



E-Con

Version 0.9.1

Advanced Digital Chips Inc.

History

Version 0.9.1 : 2010-06-24

GDB 와 연결하여 원격디버깅 내용 추가됨

Version 0.9 : 2010-06-18

First Release.

E-CON 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

8th Floor, KookMin 1 Bldg., 1009-5, Daechi-Dong, Gangnam-Gu, Seoul, 135-280, Korea.

Tel: +82-2-2107-5800 Fax: +82-2-571-4890

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

목 차

1. E-Con	6
2. ECONMAN.EXE 전체 명령어 요약	7
3. 명령어 상세 설명	9
target TARGET_NAME(option)	10
help COMMAND(option)	10
q	11
exit	11
readb ADDRESS SIZE	11
reads ADDRESS SIZE	11
readw ADDRESS SIZE	11
writeb ADDRESS BYTE-VALUE	11
writes ADDRESS 2BYTE-VALUE	12
writew ADDRESS 4BYTE-VALUE	12
fileread ADDRESS SIZE FILE_NAME	12
filewrite ADDRESS FILE_NAME	12
flash_init	12
flash_eraseall	13
flash_erase START_SECTOR SECTOR_COUNT	13
flash_filewrite ADDRESS FILENAME	13
flash_fileread ADDRESS SIZE FILENAME	13
proc_readb ADDRESS SIZE	13

proc_reads ADDRESS SIZE	14
proc_readw ADDRESS SIZE	14
proc_writeb ADDRESS BYTE-VALUE	14
proc_writes ADDRESS SIZE 2BYTE-VALUE	14
proc_writew ADDRESS SIZE 4BYTE-VALUE	14
proc_stop	14
proc_resume	14
proc_read_all_regs	15
proc_read_reg REGNUM	15
runat ADDRESS	15
runscript SCRIPT_FILENAME	16
version	16
targetlist	16
memtest ADDRESS SIZE	16
reset	16
gdbserver PORT_NUMBER	16
4. 프로그램 실행 명령어 옵션	17
5. EISC Studio 에서 Flash write 기능을 사용하기 위한 설정	18
6. SystemInit 에서 실행되는 초기화	18
6.1 Dummy	18
6.2 Cantus	19
6.3 EN773	19

6.4	KLDW	19
7.	E-Con 과 GDB 를 이용한 프로그램 디버깅방법.....	19

1. E-CON

E-CON™은 EISC JTAG Debugger 를 제어하기 위한 장비의 이름이다.

