E-Con User's Guide

Version 1.4.25

Advanced Digital Chips Inc.

변경사항

버전	변경내용
1.4.25	● EConMan 1.4.25 내용 추가
1.4.10	● 이 문서와 버전과 EConMan.exe 의 버전 일치화
1.0.1	● 오타 수정 및 명령어 입력 예 추가 ● E-Con 과 GDB 를 이용한 프로그램 디버깅 방법 내용 수정
1.0.0	● PIN 배치 그림 추가
0.x.x	● First release

Office

경기도 안양시 동안구 학의로 282, A동 22층

Tel: +82-31-463-7500

Fax: +82-31-463-7588

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

목차

1. Introduction

E-Con™ 은 USB 를 통해서 개발자의 개발 PC 와 Target Hardware 의 JTAG/SWD(serial wired debugger) 을 연결 해 주는 장비의 이름이다.

- 프로그램 다운로드 – memory of target (flash, nand, sram, sdram, ddr, etc,…)
- 프로그램 디버깅

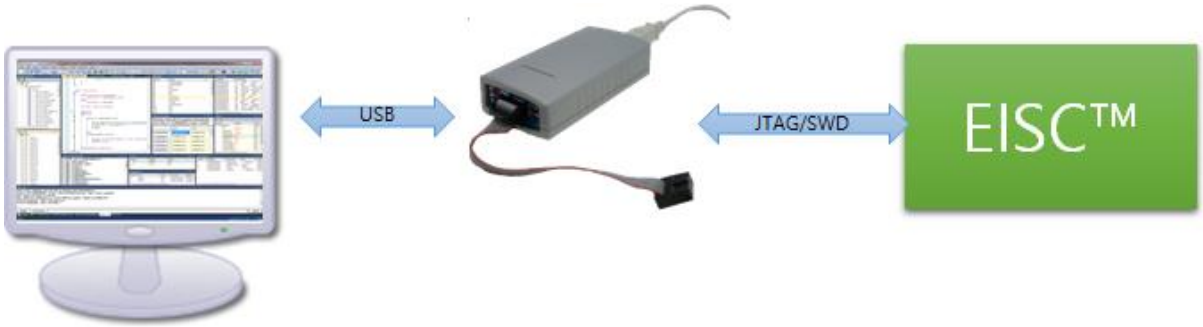


그림 1 E-Con 연결도

EISC H/W debugger 는 크게 두 가지 모델로 나뉘어 진다.

- Core Debugger
- Bus Debugger

Core debugger 는 EISC Core 를 제어 하는 방법이며 일반적으로 프로그램 디버깅시에 사용된다. Bus Debugger 는 EISC Core 를 통하지 않고 JTAG/SWD 를 통해서 직접 System bus 를 제어 할 수 있다.

E-Con 을 통해서 Target Hardware 를 제어하는 PC 용 application 은 "EConMan.exe" 라는 별도의 프로그램이 제공된다.

2. EConMan

"EConMan.exe" 는 E-Con 을 통해서 Target Hardware 를 제어하는 Microsoft Windows 용 Command line interface(CLI) 를 갖는 application 이다.

EISC Studio 3.x 를 설치하면 EConMan.exe 가 같이 설치 된다.

설치 경로를 default 로 설정 했다면 "C:₩ADChips₩EISC Studio 3₩econ" 에 위치 하게 된다.

- Support platform
  - Windows XP sp3 이상, Windows vista, Windows 7, Windows 8, Windows 8.1, Windows 10

EConMan.exe 를 실행 하면 아래와 같은 화면을 볼 수 있다.

그림 2 EConMan.exe 실행화면

이 상태에서 명령어를 입력 하거나 "EConMan.exe" 의 실행 인자로 명령어를 실행 할 수도 있다.

실행 인자로 명령어를 입력 할 때는 하나의 명령어 앞에 '-' 를 입력 하면 된다.

예를 들어서 target device 를 연결 하기 위한 명령어 "target " 를 실행 인자로 입력 할 경우

"EConMan.exe –target adStar "

와 같이 실행 하면 된다.

명령어의 개수는 제한이 없지만 입력 순서대로 실행 되며 명령어를 실행을 실패 할 경우 그 이후의 명령어는 더 이상 실행 하지 않고 사용자 입력 대기 상태로 전환된다.

2.1. 명령어요약

아래 리스트는 EConMan 에서 사용 할 수 있는 전체 명령어이다.

"help" 명령어를 실행하여 전체 명령어를 확인 할 수도 있다.

