 를 사용하도록 되어 있다.

그리고 library 에서 동작 또는 error 와 관련된 message 를 출력하는데, 이 debug message 가 출력되는 UART channel 은 5 번으로 되어 있다.

이 부분 역시 변경하여 사용 가능하며, 변경 후에는 library 를 rebuild 해야 적용이 된다.

다음은 OSC_CLOCK 을 적어주는 부분으로, board 에 사용한 Crystal 값을 적어주는 곳이다. 기본 값은 개발보드에 12MHz crystal 이 붙어 있어 120000000 으로 되어 있다.

OSC_CLOCK 를 변경하는 경우 adLunaInit.c 파일의 pll_init 함수에서 입력 Clock 에 맞게 PLL setting 을 해주어야 원하는 동작 clock 으로 adLuna 를 동작시킬 수 있다.

```

71  /******
72  | * SOUND Mixer
73  |*****
74  /*
75  | 버퍼 사이즈를 줄이면 메모리 사용량은 줄어 들지만
76  | 인터럽트 발생 횟수가 늘어 난다.
77  |*/
78  #define WAVE_BUF_MAX (5*1024)
79  #define WAVE_RELOAD_BUF_MAX (1024*5)
80  #define DEFAULT_VOLUME 255//max 255
81
82  /*
83  | 최종 출력 sample rate
84  |*/
85  #define SND_OUTPUT_HZ 48000
86  #define SND_OUTPUT_CHANNEL 3
87

```

다음으로는 sound 출력 설정으로 sound buffer size 와 출력 Hz 와 출력 channel 설정에 대한 부분이다.

lib_config.h 파일의 모든 설정이 완료 되었으면 library 를 다시 build 하여 libadLuna.a 파일을 새로 생성하여 사용하여야 한다.

4. UART

4.1 uart_config

BOOL uart_config(**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 channel 값. 초기 설정할 channel 을 입력한다.

baud UART baud rate 값. Baud rate 값을 설정한다.
SDK 는 기본적으로 115200 을 사용한다.

databits UART 전송 data bit 을 설정한다. SDK 에서는 data bit 를 다음과 같이 정의하고 있다.

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

stopbit UART 의 stop bit 를 설정한다. SDK 에서는 stop bit 를 다음과 같이 정의하고 있다.

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

parity UART 의 parity bit 를 설정한다. SDK 에서는 parity bit 를 다음과 같이 정의하고 있다.

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

Return Value

TRUE(1) or FALSE(0)

4.2 uart_putchar

BOOL uart_putchar(**int** n, **char** ch)

UART n 채널을 통해 ch 값(한 문자)을 출력한다.

Parameter

n 값을 출력할 UART channel 값. adStar-L 는 UART 5 개 channel 을 가지고 있다.
Ch UART 를 통해 출력할 값.

Return Value

TRUE(1) or FALSE(0)

4.3 uart_putdata

BOOL uart_putdata(**int** n, U8* buf, int len)

UART n 채널을 통해 buf 에 저장되어 있는 문자를 len 만큼 출력한다..

Parameter

n 값을 출력할 UART channel 값. adStar-L 는 UART 5 개 channel 을 가지고 있다.
buf 출력할 문자열이 저장되어 있는 buffer.
len 출력할 문자열의 개수.

Return Value

TRUE(1) or FALSE(0)

4.4 uart_putstring

BOOL uart_putstring(**int** n, U8* buf)

UART n 채널을 통해 buf 에 저장되어 있는 문자열을 출력한다.

Parameter

n 값을 출력할 UART channel 값. adStar-L 는 UART 5 개 channel 을 가지고 있다.
buf 출력할 문자열이 저장되어 있는 buffer.

Return Value

TRUE(1) or FALSE(0)

4.5 uart_getch

BOOL uart_getch(**int** n, **char*** ch)

UART n 채널을 통해 1Byte 값(한 문자)을 받아 ch 에 저장한다.

Parameter

n 값을 입력 받을 UART channel 값. adStar-L 는 UART 5 개 channel 을 가지고 있다.
ch UART 를 통해 값을 입력 받을 변수.

Return Value

TRUE(1) or FALSE(0)

4.6 uart_getdata

int uart_getdata(**int** n, **U8*** buf, **int** bufmax)

UART n 채널을 통해 bufmax 만큼 값(문자)을 읽어서 buf 에 저장한다.

Parameter

n 값을 입력 받을 UART channel 값.
Buf UART 를 통해 값을 입력 받을 buffer.
Bufmax 값(문자)을 읽을 개수. (Byte 단위)

Return Value

읽은 data byte 수. Bufmax 값과 비교하여 같으면 함수가 정상적으로 처리 한 것이고, 그렇지 않을 경우에는 정상 처리하지 못한 것이다.

4.7 uart_rx_flush

void uart_rx_flush(**int** ch)

UART n 채널의 rxfifo 를 초기화 시킨다.

Parameter

ch 초기화 시킬 UART channel 값.

Return value

없음.

4.8 uart_tx_flush

void uart_tx_flush(**int** ch)

UART n 채널의 txfifo 를 초기화 시킨다.

Parameter

ch 초기화 시킬 Uart channel 값.

Return value

없음.

4.9 set_debug_channel

Void set_debug_channel(**int** ch)

디버깅용으로 사용하는 **debugprintf**, **debugstring**, **PRINTVAR**, **PRINTLINE** 함수에 의해 디버깅 메시지가 출력 될 UART 채널을 결정한다.

Parameter

ch 디버깅용으로 사용할 UART 채널.

Return value

없음.

4.10 get_debug_channel

int get_debug_channel()

디버깅용으로 사용하는 UART 채널 값을 return 한다.

Parameter

없음

Return Value

현재 설정된 디버깅용 UART 채널 값.

4.11 debugprintf

void debugprintf(**const char***const format, . . .)

c 언어의 printf 와 같은 역할을 하는 함수로, UART 를 통해, 숫자, 문자, 변수의 내용을 출력할 때 사용한다. 출력되는 UART 채널은 set_debug_channel 함수를 통해 설정해 놓은 디버깅 채널을 사용한다. 기본으로 0 번 채널이 설정되어 있다. 참고로 lib_config.h 파일에서 Tx interrupt 를 사용하도록 설정하여도, 이 함수는 polling 방식으로 동작한다.

Usage

```
debugprintf("result number : %dWrWn",result);
```

→UART 를 통해 "result number ."라는 문자열과 result 변수 값을 10 진수로 출력한다.

그리고 줄 바꿈을 한다. printf 함수와 사용법이 같다. 단 줄 바꿈 시 WrWn 을 모두 써주어야 한다.

4.12 debugstring

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

문자열을 출력하는 함수로, 문자열만 출력하기를 원할 때 사용하는 함수이다. 출력되는 UART 채널은 set_debug_channel 함수를 통해 설정해 놓은 디버깅 채널을 사용한다. 기본으로 0 번 채널이 설정되어 있다. 참고로 debugprintf 함수를 사용하여도 문자열만 출력 가능하며, lib_config.h 파일에서 Tx interrupt 를 사용하도록 설정하여도, 이 함수는 polling 방식으로 동작한다.