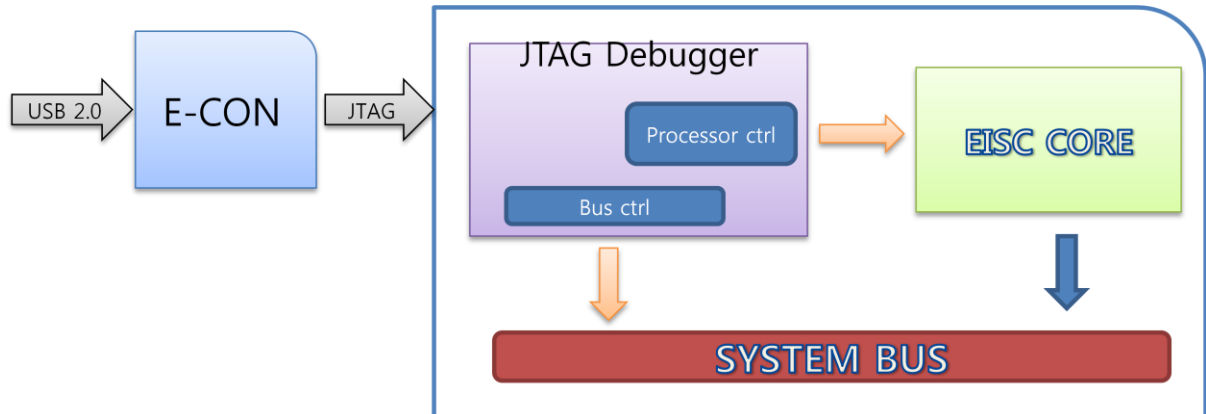


그림 1 E-CON 과 JTAG Debugger

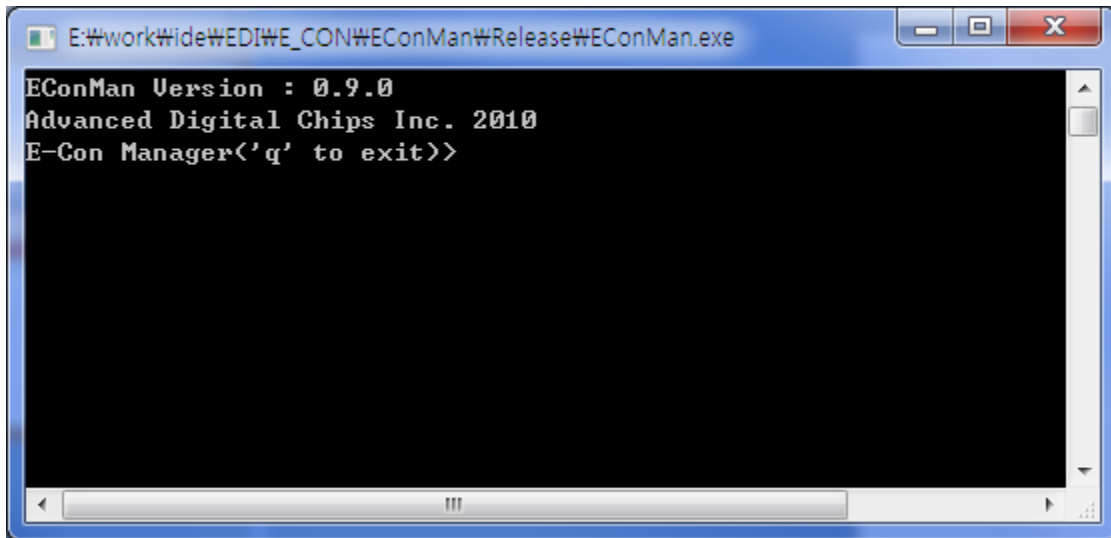
EISC JTAG Debugger 는 두 가지 모델로 나누어 질 수 있다. EISC CORE 를 제어 하는 부분과 시스템버스 를 제어하는 부분으로 나누어 진다.

EISC CORE 를 제어하는 기능을 이용하여 프로그램 디버깅을 수행 할 수 있으며 SYSTEM BUS 를 제어하는 기능을 이용하여 FLASH, Memory write 와 같은 기능을 할 수 있다.

E-CON 을 제어하는 프로그램은 "EConMan.exe" 라는 별도의 프로그램을 제공한다.

EConMan 은 Microsoft Windows 환경에서만 구동된다.

EConMan 을 실행 하면 아래 그림과 같은 화면을 볼 수 있다.



이 상태에서 사용자가 필요한 명령어를 실행 하면 된다.

2. ECONMAN.EXE 전체 명령어 요약

아래 목록은 EConMan 에서 사용할 수 있는 전체 명령어이다.

“help”명령어를 실행하여 전체 명령어 목록을 확인 할 수 있다.

Command (full, short)	Argument 1	Argument 2	Argument 3	Description
target ,ta	TARGET_NAME (option)	X	X	Connect E-Con and Target System. If target name not defined, it will search target.
help ,h	Command(option)	X	X	Show help message.
q	X	X	X	Exit
exit	X	X	X	Equal to "Exit"
readb ,rb	ADDRESS	SIZE(option)	X	Read SIZE*1byte. Default size is 1.
reads ,rs	ADDRESS	SIZE(option)	X	Read SIZE*2byte Default size is 1.
readw ,rw	ADDRESS	SIZE(option)	X	Read SIZE*4byte.

				Default size is 1.
writeb ,wb	ADDRESS	BYTE VALUE	X	Write DATA(1byte).
writes ,ws	ADDRESS	2BYTE-VALUE	X	Write DATA(2byte).
writew ,ww	ADDRESS	4BYTE-VALUE	X	Write DATA(4byte).
filewrite ,fw	ADDRESS	FILENAME		Read File(HOST PC) and Write it to Target Memory
fileread ,fr	ADDRESS	SIZE	SAVE FILENAME	Read Target Memory and Save as FILENAME(HOST PC)
flash_init	X	X	X	Check and initialize Target Flash Memory controller and information Indeed, you do not need this command. Flash_xxx command always call this function.
flash_filewrite ,ffw	ADDRESS	FILENAME		Read File(HOST PC) and erase sector and Write it to Target Flash Memory.
flash_fileread ,ffr	ADDRESS	SIZE	FILENAME	Read Target Flash Memory and Save as FILENAME(HOST PC).
flash_eraseall ,ffea	X	X	X	Erase Flash All Sector.
flash_erase ,ffe	START SECTOR	SECTOR COUNT	X	Erase sectors.
proc_readb ,prb	ADDRESS	BYTE SIZE	X	Read Target Memory
proc_reads ,prs	ADDRESS	2BYTE-SIZE	X	Read Target Memory
proc_readw ,prw	ADDRESS	4BYTE-SIZE	X	Read Target Memory
proc_writeb ,pwb	ADDRESS	BYTE-VALUE	X	Write Data to Target Memory
proc_writes ,pws	ADDRESS	2BYTE-VALUE	X	Write Data to Target Memory
proc_writew ,pww	ADDRESS	4BYTE-VALUE	X	Write Data to Target Memory
proc_stop ,pst	X	X	X	Stop Processor of Target