**EConMan 의 명령어 중 일부는 Processor Type 에 따라 사용할 수 없을 수도 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	exit
errorexit	X	X	X	Exit if previous command is error.
sleepms ,slm	ms sec	X	X	The time interval for which execution is to be suspended, in milliseconds
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.
systeminit ,sysinit	X	X	X	Run pre-defined system initialization.
systeminitcfg,sysinitcfg	file name	x	x	Run "econ/config/file name" instead of pre-

				defined system initialization.
memtest ,mtest	ADDRESS	SIZE	X	Memory Read/Write Testing using simple algorithm.
jtagspeed ,jtags	N	X	X	Set E-Con clock speed. JTAG Clock = 30Mhz/(N+1) SWD Clock = 12Mhz/(N+1)
swdspeed,swds	N	X	X	SWD Clock = 12Mhz/(N+1)
gdbserver ,gdb	portnumber	X	X	Run GDB Server
set_def	Number	Value(32bit)	/x	Set a specified value to a User Define Register. Now, supports only 0 Register for debugger
get_def	Number	X	X	Set a specified value to a User Define Register.
reset, res	X	X	X	Reset Target System
debugmodereset, dreset	X	X	X	set debugmode(set_def) and reset.
Bus Debugger Command				
readb, rb	ADDRESS	1BYTE-SIZE	X	Read SIZE*1byte.
reads, rs	ADDRESS	2BYTE-SIZE	X	Read SIZE*2byte
readw, rw	ADDRESS	4BYTE-SIZE	X	Read SIZE*4byte.
writeb, wb	ADDRESS	1BYTE-VALUE	X	Write 1Byte DATA.
writes, ws	ADDRESS	2BYTE-VALUE	X	Write 2Byte DATA.
writew, ww	ADDRESS	4BYTE-VALUE	X	Write 4Byte DATA.
writetest	ADDRESS	SIZE	iteration count(option)	Write, Read and compare

filewrite ,fw	ADDRESS	FILENAME	readyrun, noverify (option)	Read File(HOST PC) and Write it to Target Memory
fileread ,fr	ADDRESS	SIZE	SAVE FILENAME	Reads data from 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	readyrun, noverify (option)	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 ,fea	X	X	X	Erase Flash All Sector.
flash_erase ,fe	START SECTOR	SECTOR COUNT	X	Erase sectors.
flash_memset,fmemset	ADDRESS	VALUE(1byte)	SIZE	Fill memory with a constant byte.
flash_writetest	ADDRESS	SIZE	X	write, read and compare
nand_init	X	X	X	Check and initialize Target NAND Flash Memory controller and information Indeed, you do not need this command. nand_xxx command always call this function
nand_filewrite ,nfw	ADDRESS	FILENAME	X	Read File(HOST PC) and erase sector and Write it to Target NAND Flash Memory.

nand_fileread ,nfr	ADDRESS	SIZE	FILENAME	Read Target NAND Flash Memory and Save as FILENAME(HOST PC).
nand_eraseall ,nea	X	X	X	Erase Flash All Block.
nand_erase ,ne	START BLOCK	BLOCK COUNT	X	Erase blocks.
Core Debugger Command				
proc_readb ,prb	ADDRESS	1BYTE-SIZE	X	Reads data by Core debugger
proc_reads ,prs	ADDRESS	2BYTE-SIZE	X	Reads data by Core debugger
proc_readw ,prw	ADDRESS	4BYTE-SIZE	X	Reads data by Core debugger
proc_writeb ,pwb	ADDRESS	1BYTE-VALUE	X	Writes data by Core debugger.
proc_writes ,pws	ADDRESS	2BYTE-VALUE	X	Writes data by Core debugger.
proc_writew ,pww	ADDRESS	4BYTE-VALUE	X	Writes data by Core debugger.
proc_stop ,pst	X	X	X	Sends a halt request to the target
proc_resume ,(pre/c)	X	X	X	Resume the target
proc_read_all_regs ,prar	X	X	X	Print All registers of CPU.
proc_read_reg ,prr	REG_NUM	X	X	Print specific register of CPU
proc_write_reg,pwr	REG_NUM	X	X	Write value to the register of CPU
proc_ibreak ,pib	ADDRESS	SET/CLEAR	X	Instruction breakpoint
proc_dbreak ,pdb	ADDRESS	SET/CLEAR	X	Data Access breakpoint
runat ,ra	ADDRESS	X	X	Run program at Address
setpc				set PC register with a specified value.
step,s	X	X	X	Execute one machine instruction.

2.2. 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 SYSTEM BUS 를 통해서 그 역활을 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.

2.2.1. target TARGET_NAME(option)

ECON 과 연결한다.

TARGET_NAME 을 연결 한다.

JTAG BUS 모드로 진입한다.

"target" or "target cantus"

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.

2.2.2. help COMMAND(option)

도움말을 출력한다.

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

2.2.3. q

프로그램을 종료한다.

"exit" 와 동일하다.

2.2.4. exit

프로그램을 종료한다.

"q" 와 동일하다.

2.2.5. sleepms

일정 Milliseconds 동안 EConMan 의 명령 수행을 정지 한다. runscript 에서 필요할 경우 사용한다.

"sleepms 1000"

2.2.6. runat ADDRESS

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

즉 해당 어드레스에서 프로그램을 실행 시킬 수 있다.

"runat 0x1BA"

2.2.7. runscript SCRIPT_FILENAME

명령어로 이루어진 text 파일을 읽어서 그 명령어를 순차적으로 실행한다.

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

"runscript test_script.txt"

```
target adstar
sysinit
# comment
flash_erase 0 1
```

주석은 '#' 으로 시작한다.