Usage

```
debugstring("=== adStar-L Start ===WrWn");
```

→""안의 문자열을 UART 를 통해 출력한다.

4.13 PRINTLINE

매크로 함수로 PRINTLINE 을 호출한 곳의 줄(line)값을 UART 통해 출력한다. 채널은 설정해 놓은 디버깅 채널을 사용한다. 출력되는 UART 채널은 set_debug_channel 함수를 통해 설정해 놓은 디버깅 채널을 사용한다. 기본으로 0 번 채널이 설정되어 있다.

Usage

```
PRINTLINE;
```

→ 함수를 호출한 곳의 line 값을 출력한다..

4.14 PRINTVAR(A)

매크로 함수로 PRINTVAR 를 호출한 곳의 줄(line)값과 A 값을 UART 를 통해 출력한다.출력되는 UART 채널은 set_debug_channel 함수를 통해 설정해 놓은 디버깅 채널을 사용한다. 기본으로 0 번 채널이 설정되어 있다.

Usage

```
int a = 10;
```

```
PRINTVAR(a);;
```

→ 변수 a 의 값을 출력한다. 레지스터 값을 출력 할 수도 있다.

4.15 UART Example

```
#include "sdk.h"
```

```
Int main()
```

```
{
    evmboardinit();
    uart_config(5, 115200, DATABITS_8, STOPBITS_1, UART_PARNONE );
        // UART 5 번 채널을 다음과 같이 설정한다.
        // Baud rate = 115200
        // Data bit = 8bit
        // Stop bit = 1
        // Parity = none

    debugstring("====WrWn");
    debugprintf("ADLUNA UART Example. System clock(%dMhz)WrWn",get_ahb_clock()/1000000);
    debugstring("====WrWn");

    U8 ch;
    While(1)
    {
        if(uart_getch(5, &ch)    // UART 1 번에서 1Byte 데이터를 읽어 ch 에 저장한다.
        {
            uart_putch(5, ch);// UART 1 번에서 입력 받은 데이터를 UART 0 번으로 출력한다.
        }
    }
}
```

< UART Interrupt >

adStar-L SDK library 는 lib_config.h 파일의 정의에 따라 UART Interrupt 를 사용하도록 설정되어 있다.

Default 값은 UART Rx Interrupt 를 설정하도록 되어 있다.

```

47  | /******
48  |   UART Config
49  | *****/
50  | /*
51  |   인터럽트 모드일 경우   uart 송/수신 data 를 임시로 저장 할 버퍼의 사이즈이다.
52  | */
53  | #define UART_BUF_SIZE 128
54  | #define CONFIG_UART_RX_INTERRUPT
55  | //#define CONFIG_UART_TX_INTERRUPT
56  |
57  | /*
58  |   debugstring, debugprintf 함수의 최종 출력 채널
59  | */
60  | #define DEBUG_CHANNEL 5
61  | ..

```

< lib_config.h >

기본적으로 UART Rx Interrupt 를 사용하도록 설정 되어 있기 때문에, uart_config()함수를 호출하면 UART 초기화를 진행하고, Interrupt 를 설정한다. Interrupt 가 설정된 상태에서 UART Interrupt 가 발생하면 uart.c 의 UARTxISR()가 호출되어 Interrupt Routine 을 수행한다. (uart.c, uart.h 참조)

5. Interrupt

5.1 init_interrupt

```
void init_interrupt(void);
```

interrupt 를 초기화 하는 함수이다. 인터럽트 관련 함수를 사용하기 위해서 반드시 호출해주어야 하는 함수로 adStar-L SDK 의 startup_adStar-L.s 에서 호출하여 인터럽트를 초기화한다.

5.2 set_interrupt

```
BOOL set_interrupt(INTERRUPT_TYPE intnum, void (*fp)());
```

해당 interrupt 가 발생하였을 때 호출할 함수를 등록하는 함수이다.

Uart, Sound mixer 등의 Interrupt 함수는 SDK 에서 생성 및 설정 되어 있으므로 중복 등록하지 않도록 주의하기 바란다.

Parameter

intnum	interrupt type (interrupt type 에 대한 정보는 다음 장 참고)
(*fp)()	interrupt 가 발생했을 때 호출할 함수

Return value

TRUE or FALSE

5.3 enable_interrupt

```
void enable_interrupt(INTERRUPT_TYPE intnum, BOOL b);
```

해당 interrupt 를 enable 하는 함수이다.set_interrupt 함수로 interrupt 함수를 등록하고, enable_interrupt 함수로 활성화 시킨다.

Parameter

intnum	interrupt type (interrupt type 에 대한 정보는 다음 장 참고)
b	1 or TRUE => interrupt enable
	0 or FALSE => interrupt disable

Return value

TRUE or FALSE

INTERRUPT_TYPE

INTERRUPT_TYPE		INTERRUPT_TYPE	
INTNUM_SWD	SWD interrupt	INTNUM_GPIO7	GPIO 7 interrupt
INTNUM_TICKTIMER	TICK Timer interrupt	INTNUM_GPIO8	GPIO 8 interrupt
INTNUM_TIMER0	Timer 0 interrupt	INTNUM_SPI0	SPI 0 interrupt
INTNUM_TIMER1	Timer 1 interrupt	INTNUM_SPI1	SPI 1 interrupt
INTNUM_TIMER2	Timer 2 interrupt	INTNUM_TWI0	TWI 0 interrupt
INTNUM_TIMER3	Timer 3 interrupt	INTNUM_TWI1	TWI 1 interrupt
INTNUM_TIMER4	Timer 4 interrupt	INTNUM_DMA0	DMA 0 interrupt
INTNUM_UART0	UART 0 interrupt	INTNUM_DMA1	DMA 1 interrupt
INTNUM_UART1	UART 1 interrupt	INTNUM_DMA2	DMA 2 interrupt
INTNUM_UART2	UART 2 interrupt	INTNUM_DMA3	DMA 3 interrupt
INTNUM_UART3	UART 3 interrupt	INTNUM_CAP_OVER0	Captureover 0 interrupt
INTNUM_UART4	UART 4 interrupt	INTNUM_CAP_OVER_2	Captureover 2 interrupt
INTNUM_UART5	UART 5 interrupt	INTNUM_CAP_OVER4	Captureover 4 interrupt
INTNUM_GPIO0	GPIO 0 interrupt	INTNUM_ADC	ADC interrupt
INTNUM_GPIO1	GPIO 1 interrupt	INTNUM_USB	USB interrupt
INTNUM_GPIO2	GPIO 2 interrupt	INTNUM_SOUND	Soundmixer interrupt
INTNUM_GPIO3	GPIO 3 interrupt	INTNUM_WATCHDOG	Watch dog interrupt
INTNUM_GPIO4	GPIO 4 interrupt	INTNUM_PDBG	PDBG interrupt
INTNUM_GPIO5	GPIO 5 interrupt	INTNUM_PMC	PMC interrupt
INTNUM_GPIO6	GPIO 6 interrupt		