proc_resume ,pre	X	X	X	Resume Processor of Target
proc_read_all_regs ,prar	X	X	X	Print All registers of CPU.
proc_read_reg ,pr	REG_NUM	X	X	Print register of CPU
runat ,ra	ADDRESS	X	X	Run program at Address
runscript ,rs	SCRIPT FILENAME	X	X	Read Command List File and Run commands.(Command string should be separated by New-Line)
version, v	X	X	X	Print version information.
targetlist ,tali	X	X	X	Print supported Target Name.
reset ,res	X	X	X	Reset Target System
systeminit	X	X	X	Run pre-defined system initialize
memtest	ADDRESS	SIZE		Memory Read/Write Testing using simple algorithm.
gdbserver	portnumber	X	X	Run GDB Server

3. 명령어 상세 설명

모든 명령어는 대소문자 구분을 하지 않는다.

접미사는 단위를 의미 한다

-b : byte

-s : 2byte(short)

-w : 4byte(word)

모든 Read 나 Write 명령어의 경우 ADDRESS 가 그 data 형의 경계에 존재 해야 된다.

즉 0x1 번지에 2byte 나 4byte 의 Read/Write 명령어의 경우 잘못된 값을 읽거나 쓸 수 있다.

-b 명령어는 어떤 번지도 상관 없다.

-s 명령어의 경우 최하위 번지가 2 의 배수여야만 한다.

-w 명령의 경우 최하위 번지가 4 의 배수여야만 한다.

특정 번지에서 읽은 값 출력 형식은 16 진수로 표현된다.

0 1 2 3 4 5 6 7 8 9 A B C D E F
ADDRESS(16byte 단위) :

proc_ 이라는 접두어가 붙지 않는 read/write 함수의 경우 시스템 BUS 를 통해서 그 역할을 수행한다. 즉 Cache memory 나 SPM 의 접근이 없다. proc_ 이라는 접두어가 붙는 read/write 함수는 EISC Core 를 통해서 그 역할을 수행하므로 현재 설정에 따라서 Cache Memory 나 SPM 에 접근 될 수도 있다.

TARGET TARGET_NAME(OPTION)

ECON 과 연결한다.

TARGET_NAME 을 연결 한다.

JTAG BUS 모드로 진입한다.

모든 명령어 이전에 반드시 실행되어야 할 명령어이다.

TARGET_NAME 이 지정 하지 있지 않을 경우 현재 프로그램에서 지원되는 모든 Device 를 호출 하여 device 를 찾는다. 이 경우 TARGET_NAME 이 지정 될 경우 보다 다소 느려질 수 있다.

HELP COMMAND(OPTION)

도움말을 출력한다.

COMMAND 가 지정될 경우 특정 도움말만 출력 하지만 그렇지 않을 경우 모든 command 에 대한 도움말을 출력한다.

Q

프로그램을 종료한다.

"exit" 와 동일하다.

EXIT

프로그램을 종료한다.

"q" 와 동일하다.

READB ADDRESS SIZE

특정 주소에서 1byte 단위로 SIZE 만큼 읽어서 그 값을 출력한다.

READS ADDRESS SIZE

특정 주소에서 2byte 단위로 SIZE 만큼 읽어서 그 값을 출력한다.

ADDRESS 는 반드시 2 의 배수여야 한다.

READW ADDRESS SIZE

특정 주소에서 4byte 단위로 SIZE 만큼 읽어서 그 값을 출력한다.

ADDRESS 는 반드시 4 의 배수여야 한다.

WRITEB ADDRESS BYTE-VALUE

특정 주소에 BYTE-VALUE 를 write 한다.

"writeb 0x123 0x12"

WRITES ADDRESS 2BYTE-VALUE

특정 주소에 BYTE-VALUE 를 write 한다.

"writeb 0x122 0x1234"

WRITEW ADDRESS 4BYTE-VALUE

특정 주소에 BYTE-VALUE 를 write 한다.