2.2.8. version

프로그램 버전 정보 및 간략한 update 정보를 출력한다.

"version"

2.2.9. targetlist

현재 지원되는 모든 Target 를 출력한다.

"targetlist"

2.2.10. memtest ADDRESS SIZE

내부 알고리즘을 이용하여 메로리 Read/Write 테스트를 진행한다.

"memtest 0x20000000 1024"

2.2.11. reset, rst

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.
   * *CANTUS 는 지원하지 않는다.*

"reset"

2.2.12. debugmodereset, dreset

set_def 명령어과 reset 명령어가 순서데로 실행되는 명령어이다.
즉 reset 후 시스템이 정지된 상태가 된다.
일부 시스템에서는 지원하지 않을 수 있다.

2.2.13. jtagspeed N , swdspeed N

JTAG/SWD CLOCK Speed 를 설정 한다.

JTAG Clock = 30Mhz/(N+1)

SWD Clock = 12Mhz/(N+1)

"jtagspeed 1" 15Mhz/6Mhz 로 설정된다.

2.2.14. gdbserver PORT_NUMBER

GDB 와 통신채널로 port number 를 설정 한 후 GDB 의 Remote debug server 역할을 수행한다. 자세한 내용은 "6. E-CON 과 GDB 를 이용한 프로그램 디버깅방법"에서 다룬다.

"gdbserver 7878"

2.2.15. set_def(get_def) REG Value

H/W Debugger 에 미리 선언된 내부 register 중 하나의 값을 읽거나 쓸 수 있는 명령어이다.

현재까지 0 번 register 만 지원 한다.

set_def 0 0xfffffffe : Reset 명령(H/W reset 아님) 실행 시 CPU 가 동작 하지 않고 대기 상태로 전환된다)

참고 reset, debugmodereset

'

2.2.16. readb ADDRESS SIZE

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

"readb 0x20000000 16"

2.2.17. reads ADDRESS SIZE

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

ADRRESS 는 반드시 2 의 배수여야 한다.

"reads 0x20000002 7"

2.2.18. readw ADDRESS SIZE

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

ADRRESS 는 반드시 4 의 배수여야 한다.

"readw 0x20000004 3"

2.2.19. writeb ADDRESS 1BYTE-VALUE

특정 주소에 BYTE-VALUE 를 write 한다.

"writeb 0x20000000 0x12"

2.2.20. writes ADDRESS 2BYTE-VALUE

특정 주소에 2BYTE-VALUE 를 write 한다.

"writes 0x20000002 0x1234"

2.2.21. writew ADDRESS 4BYTE-VALUE

특정 주소에 4BYTE-VALUE 를 write 한다.

"writew 0x20000004 0x12345678"

2.2.22. writetest ADDRESS SIZE COUNT

특정 주소에 SIZE*BYTE 만큼 쓰기/읽기를 COUNT 만큼 실행한다.

만약 정상적으로 읽히지 않는 다면 에러메시지를 출력 한다.

COUNT 인자는 옵션이다.

쓰는 방식은 4Byte 단위로 0,4,8,12,16,... write 한 후 다시 읽어서 비교 한다.

2.2.23. fileread ADDRESS SIZE FILE_NAME

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

"fileread 0 0x100 dump.bin"

2.2.24. filewrite ADDRESS FILE_NAME noverify/readyrun(option)

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

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

noverify: 검증하지 않음(option)

readyrun: file 의 첫번재 4byte 를 프로세서의 시작주소로 설정한다.(option)

PC=*(unsigned int*)(ADDRESS)

"filewrite 0 dump.bin"

2.2.25. 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.

2.2.26. flash_eraseall

Flash Memory 전체를 Erase 한다.

2.2.27. flash_erase START_SECTOR SECTOR_COUNT

START_SECTOR 부터 SECTOR_COUNT 만큼 Erase 한다.

"flash_erase 0 1"

2.2.28. flash_filewrite ADDRESS FILENAME noverify/readyrun(option)

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 안의 기존 data 는 지워진다.

noverify: 검증하지 않음 (option)

readyrun: file 의 첫번재 4byte 를 프로세서의 시작주소로 설정한다.(option)

PC=*(unsigned int*)(ADDRESS)

"flash_filewrite 0x0 dump.bin"

2.2.29. flash_fileread ADDRESS SIZE FILENAME

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

"flash_fileread 0x0 1024 dump.txt"

2.2.30. flash_writetest

"writetest" 명령어와 동작 방식은 동일하나 flash memory 이므로 write 전에 erase 를 먼저 실행 하게 된다.

참고 writetest

2.2.31. nand_init

NAND Flash Memory Controller 를 초기화한다.

장착된 NAND Flash memory 정보를 수집 및 출력한다.

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

2.2.32. nand_eraseall

NAND Flash Memory 전체를 Erase 한다.

2.2.33. nand_erase START_block block_COUNT

START_BLOCK 부터 BLOCK_COUNT 만큼 Erase 한다.