5.4 Interrupt Example

```
#include "sdk.h"

void GPIO_ISR( )
{
    debugprintf("GPIO 6.2 Falling edge ISRWrWn");
}

int main()
{
    evmboardinit();
    uart_config(5, 115200, DATABITS_8, STOPBITS_1, UART_PARNONE );

    *R_GPODIR(6) = 1 << 2; // P6.2 Input

    *R_GPIRQMODE(6) = 0x2; // Falling interrupt Mode
    *R_GPIRQEN(6) = 1 << 2; // P6.2 Interrupt enable
    set_interrupt( INTNUM_GPIO6, GPIO_ISR );
        //GPIO 6 interrupt 가 발생했을 때 호출 될 함수를 등록한다.
    enable_interrupt( INTNUM_GPIO, TRUE );
        // GPIO 6 interrupt 를 enable 시킨다.

    while(1)
        return 0;
}
```

- ◆ GPIO interrupt 는 한 Port(GPx.0~7)당 1 개의 핀만 interrupt 로 활성화 시킬 수 있다. R_GPIRQ register 에 한 bit 만 1 로 설정해야 한다. 두 개 이상 설정하면 아무런 비트도 설정되지 않는다.

6. TIMER

6.1 set_timer

BOOL set_timer(**int** nCh, **U32** ms)

nCh 채널 timer 를 설정 및 동작시키는 함수이다. 사용할 timer 채널과 주기를 결정하고 timer control register 에서 timer 동작을 enable 한다. 그리고 마지막으로 timer interrupt 를 enable 한다.

set_interrupt 함수로 timer interrupt 등록된 후 이 함수를 호출하면 설정한 주기만큼 timer interrupt 가 발생한다.

Parameter

nCh	설정할 timer 채널 값.
ms	timer interrupt 주기. (ms 단위)

Return Value

TRUE(1) or FALSE (0)

6.2 stop_timer

BOOL stop_timer(**int** nCh)

nCh 채널 timer 를 정지시키는 함수이다. nCh 채널의 timer 동작을 disable 시킨다.

Parameter

nCh	정지시킬 timer 채널 값.
-----	------------------

Return Value

TRUE(1) or FALSE (0)

6.3 delayms

BOOL delayms(**U32** ms)

ms 만큼 delay 가 걸린다. (단위는 ms 이다)

Parameter

ms	delay 걸리는 시간 단위는 ms.
----	----------------------

Return Value

TRUE(1) or FALSE (0)

6.4 set_pwm

void set_pwm(U32 nCh, int hz, int number, bool bperiod, bool en)

pwm 파형을 출력한다.

Parameter

nCh	pwm 출력할 timer 채널.
hz	pwm 출력할 주파수.
number	pwm 출력할 파형의 개수. One period 모드일 경우 해당 number 만큼 파형을 출력하고 멈춘다. Interrupt 를 설정해 놓으면 해당 number 만큼 파형 출력 후 interrupt 를 발생시킨다.
bperiod	period mode 를 설정한다. true = period mode, pwm 파형이 계속 출력된다. false = one period mode, pwm 파형이 number 만큼 출력되고 stop 한다.
en	pwm 출력 enable/disable 을 결정한다.

6.5 TIMER Example

```
#include "sdk.h"

void TIMER0ISR( )
{
    debugprintf("==TIMER0ISR==WrWn");
}

int main()
{
    boardinit();
    uart_config(5, 115200, DATABITS_8, STOPBITS_1, UART_PARNONE );
    set_interrupt( INTNUM_TIMER0, TIMER0ISR );
    //Timer 0 번 Interrupt 가 발생하였을 때 호출 될 interrupt 함수를 등록한다.
    set_timer( 0, 1000);
    // 1 초마다 Timer0 Interrupt 가 발생한다.
    delayms(5000);
    // 5 초간 delay 가 발생.
    stop_timer( 0 );
    // Timer 0 번 Interrupt 를 disable 시킨다. 더 이상 Timer 가 발생하지 않음.
    while(1)
    return 0;
}
```

7. Sound

이번 장에서는 adLuna sound 출력에 대해 알아보겠다.
adLuna 는 8bit/16bit, mono/stereo WAV 를 지원한다.

7.1 sound_init()

BOOL sound_init()

Sound 출력을 위해 Soundmixer 를 초기화 하는 함수.

Parameter

없음.

Return value

없음.

7.2 sound_loadwav

WAVE* sound_loadwav(**char*** filename);

WAV 파일을 load 하는 함수. WAV 파일을 load 하여 WAVE 구조체를 생성 메모리에 저장한다.

Parameter

filename WAV 파일

Return value

WAV 파일의 정보와 데이터가 저장된 WAVE 구조체.

7.3 sound_loadwavp

WAVE* sound_loadwavp(**U8*** buf, **U32** len);

WAV data 를 메모리에서 load 하는 함수.

Parameter

buf WAV data 가 저장되어 있는 buffer.
len WAV data 의 길이.

Return value

WAV 파일의 정보와 데이터가 저장된 WAVE 구조체.

7.4 sound_release

BOOL sound_release(**WAVE*** pWave);

load_soundwav() 함수에 의해 생성된 메모리를 해제한다. 사용할 수 있는 메모리가 제한적이기 때문에 사용하지 않는 sound 대해 sound_release()함수를 호출하여 사용하지 않는 메모리를 반환하는 것이 좋다.

Parameter

pWave sound_loadwav()함수에 의해 생성된 WAVE 구조체.

Return value

TRUE(1) or FALSE(0)

7.5 sound_play

BOOL sound_play(**WAVE*** pWave);

load_soundwav() 함수로 load 한 sound 를 play 한다.

Parameter

pWave sound_loadwav()함수에 의해 생성된 WAVE 구조체.

Return value

TRUE(1) or FALSE(0)

7.6 sound_stop

BOOL sound_stop(**WAVE*** pWave)

play 중인 sound 를 stop 시킨다.

Parameter

pWave 현재 play 중인 sound 의 WAVE 구조체.

Return value

TRUE(1) or FALSE(0)

7.7 sound_vol

```
void sound_vol(U8 vol);
```

sound volume 을 조정한다.

Parameter

vol volume 값. 범위는 0 부터 255 이다.

Return value

없음

7.8 sound_vol_wav

```
void sound_vol_wav(WAVE* pWav, U8 vol);
```

음원 파일 각각의 volume 을 조정한다. sound_vol() 함수와 별도로 volume 을 설정한다. 두 개의 음원을 동시에 출력할 때 음원 마다 volume 크기를 다르게 하고 싶을 경우에 사용하면 된다.

Parameter

pWave volume 을 조정할 sound 구조체
vol volume 값. 범위는 0 부터 255 이다.