"writeb 0x124 0x12345678"

FILEREAD ADDRESS SIZE FILE_NAME

특정 주소에 SIZE*BYTE 만큼 읽어서 host pc 에 FILE_NAME 이라는 이름으로 저장한다.

"fileread 0 0x100 dump.bin"

FILEWRITE ADDRESS FILE_NAME

host pc 의 FILE_NAME 이라는 파일을 읽어서 target memory 에 저장한다.

FAT 와 같은 파일시스템의 형태로 저장 하는 것이 아니라 특정 번지에 저장한다.

"fileread 0 dump.bin"

FLASH_INIT

Flash Memory Controller 를 초기화한다.

장착된 Flash memory 정보를 수집 및 출력한다.

flash_xxx 관련 함수들이 항상 이 함수를 먼저 호출 하게 되므로 이 명령어를 반드시 실행할 필요는 없다.

FLASH_ERASEALL

Flash Memory 전체를 Erase 한다.

FLASH_ERASE START_SECTOR SECTOR_COUNT

START_SECTOR 부터 SECTOR_COUNT 만큼 Erase 한다.

FLASH_FILEWRITE ADDRESS FILENAME

Host PC 의 FILENAME 을 읽어서 Target Flash Address 에서 부터 저장 한다.

이 함수는 필요한 sector size 만큼 erase 한 후 file data 를 write 하므로 별도의 erase 함수를 호출 할 필요가 없다.

file data 가 write 되지 않는 sector 의 안의 기존 data 는 지워진다.

FLASH_FILEREAD ADDRESS SIZE FILENAME

Target Flash memory 의 특정 번지에서 SIZE byte 만큼 읽어서 Host PC 의 FILENAME 이라는 파일로 저장한다.

PROC_READB ADDRESS SIZE

내부 EISC Core 를 이용하여 특정 주소에서 1byte*SIZE 만큼 읽어서 출력한다.

PROC_READS ADDRESS SIZE

내부 EISC Core 를 이용하여 특정 주소에서 2byte*SIZE 만큼 읽어서 출력한다.

PROC_READW ADDRESS SIZE

내부 EISC Core 를 이용하여 특정 주소에서 4byte*SIZE 만큼 읽어서 출력한다.

PROC_WRITEB ADDRESS BYTE-VALUE

내부 EISC Core 를 이용하여 특정 주소에 BYTE-VALUE 를 기록한다.

PROC_WRITES ADDRESS SIZE 2BYTE-VALUE

내부 EISC Core 를 이용하여 특정 주소에 2BYTE-VALUE 를 기록한다.

PROC_WRITEW ADDRESS SIZE 4BYTE-VALUE

내부 EISC Core 를 이용하여 특정 주소에 4BYTE-VALUE 를 기록한다.

PROC_STOP

Target system 의 EISC Core 강제로 멈춘다.

PROC_RESUME

정지된 EISC Core 를 재 실행 시킨다.

PROC_READ_ALL_REGS

CPU 내의 모든 register 값을 출력한다.

PROC_READ_REG REGNUM

CPU 내의 특정 register 값을 출력한다.

Register Number

REG_GPR0=0,

REG_GPR1=1

.....

REG_GPR15=15

REG_CR0=16

REG_CR1,

REG_ML,

REG_MH,

REG_ER,

REG_LR,

REG_PC=22

REG_SR,

REG_SSP,

REG_ISP,

REG_USP=26

RUNAT ADDRESS

프로그램카운터 레지스터를 ADDRESS 로 설정 한 다음 프로세서를 재 실행 시킨다.

즉 해당 어드레스에서 프로그램을 실행 시킬 수 있다.

RUNSCRIPT SCRIPT_FILENAME

명령어로 이루어진 text 파일을 읽어서 그 명령어를 순차적으로 실행한다.

명령어가 정상적으로 실행되지 않았을 경우 그 이후 명령어는 실행 하지 않는다.

VERSION

프로그램 버전 정보 및 간략한 update 정보를 출력한다.

TARGETLIST

현재 지원되는 모든 Target 를 출력한다.

MEMTEST ADDRESS SIZE

내부 알고리즘을 이용하여 메모리 Read/Write 테스트를 진행한다.

RESET

아래 두 가지 reset 제어신호를 보낸다.