"nand_erase 0 1"

2.2.34. nand_filewrite ADDRESS FILENAME

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

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

file data 가 write 되지 않는 block 안의 기존 data 는 지워진다.

"nand_filewrite 0x0 dump.bin"

2.2.35. nand_fileread ADDRESS SIZE FILENAME

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

"nand_fileread 0x0 1024 dump.txt"

2.2.36. nand x, x, x

Nand 관련 명령어들을 사용 할 수 있다.

옵션은 아래와 같다.

전달인자 1	전달인자 2	전달인자 3	기능

init	x	x	nand_init 와 동일
eraseall	x	x	nand_eraseall 와 동일
erase	x	x	nand_erase 와 동일
read	x	x	nand_read 와 동일
readfile	x	x	nand_fileread 와 동일
writefile	x	x	nand_filewrite 와 동일
fat	init	x	nand 에서 FAT filesystem 을 정보를 얻는다.
	list	x	현재 directory 수준에서 파일과 directory 를 출력한다.
	mkfs	x	FAT filesystem 을 포맷 한다.
	sync	x	아직 NAND 에 저장하지 않은 data 를 write 한다. 만약 file system 변경이 있었다면 반드시 종료 전에 실행해야 한다.
	mkdir	directory name	directory 를 생성한다.
	dcopy	Local directory	PC 상의 directory 안의 모든 파일(하위 directory 포함) 을

			NAND 의 filesystem (존재하지 않을 경우 생성) 에 복사 한다.
	cd	directory	change directory(NAND)
	fwrite	local filename	HOST PC 상의 filename 을 NAND filesystem 에 복사한다.
	fread	remote file name	NAND 에 존재하는 filename 을 HOST PC 에 복사한다. HOST PC 의 현재 폴더에 저장된다.
	fatimage	file name	FAT Image generator 로 생성된 파일시스템 Image 파일을 NAND 에 write 한다. 만약 기존 Filesystem 은 지워진다.
	del	file name	NAND 의 file 을 삭제 한다.

2.2.37. proc_readb ADDRESS SIZE

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

"proc_readb 0x20000000 16"

2.2.38. proc_reads ADDRESS SIZE

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

"proc_reads 0x20000002 7"

2.2.39. proc_readw ADDRESS SIZE

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

"proc_readw 0x20000004 3"

2.2.40. proc_writeb ADDRESS 1BYTE-VALUE

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

"proc_writeb 0x20000000 0x12"

2.2.41. proc_writes ADDRESS 2BYTE-VALUE

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

"proc_writes 0x20000002 0x1234"

2.2.42. proc_writew ADDRESS 4BYTE-VALUE

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

"proc_writew 0x20000004 0x12345678"

2.2.43. proc_stop

Target system 의 EISC Core 강제로 멈춘다.

"proc_stop"

2.2.44. proc_resume

정지된 EISC Core 를 재 실행 시킨다.

"proc_resume"

2.2.45. proc_read_all_regs

CPU 내의 모든 register 값을 출력한다.

"proc_read_all_regs"

2.2.46. 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

"proc_read_reg 1"

2.2.47. proc_write_reg REGNUM 4BYTE-VALUE

CPU 내의 특정 register 에 특정 값을 기록한다.

"proc_write_reg 1 0x12345678"

2.2.48. proc_ibreak ADDRESS SET/CLEAR

Instruction 이 있는 특정 Address 에서 Processor Stop 을 SET 또는 CLEAR.

즉 해당 주소의 Instruction 을 수행하기 전에 프로그램 실행의 중지 유무를 설정할 수 있다.

"proc_ibreak 0x322 1"

2.2.49. proc_dbreak ADDRESS SET/CLEAR

Address 의 DATA 를 Access 하기 전에 Processor Stop 을 SET 또는 CLEAR.

즉 해당 주소의 DATA 를 Access 하기 전에 프로그램 실행의 중지 유무를 설정할 수 있다.

"proc_dbreak 0x60024000 1"

2.3. 프로그램 실행 명령어 옵션

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

ecoman.exe –command1 arg1 arg2 arg3 –command2 arg1

와 같이 명령어 앞에 '-' 만 붙이면 된다.

이 경우 순차적으로 명령어를 실행 하게 되며 특정 명령어의 실행이 올바르지 않을 경우 그 이후 명령어는 실행 하지 않는다.