7.9 sound_pause

```
BOOL sound_pause(WAVE* pWave);
```

Play 중인 sound 를 pause 시킨다.

Parameter

pWave 현재 play 중인 sound 의 WAVE 구조체.

Return value

TRUE(1) or FALSE(0)

7.10 sound_resume

```
BOOL sound_resume(WAVE* pWave);
```

Pause 상태의 sound 를 resume 시킨다.

Parameter

pWave 현재 pause 중인 sound 의 WAVE 구조체.

Return value

TRUE(1) or FALSE(0)

7.11 sound_isplay

BOOL sound_isplay(**WAVE*** pWave);

현재 pWave sound 가 play 중인지 확인할 수 있다.

Parameter

pWave 현재 play 중인지 확인하려는 sound 의 WAVE 구조체.

Return value

TRUE(1 = play) or FALSE(0 = stop)

7.12 sound_isplay

BOOL sound_isplay(**WAVE*** pWave);

현재 pWave sound 가 pause 중인지 확인할 수 있다.

Parameter

pWave 현재 pause 중인지 확인하려는 sound 의 WAVE 구조체.

Return value

TRUE(1 = pause) or FALSE(0 = none pause)

7.13 Sound Example

< WAV File Play >

```
#include "sdk.h"

INCBIN(okwave, "01.wav"); // Add 01.wav file data

Int main()
{
    evmboardinit();
    uart_config(5,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);
    debugstring("====WrWn");
    debugprintf("ADLUNA Wave Play.System Clock(%dMhz)WrWn",get_ahb_clock()/1000000);
    debugstring("====WrWn");

    sound_init();                // Sound 출력을 위한 sound mixer 초기화
    WAVE* pWavOK = sound_loadwvp(okwave_data, okwave_end - okwave_data);
                                // wav 파일 load

    sound_play(sound);           // wav 파일 재생
    delayms(1000);
    If(sound_isplay(sound))      // 1 초후에 wav 파일 재생 중인지 확인
    {
        sound_stop(sound);      // 재생 중이면 재생 중지
    }

    sound_release(sound);        // 더 이상 사용하지 않을 sound 라 메모리에서 제거.

    ...
    ...

    while(1);
    return 0;
}
```

8. TWI

8.1 twi_set_freq

void twi_set_freq(**U8** ch, **U32** freq)

TWI 동작 주파수를 설정한다. (100KHz or 400KHz) TWI 는 2 개 channel 을 지원하는데, 원하는 channel 을 선택하여 사용할 수 있다.

Parameter

ch channel 값.
freq 주파수 값.

8.2 twi_write

bool twi_write(**U8** ch, **U8** devaddr, **U8** reg, **U8** data)

TWI write 동작을 수행한다. Register (or memory)에 1byte data 를 write 한다.

Parameter

ch channel 값.
devaddr device address.
reg register or memory address.
data write data.

Return value

TRUE(1) ot FALSE(0).

8.3 twi_write_multi

bool twi_write_multi(**int** ch, **U8** devaddr, **U8*** reg, **int** len, **U8*** data, **int** length)

TWI write 동작을 수행한다. 1byte 이상의 register (or memory) address 에 1byte 이상의 data 를 write 할 경우 사용한다.

Parameter

ch channel 값.
devaddr device address.
reg register or memory address 가 들어있는 buffer.
len register or memory address 길이.
data write 할 data buffer.
length data 개수.

Return value

TRUE(1) ot FALSE(0).

8.4 twi_read

int twi_read(**U8** ch, **U8** devaddr, **U8*** reg, **int** len, **U8*** buf, **int** length)

TWI read 동작을 수행한다.

Parameter

ch	channel 값.
devaddr	device address
reg	register or memory address buffer
len	register or memory address 길이
buf	read data buffer
length	read data 개수

Return value

Read data count.

8.5 TWI Example

<TWI EEPROM Example>

```
#include "sdk.h"

#define AT24CXX_EEPROM_ADDR (0x50<<1)

#define AT24CXX_EEPROM_PAGESIZE 32
#define AT24CXX_EEPROM_MAXADDR (4096) //4096 Byte=32*1024 bit

#define TWI_CH 0

int at24cxx_eeprom_write (U16 addr, U8 *buffer, int length)
{
    U32 startaddr = addr;
    U32 endaddr = addr+length-1;
    U32 startpage;
    U32 endpage;
    U32 offset;
    U32 i;
    int wcnt=0,len, ret;
    U8 waddr[2];

    if ((length <= 0) || (addr > AT24CXX_EEPROM_MAXADDR))
    {
        debugstring("Invalid Address\n\r");
        return -1;
    }

    startpage = startaddr / AT24CXX_EEPROM_PAGESIZE;
    endpage = endaddr / AT24CXX_EEPROM_PAGESIZE;

    for(i=startpage;i<endpage+1;i++)
```

```

    {
        offset = addr%AT24CXX_EEPROM_PAGESIZE;

        if(offset + length > AT24CXX_EEPROM_PAGESIZE)
        {
            len = AT24CXX_EEPROM_PAGESIZE - offset;
        }
        else
        {
            len = length;
        }

        waddr[0] = addr >> 8;
        waddr[1] = addr & 0xff;
        ret = twi_write_multi (TWI_CH, AT24CXX_EEPROM_ADDR, waddr, 2, buffer, len);
        if (ret < 0) {
            PRINTLINE;
            break;
        }

        /* Write Cycle Time of AT24CXX EEPROM: 5ms */
        delayms(5);

        buffer += len;
        addr += len;
        wcnt += len;
        length -= len;
    }
    return wcnt;
}

```

int at24cxx_eeprom_read (u16 addr, u8 *buffer, int length)

```

{
    int rcnt;
    U8 waddr[2];
    if ((length <= 0) || (addr > AT24CXX_EEPROM_MAXADDR)) {
        debugstring("Invalid Address\n\r");
        return -1;
    }

    waddr[0] = addr >> 8;
    waddr[1] = addr & 0xff;

    rcnt = twi_read (TWI_CH , AT24CXX_EEPROM_ADDR, waddr, 2, buffer, length);
    return rcnt;
}

```

int main()

```

{

    evmboardinit();
    uart_config(5,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);
    debugstring("=====\r\n");
    debugprintf("adLuna.System Clock(%dMhz)\r\n",get_ahb_clock()/1000000);
    debugstring("=====\r\n");

    twi_set_freq(0, 100000);

    U8 wbuf;
    U8 rbuf;

    int i = 0;

```

```
for(i=0; i<10; i++)
{
    wbuf[i] = i;
    rbuf[i] = 0;
}

at24cxx_eeprom_write (0, wbuf, 10);
at24cxx_eeprom_read (0, rbuf, 10);

while(1);
return 0;
}
```

9. SPI

9.1 spi_master_init

```
void spi_master_init(int ch)
```

SPI master 초기 설정을 한다. SPI 2 개 channel 을 지원하는데, 원하는 channel 을 선택하여 사용할 수 있다.