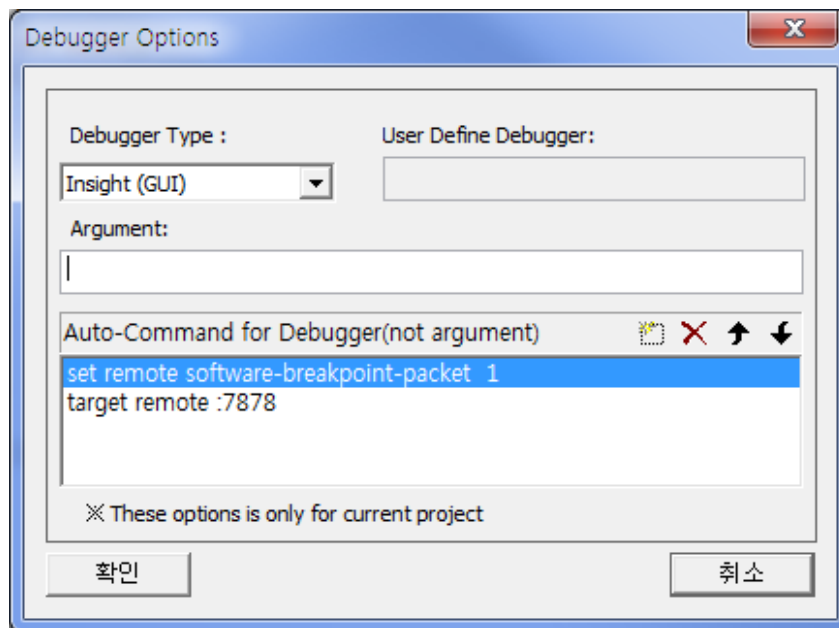
1. E-CON 의 Jtag Cable 을 통해서 reset 신호를 보낸다. (이 경우 Target system 의 reset pin 과 연결되어 있어야 한다)
2. EISC JTAG Debugger 를 통해서 reset 신호를 보낸다.

GDBSERVER PORT_NUMBER

GDB 와 통신채널로 port number 를 설정 한 후 GDB 의 Remote debug server 역할을 수행한다.

econman 을 이용하여 remote debugging 을 위한 수행 방법

1. "gdbserver 7878" 실행 , 정상적으로 실행 될 경우 더 이상 사용자 명령어를 받아들이는 prompt 가 뜨지 않는다.
2. EISC gdb 를 실행한다.
 - A. EISC Studio3 를 이용 할 경우 debug>debug option 에 아래와 같이 설정 한 후 start debugger 를 실행한다.



4. 프로그램 실행 명령어 옵션

econman.exe 실행 할 때 연속적으로 실행 될 명령어를 입력 할 수 있다.

econman.exe -command1 arg1 arg2 arg3 -command2 arg1

와 같이 명령어 앞에 '-' 만 붙이면 된다.

이 경우 순차적으로 명령어를 실행 하게 되면 특정 명령어의 실행이 올바르게 없을 경우 그 이후 명령어는 실행 하지 않는다.

ex) econman.exe -target cantus -systeminit -flash_filewrite 0 bootloader.bin -exit

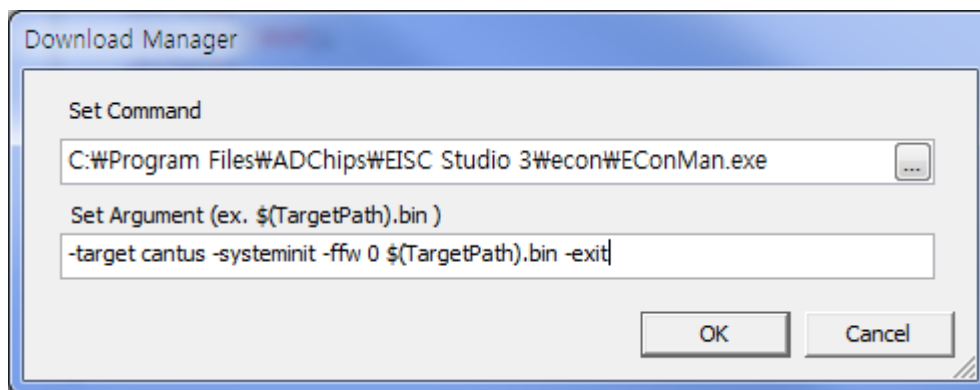
1. CANTUS Target board 를 연결
2. 미리 설정된 시스템 초기화 루틴 실행

3. 0 번에 bootloader.bin 파일을 다운로드
4. 종료

5. EISC STUDIO 에서 FLASH WRITE 기능을 사용하기 위한 설정

EISC Studio 버전 3.1 이상 버전의 경우 build>download to target, download option 기능이 있다.