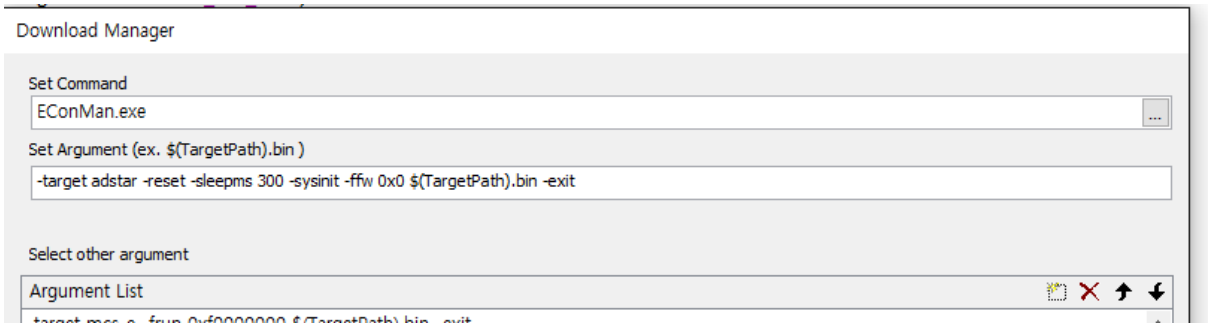
ex) econman.exe –target cantus –systeminit –flash_filewrite 0 bootloader.bin –exit

- CANTUS Target board 를 연결
- 미리 설정된 시스템 초기화 루틴 실행
- 0 번에 bootloader.bin 파일을 다운로드
- 종료

3. 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 를 다운로드 할 수 있다.



"Download Option" 을 실행하여 위 그림과 같이 설정한 이후 "Download to Target"을 실행 하면 EConMan.exe 를 이용하여 Target 에 다운로드 할 수 있다.

4. 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.

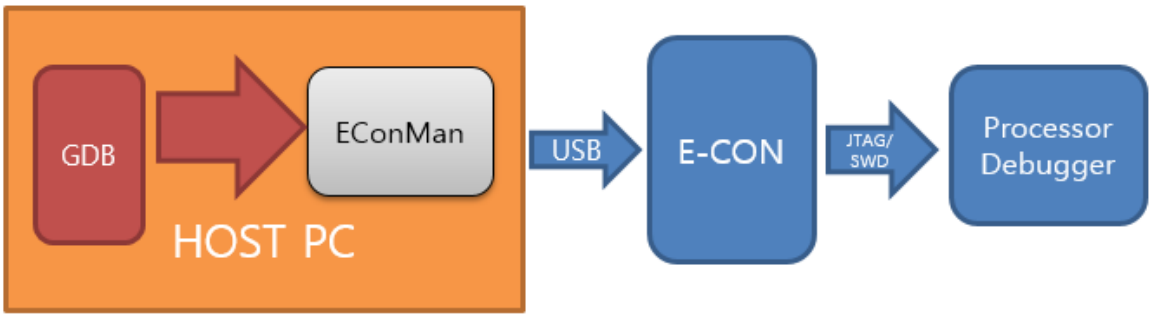
첫 번째는 프로그램을 최초 부팅 시부터 디버깅을 하는 것이고 다른 하나는 현재 동작중인 프로그램을 세워서 그 상태에서부터 디버깅 하는 것이다.

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.

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 실행하면 현재 멈춰진 상태에서부터 디버깅을 시작 할 수 있다.

EConMan 과 GDB 의 연결은 socket 통신으로 연결된다.



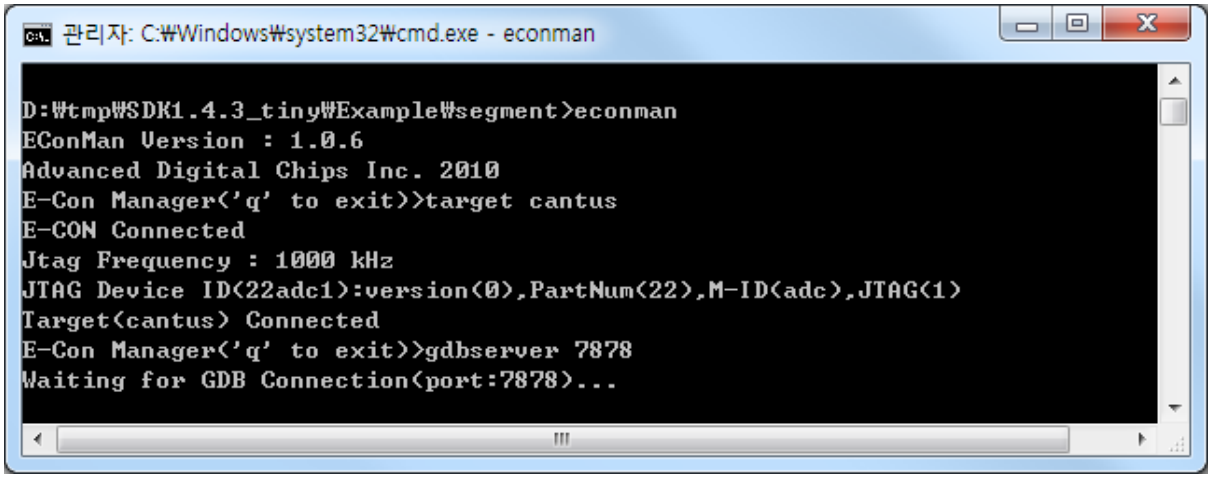
따라서 GDB 를 실행 할 때 target 을 socket 으로 연결 하여야 한다.

"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.