Parameter

ch channel 값.

9.2 spi_set_freq

```
int spi_set_freq(int ch, int mode, U32 freq)
```

SPI baud rate 을 설정한다.

Parameter

ch channel 값.
 mode master/slave 값. (SPI_MASTER / SPI_SLAVE)
 freq frequency 값.

Return value

Baud rate 값.

9.3 spi_master_xfer

```
void spi_master_xfer(int ch, U8 *wbuf, int wlength, U8 *rbuf, int rlength, int continue_xfer)
```

wlength 만큼 wbuf 의 데이터를 Slave 로 전송하고, rlength 만큼 rbuf 에 데이터를 Slave 로부터 읽어온다.

Parameter

ch channel 값.
 wbuf 전송할 data buffer.
 wlength 전송할 data 길이.
 Rbuf 읽어올 data buffer.
 rlength 읽어올 data 길이.
 Continue_xfer 계속 전송 여부 결정.

9.4 spi_wait_empty_fifo

```
void spi_wait_empty_fifo(int ch)
```

SPI Tx FIFO 가 비었는지를 확인한다.

Parameter

ch channel 값.

9.5 SPI Example

<SPI EEPROM Example>

=== w25xxx_flash.h ===

```
#ifndef __W25Xxx_H__
#define __W25Xxx_H__
int w25xxx_write_buffer (U32 addr, U8 *buffer, int length);
int w25xxx_read_buffer (U32 addr, U8 *buffer, int length);
int w25xxx_erase (U32 start_addr, int length);
int w25xxx_block_erase (U32 addr, int length);
int w25xxx_wait_ready (void);
int w25xxx_check_id (void);

#define W25XXX_PAGE_SIZE       (256)
#define W25XXX_SECTOR_SIZE     (4096)
#define W25XXX_BLOCK_SIZE     (16*W25XXX_SECTOR_SIZE)

#define W25X16_TOTAL_SIZE     (2*1024*1024)
#define W25X32_TOTAL_SIZE     (64*W25XXX_BLOCK_SIZE)
#define W25X64_TOTAL_SIZE     (128*W25XXX_BLOCK_SIZE)
#define W25X32_FLASH_MAXADDR   W25X32_TOTAL_SIZE

// W25Xxx: 3.3V -> Max ??MHz
#define W25Xxx_SPI_FREQ   23000000   // 23MHz

// W25Xxx Flash instruction
#define W25Xxx_WREN        0x06    // Write Enable
#define W25Xxx_WRDI       0x04    // Write Disable
#define W25Xxx_RDSR       0x05    // Read Status Register
// #define W25Xxx_WRSR     0x01    // Write Status Register
```

```

#define W25Xxx_READ      0x03    // Read Data Bytes
#define W25Xxx_FAST_READ 0x0B    // Read Data Bytes at Higher Speed
#define W25Xxx_DUAL_READ 0x3B    // Read Data Bytes at Higher Speed
#define W25Xxx_Page_Program 0x02  // Page Program
#define W25Xxx_BERASE   0xD8    // Block Erase
#define W25Xxx_SERASE   0x20    // Sector Erase
#define W25Xxx_CERASE   0xC7    // Sector Erase
#define W25Xxx_READID   0x90    // Read ID /* ID: 0x00, 0xEF, (0x14, 0x15, 0x16) */
#define W25Xxx_READJEDECID 0x9F  // Read ID /* ID: 0x00, 0xEF, (0x14, 0x15, 0x16) */

#define W25XXX_MAKERID   0xEF
#define W25X16_DEVID    0x14
#define W25X32_DEVID    0x15

#define W25Xxx_PDOWN    0xB9    // Power-down
#define W25Xxx_PON      0xAB    // Release Power-down / Device ID

#define NORMAL          W25Xxx_READ
#define FAST            W25Xxx_FAST_READ
#define BLOCK           W25Xxx_BERASE
#define SECTOR          W25Xxx_SERASE
#define ALL             W25Xxx_CERASE

#define W25Xxx_SR_WIP (1 << 0) // Wait In Progress bit
#define W25Xxx_SR_WEL (1 << 1) // Wait Enable Latch bit

#define W25Xxx_ID1     0x00 // Manufacture ID
#define W25Xxx_ID2     0xEF // Memory Type
#define W25X16_ID3     0x14 // Memory Capacity
#define W25X32_ID3     0x15 // Memory Capacity
#define W25X64_ID3     0x16 // Memory Capacity

#define W25Xxx_JEDECID1 0xEF // Manufacture ID
#define W25Xxx_JEDECID2 0x30 // Memory Type
#define W25X16_JEDECID3 0x15 // Memory Capacity
#define W25X32_JEDECID3 0x16 // Memory Capacity
#define W25X64_JEDECID3 0x17 // Memory Capacity

```

```
#endif //__W25Xxx_H__
```

```
=== w25xxx_flash.c ===
```

```
#include "adStar-L.h"
#include "w25xxx_flash.h"
#include "spi.h"

int w25xxx_write_buffer (U32 addr, U8 *buffer, int length)
{
    int wlen;
    U8 wbuf[4];
    int ret = 0;
    int i;

    U32 startaddr = addr;
    U32 endaddr = addr+length-1;
    U32 startpage;
    U32 endpage;

    startpage = startaddr/W25XXX_PAGE_SIZE;
    endpage = endaddr/W25XXX_PAGE_SIZE;

    for(i=startpage;i<endpage+1;i++)
    {
        U32 offset = addr%W25XXX_PAGE_SIZE;
        if(offset + length > W25XXX_PAGE_SIZE)
        {
            wlen = W25XXX_PAGE_SIZE - offset;
        }
        else
        {
            wlen = length;
        }

        asm("clr 13");

        wbuf[0] = W25Xxx_WREN;
        spi_master_xfer(SPI_CH1,wbuf, 1, NULL, 0, 0);
        ret = w25xxx_wait_ready();
        if (ret != 0)
            break;

        wbuf[0] = W25Xxx_Page_Program;
```

```

wbuf[1] = (addr>>16);
wbuf[2] = (addr>>8);
wbuf[3] = addr;

spi_master_xfer(SPI_CH1,wbuf, 4, NULL, 0, 1); // Write Command & Address
// Continue to write data next
spi_master_xfer(SPI_CH1,buffer, wlen, NULL, 0, 0); // Write Data

/* Write Cycle Time of W25Xxx FLASH: 3.3V -> 11ms, 1.2ms */
ret = w25xxx_wait_ready();
if (ret != 0)
    break;

buffer += wlen;
addr += wlen;
length -=wlen;

wbuf[0] = W25Xxx_WRDI;
spi_master_xfer(SPI_CH1,wbuf, 1, NULL, 0, 0);
ret = w25xxx_wait_ready();
if (ret != 0)
    break;

asm("set 13");
}
return ret;
}

/*
 * mode: NORMAL, FAST
 */
int w25xxx_read_buffer (U32 addr, U8 *buffer, int length)
{
    int ret = 0;
    U8 wbuf[4];

    wbuf[0] = W25Xxx_READ;
    wbuf[1] = (addr>>16);
    wbuf[2] = (addr>>8);
    wbuf[3] = addr;
    asm("clr 13");
}