이 기능을 이용하여 econman.exe 를 실행 하여 target board 에 binary file 을 다운로드 할 수 있다.



download option 을 실행하여 위 그림과 같이 설정한 이후 download 를 실행 시키면 EConMan.exe 를 이용하여 Target 에 다운로드 할 수 있다.

6. SYSTEMINIT 에서 실행되는 초기화

EConMan.exe 실행파일은 특정 Target 마다 미리 정해진 초기화 기능들이 내포되어 있다.

미리 정해진 초기화 기능은 다음과 같다.

(Flash 가 내장되어 있는 경우 항상 마지막에 Flash 정보를 읽는다.)

6.1 DUMMY

Dummy 를 Target 을 설정 할 경우 어떤 코드도 실행되지 않는다.

6.2 CANTUS

1. 0x2040 으로 PLL 설정(XIN 이 11.2896Mhz 일 경우 96Mhz PLL)
2. 0x3300 으로 Flash Control Register 설정
3. Flash 정보를 읽는다.

6.3 EN773

1. Serial Flash Controller 를 초기화 한다.(Quad Mode, 1Clock)
2. Serial Flash 정보를 읽는다.

6.4 KLDW

1. Serial Flash Controller 를 초기화 한다.(Quad Mode, 1Clock)
2. Serial Flash 정보를 읽는다.

7. E-CON 과 GDB 를 이용한 프로그램 디버깅방법

E-Con 을 이용하여 Target Board 와 GDB 를 연결하여 프로그램을 디버깅 할 수 있다.

GDB 를 통해서 디버깅을 하기 위해서는 반드시 프로세서를 멈춰야 한다. 프로그램 디버깅을 어느 시점에서부터 할 것인가에 따라서 두 가지 형태로 나눌 수 있다.

첫 번째는 프로그램을 최초 부팅 시부터 디버깅을 하는 것이고 다른 하나는 현재 동작중인 프로그램을 세워서 그 상태에서부터 디버깅 하는 것이다.

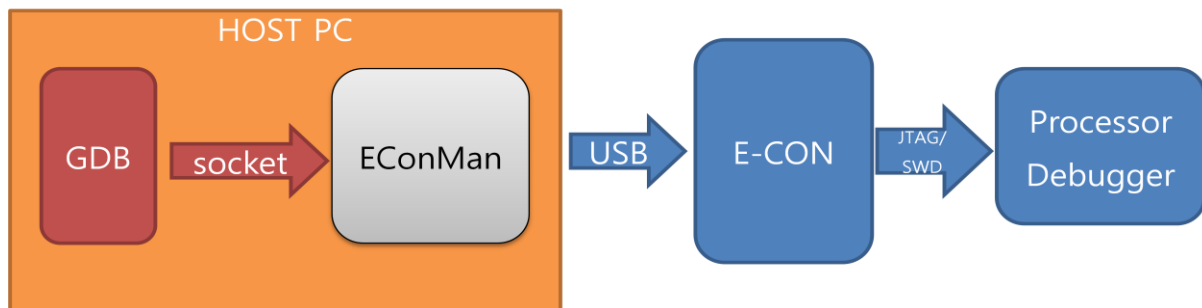
첫 번째 방법으로 디버깅 하기 위해선 Target Board 를 JTAG Debug Mode 로 부팅하여야 한다. JTAG Debug Mode 로 부팅 하게 되면 프로세서는 Reset vector 를 읽어온 상태에서 멈추게 된다. 따라서 최초 부팅 시부터 프로그램 디버깅을 할 수 있게 된다.

E-Con 과 JTAG Debug Mode PIN 이 연결되어 있다면 E-CON 이 제어 하므로 별도의 외부 작업은 필요 없다. 하지만 연결되어 있지 않다면 사용자가 해당 PIN 설정을 해야만 한다. 그렇지 않을 경우 Reset 시에 프로그램이 실행 되기 때문에 최초 부팅 시부터 디버깅을 할 수 없다.

두 번째 방법인 현재 동작중인 프로그램을 디버깅 하는 방법은 EConMan 을 실행하여 "Target TARGET_NAME" 명령어를 실행하면 프로세서를 멈추게 된다.