4.1. EISC Studio Command Prompt 에서 Debugging

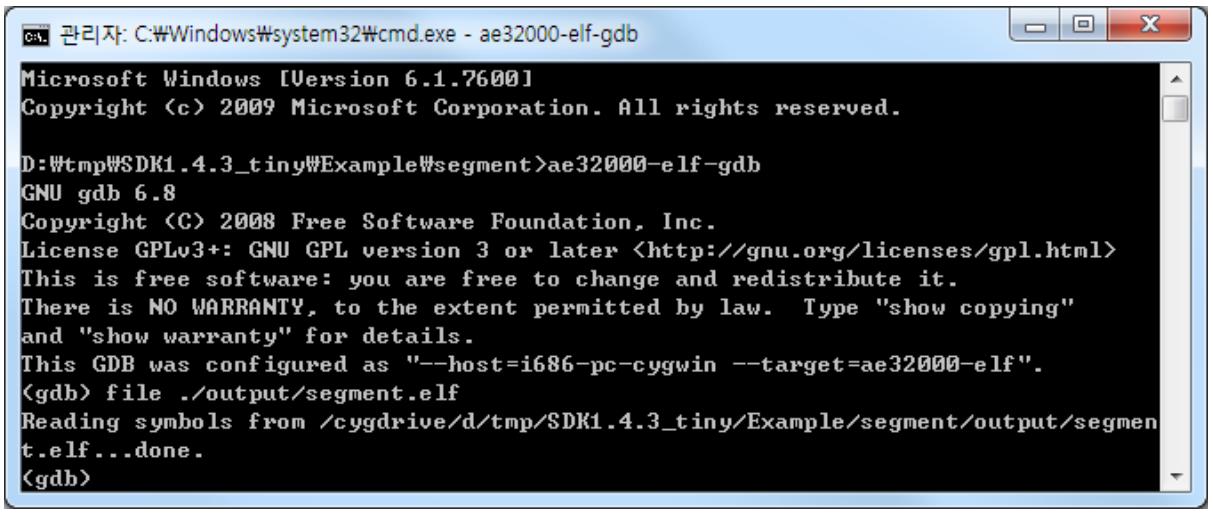
Prompt 창에서 GDB 명령 입력으로 Debugging 이 이루어 진다. GDB 사용에 익숙하지 않은 사용자는 "2. EISC Studio 3 에서 Debugging"의 내용을 따른다.

A. EConMan 을 실행하고, target 을 연결한 후 "gdbserver 7878" 실행 하여 socket 을 open 한다.
정상적으로 실행 될 경우 더 이상 사용자 명령어를 받아들이는 prompt 가 뜨지 않는다.



B. EISC Studio Command Prompt 를 하나 더 실행하여, ae32000-elf-gdb.exe 를 실행한다.

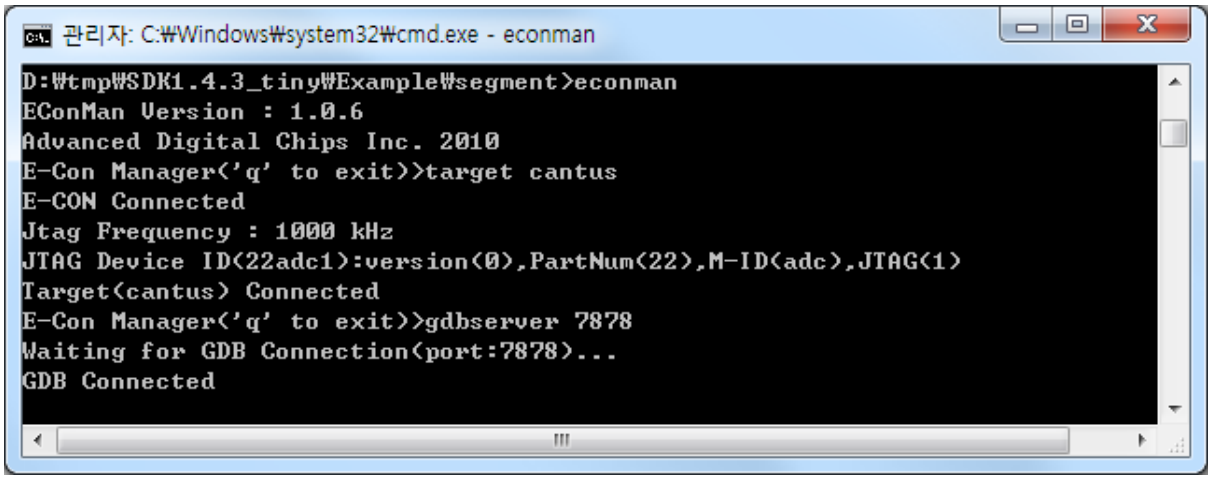
C. "file"명령어를 이용하여 *.elf file 을 Open 한다.