```

```
spi_master_xfer(SPI_CH1,wbuf, 4, buffer, length, 0);

ret = w25xxx_wait_ready();
asm("set 13");

return ret;
}

/*
 * SECTOR erase
 */
int w25xxx_erase (U32 addr, int length)
{
    U8 wbuf[4];
    int ret = 0;
    unsigned int esize = 0;
    unsigned int sector_size = W25XXX_SECTOR_SIZE;
    unsigned int block_size = W25XXX_BLOCK_SIZE;

    length += sector_size - 1;
    length &= ~(sector_size - 1);

    addr &= ~(sector_size - 1);

    while (length) {

        if (length >= block_size) {
            esize = block_size;
        }
        else if (length >= sector_size) {
            esize = sector_size;
        }
        }

        wbuf[0] = W25Xxx_WREN;
        spi_master_xfer(SPI_CH1, wbuf, 1, NULL, 0, 0);
        ret = w25xxx_wait_ready();
        if (ret != 0)
            break;

        if (esize == block_size) {
            wbuf[0] = W25Xxx_BERASE;
```

```
    }
    else if (esize == sector_size) {
        wbuf[0] = W25Xxx_SERASE;
    }

    wbuf[1] = (addr >> 16);
    wbuf[2] = (addr >> 8);
    wbuf[3] = addr;

    spi_master_xfer(SPI_CH1, wbuf, 4, NULL, 0, 0);

    ret = w25xxx_wait_ready();
    if (ret != 0)
        break;

    addr += esize;
    length -= esize;

    wbuf[0] = W25Xxx_WRDI;
    spi_master_xfer(SPI_CH1, wbuf, 1, NULL, 0, 0);
    ret = w25xxx_wait_ready();
    if (ret != 0)
        break;
}
return 0;
}

int w25xxx_wait_ready (void)
{
    U8 inst = W25Xxx_RDSR;
    U8 status;
    int delay = 0;
    int ret = 0;

    do {
        spi_master_xfer (SPI_CH1, &inst, 1, &status, 1, 0);
        delay++;
        if (delay > 0x80000) {
            ret = -1;
            break;
        }
    }
```

```
    } while (status & W25Xxx_SR_WIP);
    return ret;
}

int w25xxx_check_id (void)
{
    U8 buf[4];

    buf[0] = W25Xxx_READID;
    buf[1] = 0;
    buf[2] = 0;
    buf[3] = 0;

    spi_master_xfer(SPI_CH1, buf, 4, buf, 2, 0);

    w25xxx_wait_ready();

    if ((buf[0] != W25XXX_MAKERID) || (buf[1] != W25X32_DEVID)) {
        debugprintf(" 0x%x ", buf[0] );
        debugprintf("0x%x W25X32_ID3", buf[1] ); // W25X32_ID3
        return -1;
    }
    return 0;
}
```

=== main.c ===

```

#include "adStar-L.h"
#include "w25xxx_flash.h"

extern void boardinit();

void SPI_pin_init()
{
    *R_PAF4 &= ~(0xf<<12);
    *R_PAF4 |= 0x5<<12;    //spi_sck1, spi_cs1

    *R_PAF5 &= ~(0xf<<0);
    *R_PAF5 |= 0x5<<0;    //spi_miso1, spi_mosi1
}

int main()
{
    boardinit();
    uart_config(1,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);

    debugstring("=====  
WrWn");
    debugprintf("ADSTAR-L SPI_Flash.System Clock(%dMhz)WrWn",get_ahb_clock()/1000000);
    debugstring("=====  
WrWn");

    SPI_pin_init();

    int ret;

    spi_master_init(SPI_CH1);
    int baud = spi_set_freq(SPI_CH1,SPI_MASTER, W25Xxx_SPI_FREQ);
    debugprintf("SPI SCK : %d KHzWrWn", SPI_SCK(baud)/1000 );

    ret = w25xxx_check_id();
    if (ret != 0) {
        debugstring("W25Xxx ID Check FailWrWn!WrWn");
        while(1);
    }

    U8 rbuf[256];
    U8 wbuf[256];
    int i;
    for(i=0;i<256;i++)
        wbuf[i]=i;

```

```
for(i=0;i<256;i++)
    rbuf[i]=0;

w25xxx_erase(0,W25XXX_SECTOR_SIZE);
w25xxx_write_buffer(0,wbuf,256);
w25xxx_read_buffer(0,rbuf,256);
for(i=0;i<256;i++)
{
    if(wbuf[i]!=rbuf[i])
    {
        debugprintf("wbuf[%d] = %d != rbuf[%d] = %d \r\n",i,wbuf[i],i,rbuf[i]);
        debugstring("Test Error\r\n");
        while(1);
    }
    else
    {
        debugprintf("wbuf[%d] = %d == rbuf[%d] = %d \r\n",i,wbuf[i],i,rbuf[i]);
    }
}
while(1);
return 0;
}
```

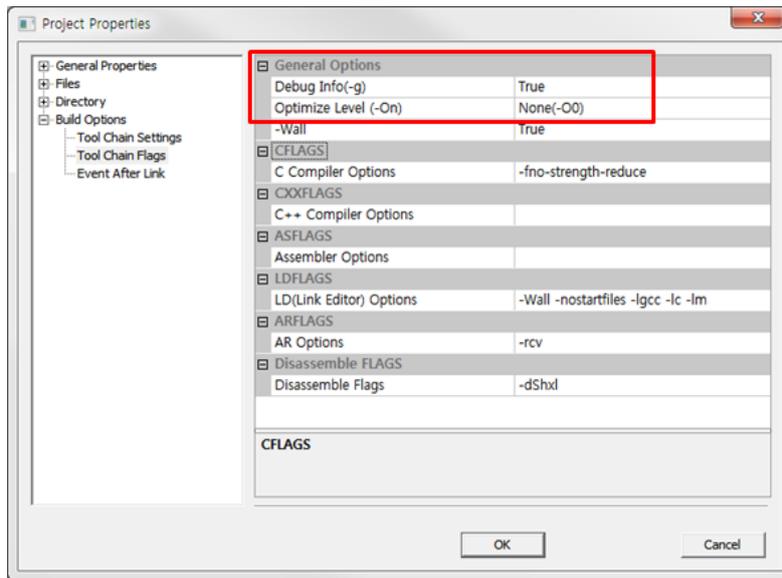
10. Debugging

이번 장에서는 debugging 에 대해서 설명한다.

Debugging 은 E-Con 과 EConMan 프로그램을 필요로 한다. E-Con 을 사용하기 위해서는 device driver 가 설치 되어야 하는데, 설치를 하지 않았다면, "1. Software 개발환경"의 device driver 설치하는 부분을 참고하기 바란다.