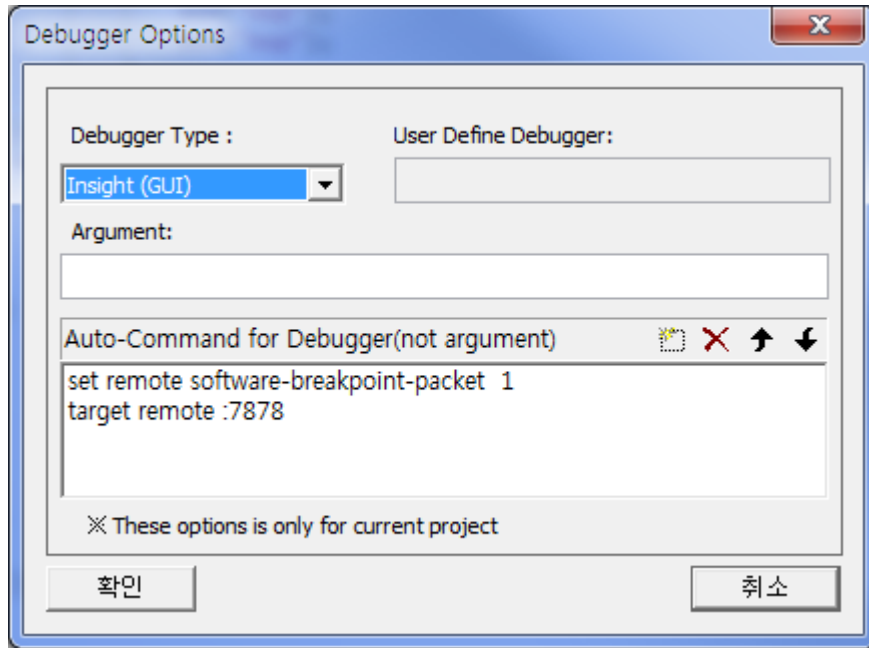
이 상태에서 "gdbserver PORT" 명령어를 실행하고 EISC Studio 3 에서 Start debugger 를 실행 하면 현재 멈춰진 상태에서부터 디버깅을 시작 할 수 있다.

EConMan 과 GDB 의 연결은 socket 통신으로 연결된다.



따라서 GDB 를 실행 할 때 target 을 socket 으로 연결 하여야 한다.

"target remote localhost:7878" 이란 명령어가 그것이다. 7878 은 포트넘버이다. 일반적으로 1024 보다 낮은 포트는 시스템에서 예약되어 사용되는 경우가 많으므로 그 이상의 넘버를 지정하는 것 이 안전하다. "localhost" 는 생략 가능 하다.



따라서 EISC Studio 의 debug option 에서 위의 그림과 같이 설정 한 후 start debugger 를 실행하면 EConMan 과 socket 7878 을 이용하여 연결된다.

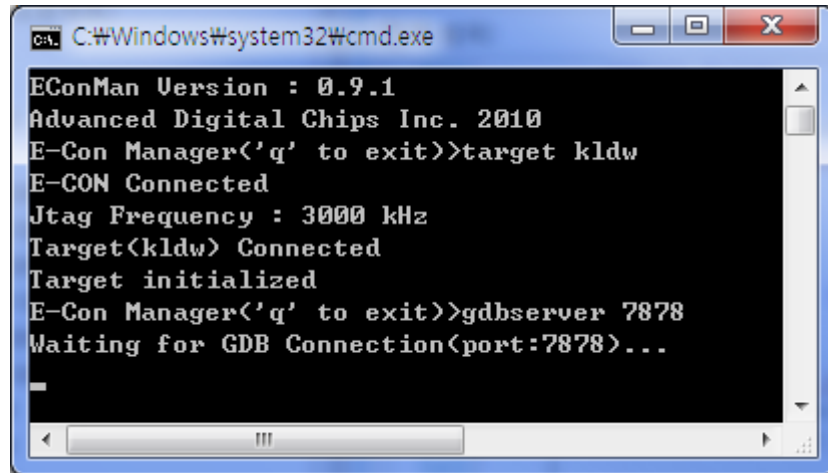
현재 target 이 연결된 상태라면 target 에서 실행이 멈춰진 소스라인을 자동으로 보여준다.

target 에 이미 프로그램이 실행된 상태이기 때문에 “run” 명령어를 사용해선 안된다.

이 상태에서 디버깅을 시작 하면 된다.