D. "set remote hardware-breakpoint-packet 1"을 입력한다.

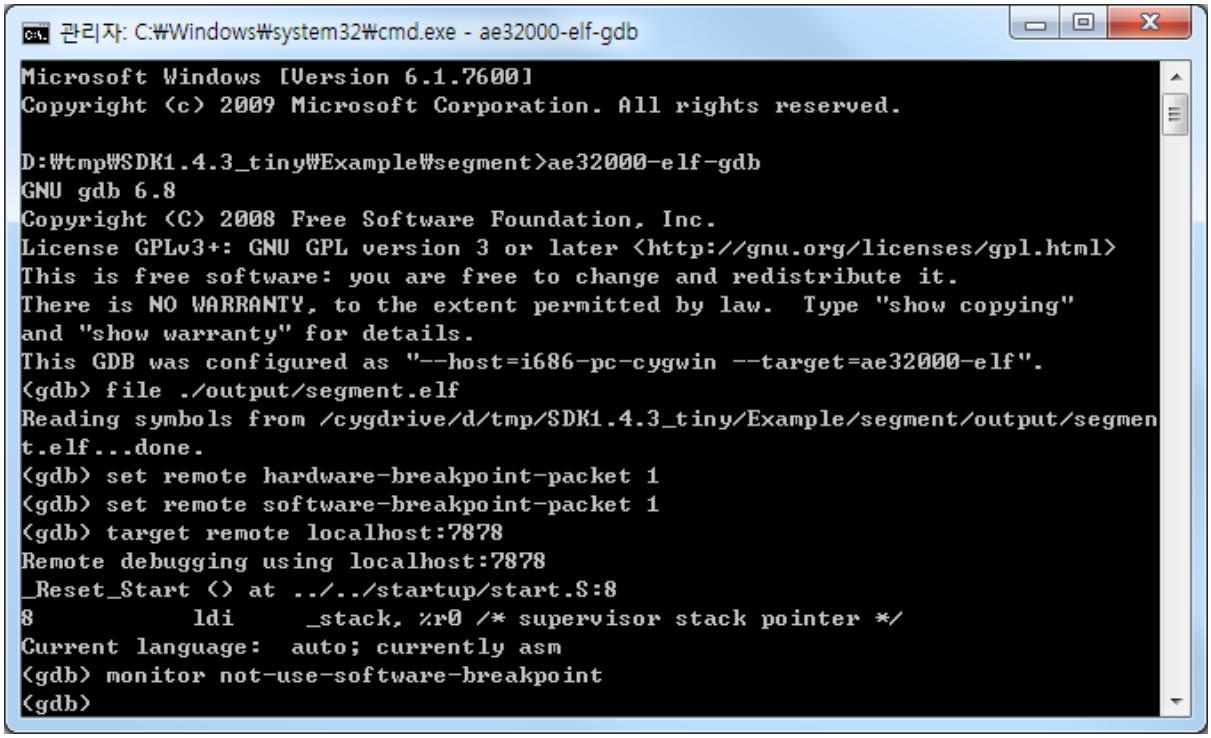
E. "set remote software-breakpoint-packet 1"을 입력한다.