10.1 Debugging 준비

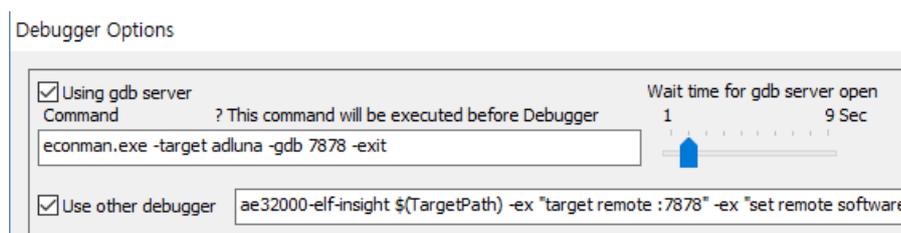
E-Con 과 EConMan 이 준비 되었다면 debugging 을 할 project 를 열어 Project → Properties 를 실행한다.



실행하면 위와 같은 창이 뜨게 되고, 빨간 상자 안의 내용, Debug Info(-g)를 TRUE 로 설정하고, Optimize Level(-On)을 None(-O0)로 설정을 한 후 OK 를 누른다. 그런 다음 Build → Rebuild Project 를 실행하여 Rebuild 를 진행한다.

Debugging 을 위해서는 Debug Info 를 TRUE 로 하여 Build 에 의해 생성되는 elf 파일에 Debug 정보를 포함시켜야 하고, Optimize Level 을 None 으로 하여 최적화 없이 compile 해야 debugging 시 보다 정확하게 source 를 debugging 할 수 있다.

Project properties 를 설정 했으면, 다음으로 Debug 메뉴의 Debug option 을 실행하여 다음과 같이 -ta 뒤에 adluna 를 입력한다.

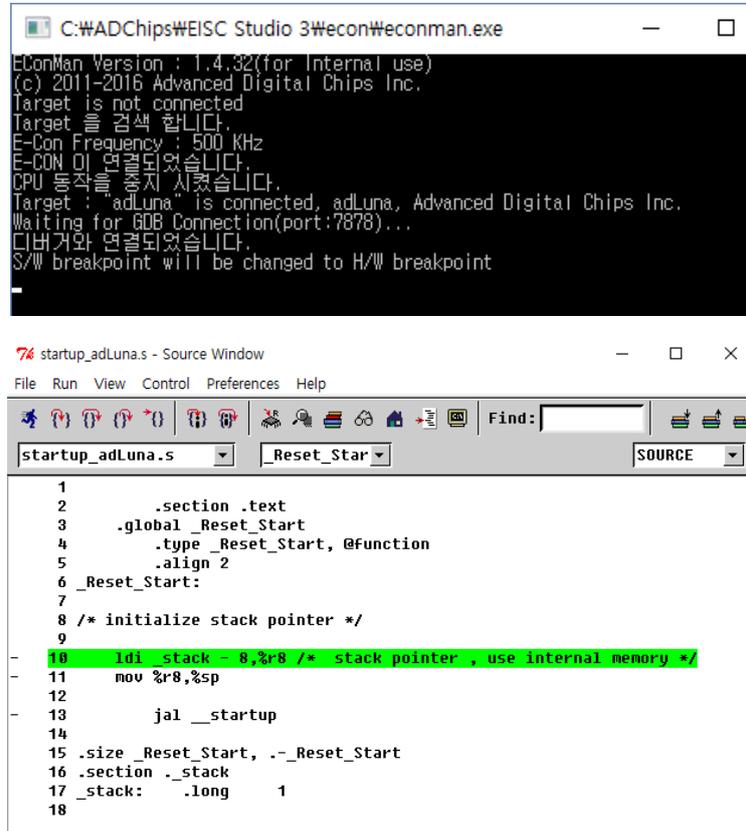


그런 다음 ok 를 누르면, Debugging 준비과정은 끝난다.

10.2 Debugging

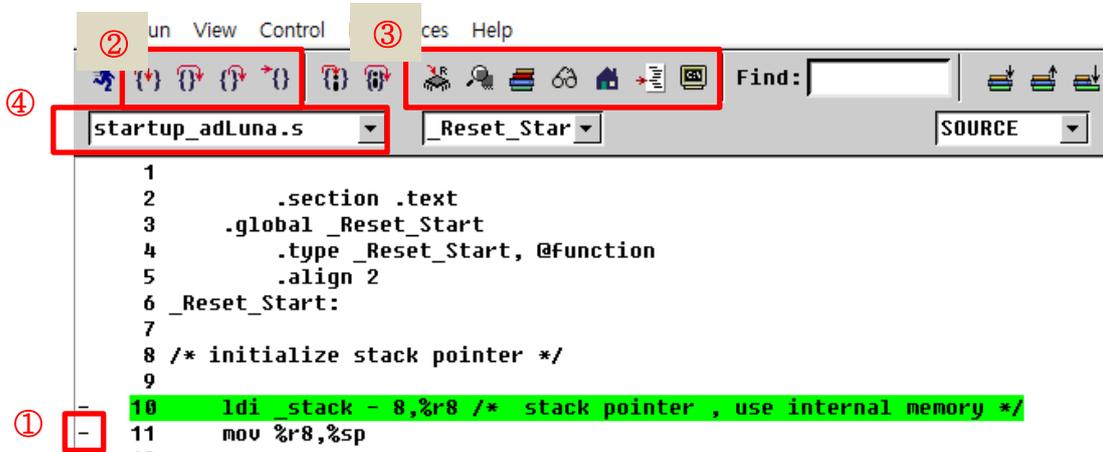
Debugging 은 다음의 순서로 진행한다.

- 1) Build 하여 생성 된 bin 파일을 0 번지에 download 한다.
- 2) Debugging 할 project 가 열려있는 상태에서 Debug → Start Debugger (F5)를 실행한다.
그러면 다음과 같이 두개의 창이 뜨면서 debugging 을 시작할 수 있다.

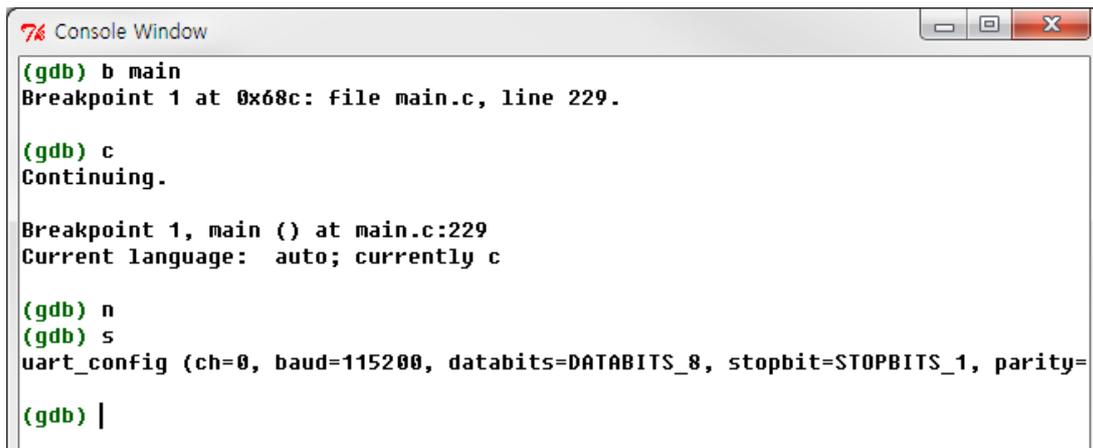


위와 같은 창이 뜨고, 녹색 바가 표시되면 정상 연결된 것으로 debugging 을 시작할 수 있다.
녹색 바가 아닐 경우 연결이 제대로 안된 것으로 다시 시도해야 한다.