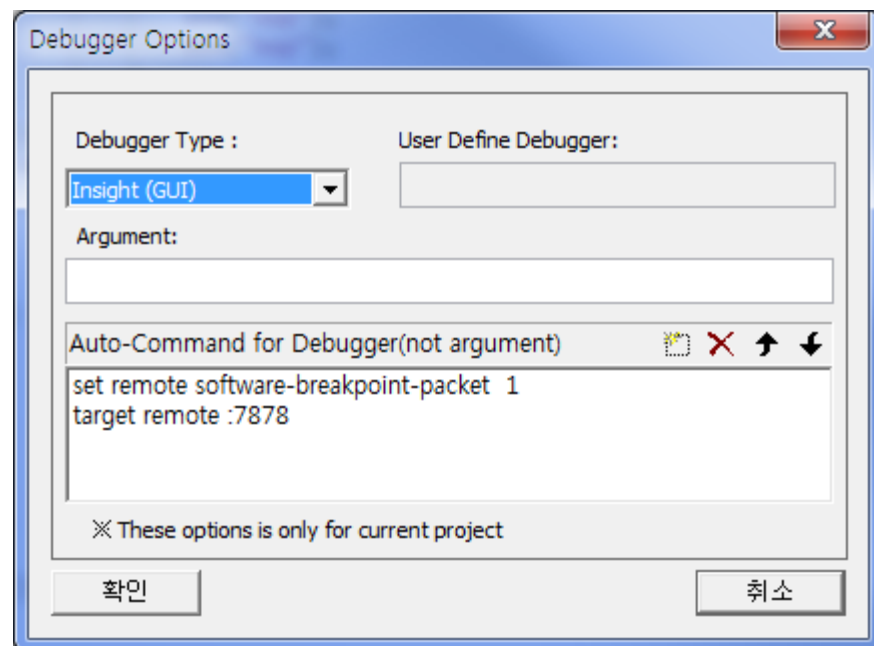
전체 연결 순서 (디버깅 할 프로그램은 이미 다운로드 된 상태를 전제로 한다)

1. Target Board 를 JTAG Debug mode 로 설정 한 후 Reset 시킨다.
2. EConMan 에서 “Target”명령어를 실행하여 Board 을 연결 한다.
 - A. ECON 과 JTAG Debug PIN 이 연결되어 있고 외부 JTAG Debug mode 설정이 없다면 “reset” 명령어를 실행한다.
3. “gdbserver 7878” 로 gdb 와 연결한다.



```
C:\Windows\system32\cmd.exe
EConMan Version : 0.9.1
Advanced Digital Chips Inc. 2010
E-Con Manager<'q' to exit>>target kldw
E-CON Connected
Jtag Frequency : 3000 kHz
Target(kldw) Connected
Target initialized
E-Con Manager<'q' to exit>>gdbserver 7878
Waiting for GDB Connection(port:7878)...
```

4. EISC Studio 3 에서 debug option 을 아래 그림과 같이 설정 한다.
- A. 이 설정은 저장되므로 모든 프로젝트에 동일하게 적용된다. 따라서 실행 시 마다 설정을 할 필요는 없다.



5. start debugger 를 실행한다.

```

1
2 .global __Reset_Start
3 __Reset_Start:
4 .section .text
5 ldi _stack, %R7
6 mov %r7, %sp
7
8 ldi 0xF0000800, %R7 #MOD
9 ldi 0x00000102, %R6
10 st %r6, (%r7, 0)
11
12 ldi 0xF0000804, %R7 #BRT
13 ldi 0x00000000, %R6
14 st %r6, (%r7, 0)
15
16 #ldi 0xF0000810, %R7 #QUAD
17 #ldi 0x00000038, %R6
18 #st %r6, (%r7, 0)
19
20 # Initialize data and bss section
21 .extern __rom_data_start
22 .extern __ram_data_start
23 .extern __ram_data_end
24 .extern __bss_start
25 .extern __bss_end
26 __copy_data:
27 ldi __rom_data_start, %r0
28 ldi __ram_data_start, %r1
29

```

Program not running. Click on run icon to start.

d8 5

Reset 상태에서 멈춰진 상태를 볼 수 있다.

6. H/W breakpoint 를 설정 한다
 - A. H/W Breakpoint 명령어는 "hbreak", "hb" 이다. console window 에서 이 명령어를 이용하여 특정 라인 또는 특정 함수에 breakpoint 를 설정 할 수 있다.
 - i. "hb main" : main 함수에 breakpoint 를 설정
 - ii. "hb main.c:123" : main.c 의 123 라인에 breakpoint 설정
7. "continue" 를 실행하면 breakpoint 가 발생 되는 시점 까지 기다리게 된다.
 - A. "run" 를 실행하며 안된다.

좀 더 자세한 GDB 명령어는 아래 링크 주소에서 얻을 수 있다.

<http://sourceware.org/gdb/>