- 3) Debugger 창에 대해 설명하면 다음과 같다.



- ① : 마우스로 '-' 클릭 시 break point 설정/해제
 - ② : 앞에서부터 'step', 'next', 'finish', 'continue' 동작
 - ③ : 앞에서부터 'register', 'memory', 'stack', 'watch expressions', 'local variable' 확인창 출력.
 제일 마지막 'c: '는 'console'창을 띄운다. Console 창에서는 GDB 명령을 사용하여 debugging 을 수행할 수 있다.
 - ④ : Source code 이동.
- 4) console 창에서의 debugging.



```

7% Console Window
(gdb) b main
Breakpoint 1 at 0x68c: file main.c, line 229.

(gdb) c
Continuing.

Breakpoint 1, main () at main.c:229
Current language: auto; currently c

(gdb) n
(gdb) s
uart_config (ch=0, baud=115200, databits=DATABITS_8, stopbit=STOPBITS_1, parity=
(gdb) |
  
```

- b : break point 설정.
 c : continue 명령. 다음 break point 까지 진행.
 n : next 명령. 하나의 코드 실행. (함수일 경우에도 그대로 진행)
 s : step 명령. 하나의 코드 실행. (함수인 경우 함수 내부로 들어가 진행)
 q : 종료.

11. ETC

이번 장에서는 프로그램 동작 시간과 남은 메모리(Ram size)를 확인하는 방법에 대해 설명하겠다.

11.1 여유 메모리(Ram size) 확인 방법

사용할 수 있는 메모리 size 가 얼마나 되는지, stack pointer address 와 heap 영역의 end address 의 차를 통해 대략적으로 확인이 가능하다.

get_free_mem() 함수가 stack pointer address 와 heap end address 의 차이 값을 return 해주므로, 이 함수를 사용하여 현재 사용할 수 있는 메모리 size 를 확인할 수 있다.

참고로 Return 하는 값은 stack pointer address 와 heap end address 의 차이를 통한 대략적인 값이지 정확한 값이 아니라는 것을 알고 있어야 한다.

Heap 영역 중간에 할당되었던 메모리를 free 할 경우 그 부분이 사용 가능해 지면서 사용할 수 있는 메모리 size 는 증가하지만 heap end address 가 이동하는 것은 아니기 때문에 get_free_mem 함수로 메모리 확인 시 값에 변화는 없게 된다.

Malloc 의 경우에도 할당 받을 메모리 size 는 줄겠지만, heap end address 의 변화는 발생하지 않아 get_free_mem 함수로 메모리 size 확인 시 값에 변화가 없게 된다.

```
int memsize = get_free_mem();
```

11.2 현재 동작 시간 확인

get_tick_count() 함수를 사용하여 현재 시간을 체크할 수 있다. main() 함수 호출 전 tick count 가 동작을 시작하여, 동작 중 get_tick_count() 함수를 호출하여 현재 프로그램이 동작 된 시간을 측정할 수 있다.

사용 예로, 두 지점에서의 tick count 차이로 얼마의 시간이 소요되었는지도 체크가 가능하다.
(시간 단위는 ms 이다.)

Ex)

```
int start_time = get_tick_count();
...
...
...
int end_time = get_tick_count();
int op_time = end_time - start_time;
```

11.3 파일 program 에 추가하기

Wav 파일 같은 경우에 program 에 wav data 를 추가해야 하는데, 다음의 INCBIN 매크로 함수를 사용하여 파일을 추가할 수 있다.

```
INCBIN(okwave, "01.wav"); // Add 01.wav file data,
```

위처럼 사용하면, 해당 project 와 동일 폴더에 있는 01.wav 라는 파일이 추가된다. 파일을 추가하면 다음과 같이 사용할 수 있다.

```
WAVE* pWavOK = sound_loadwavp((U8*)okwave_data, okwave_end - okwave_data);
                // okwave_data => 01.wav file data buffer
                // okwave_end - okwave_data => 01.wav file size
```

11.4 GPIO 사용법

GPIO register 에 직접 값을 설정하여 GPIO 를 제어할 수 있다.

< GPIO 출력 >

GPIO 출력은 다음의 순서로 진행된다.

1. 원하는 port 의 pin 을 출력으로 설정.

*R_GPODIR(N) register 사용. (N 은 port number)

ex) GPIO 5 port 의 6 번 pin 을 출력으로 사용. (GP5.6)

```
*R_GPODIR(5) = 1 << 6;
```

2. 해당 pin 으로 High 또는 Low 출력.

*R_GPOLOW(N), *R_GPOHIGH(N) 사용. (N 은 port number)

ex) GPIO 5 port 의 6 번 pin 을 High, Low 로 출력. (GP5.6)

```
*R_GPOLOW(5) = 1 << 6; // GP5.6 Low 출력
```

```
*R_GPOHIGH(5) = 1 << 6; // GP5.6 High 출력
```

해당 register 에 0 을 write 하면 아무 변화가 없으며, GPIO 출력 level 은 가장 나중에 수행한 것이 적용된다.

```
ex) *R_GPOLOW(5) = 1 << 6; // GP5.6 Low 출력
```

```
*R_GPOHIGH(5) = 1 << 6; // GP5.6 High 출력
```

```
*R_GPOLOW(5) = 1 << 6; // GP5.6 Low 출력
```

// 여기까지 수행하면 GP5.6 은 Low 출력 상태이다.

< GPIO 입력 >

GPIO 입력은 다음의 순서로 진행된다.

1. 원하는 port 의 pin 을 입력으로 설정.

*R_GPIDIR(N) register 사용. (N 은 port number)

ex) GPIO 6 port 의 2 번 pin 을 입력으로 사용. (GP6.2)

```
*R_GPIDIR(6) = 1 << 2;
```

2. 해당 pin 의 level 을 read 함.

*R_GPILEV(N) register 를 사용. (N 은 port number)

ex) GPIO 6 port 의 2 번 pin 의 level 을 read.

U8 input = (*R_GPILEV(6) >> 2) & 1; // input 값이 0 이면, GP6.2 Low, input 값이 1 이면 GP6.2 High.