F. "target remote localhost:7878"를 이용하여 'A'에서 Open 한 socket 에 접속 한다. 접속이 되면, 아래와 같이 접속 대기 중에서 접속으로 상태가 변경 된다.

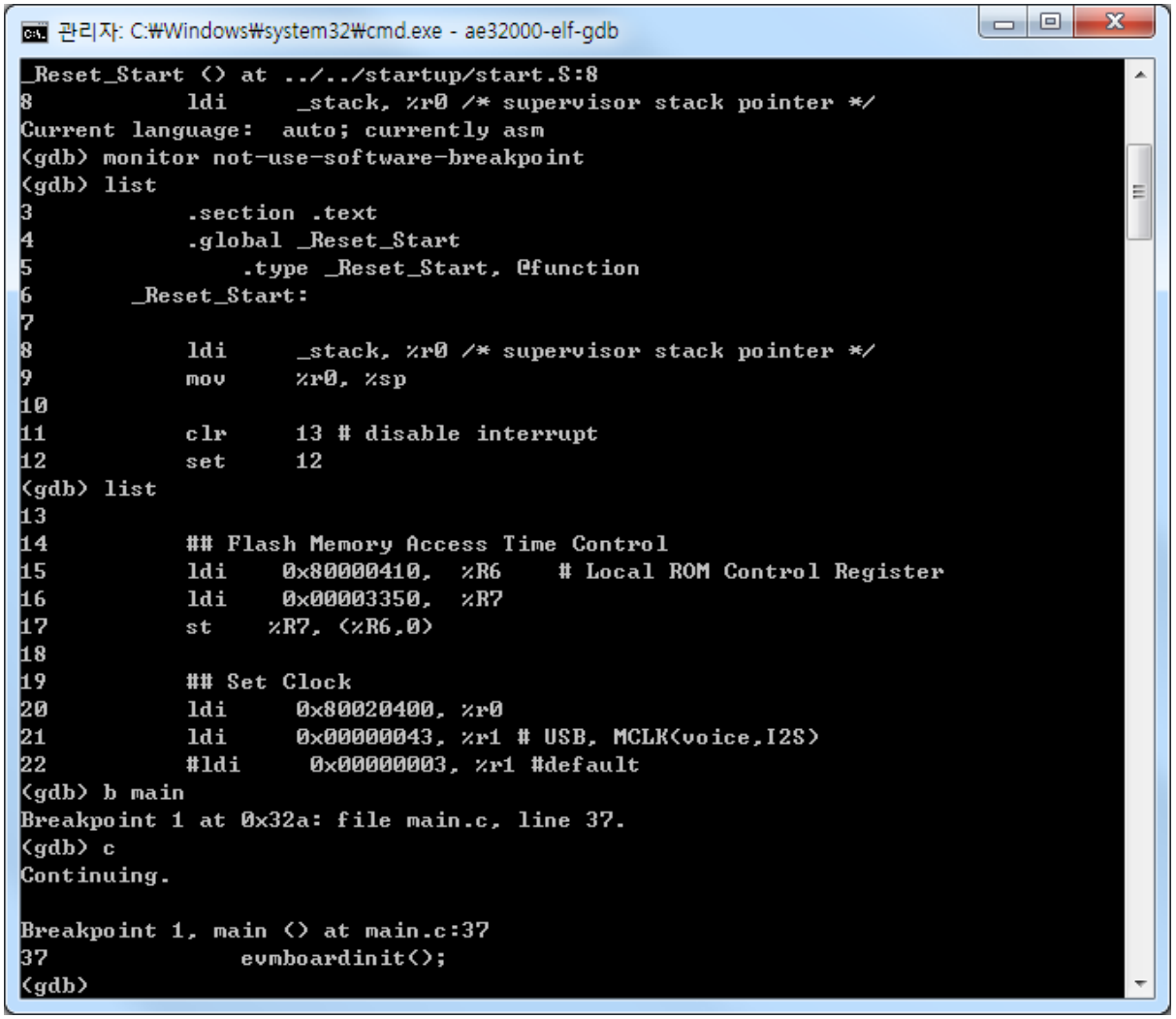


G. 이때 Process 가 멈춰 있는 지점이 표시 된다.
at ../../startup/start.S:8

H. "monitor not-use-software-breakpoint"를 설정한다. 이상 실행한 결과는 아래와 같다.

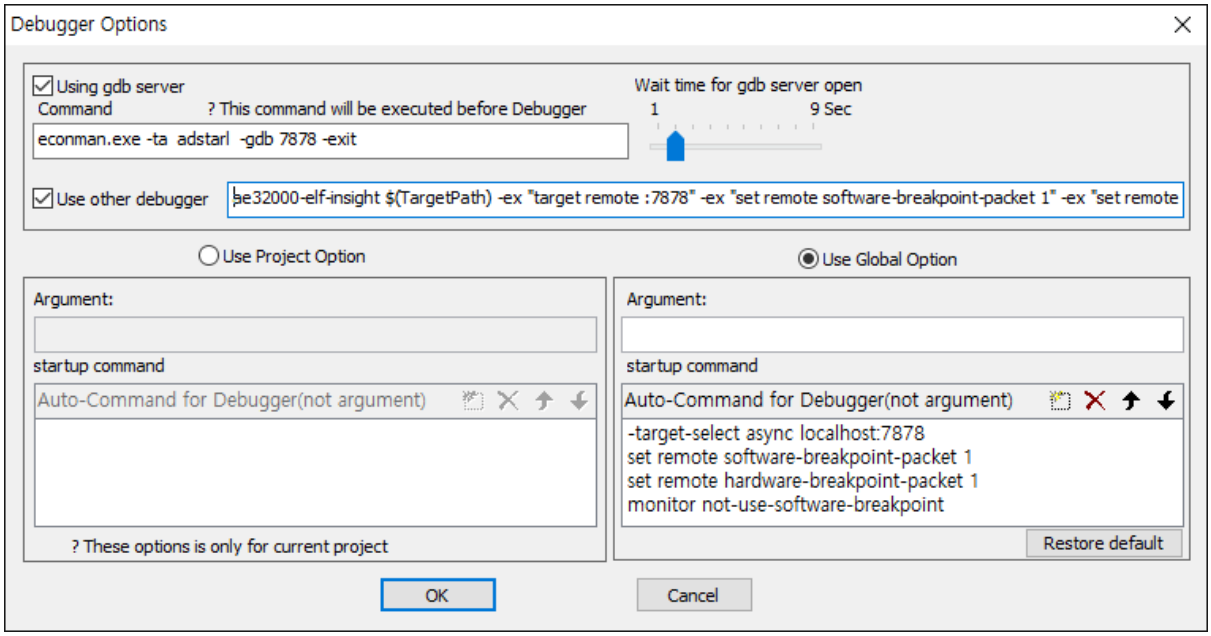


I. 이 상태에서 GDB 명령을 입력하여, Debugging 을 진행 한다.
예로 "list"명령을 이용하여 code 를 출력하고, main 에 breakpoint 를 설정하고, 진행(continue)한 결과는 다음과 같다.



4.2. EISC Studio 3 에서 Debugging

A. Menu 에서 "Debug"→"Debug Options" 에 아래와 같이 설정 한 후 "Start Debugger" 를 실행한다.

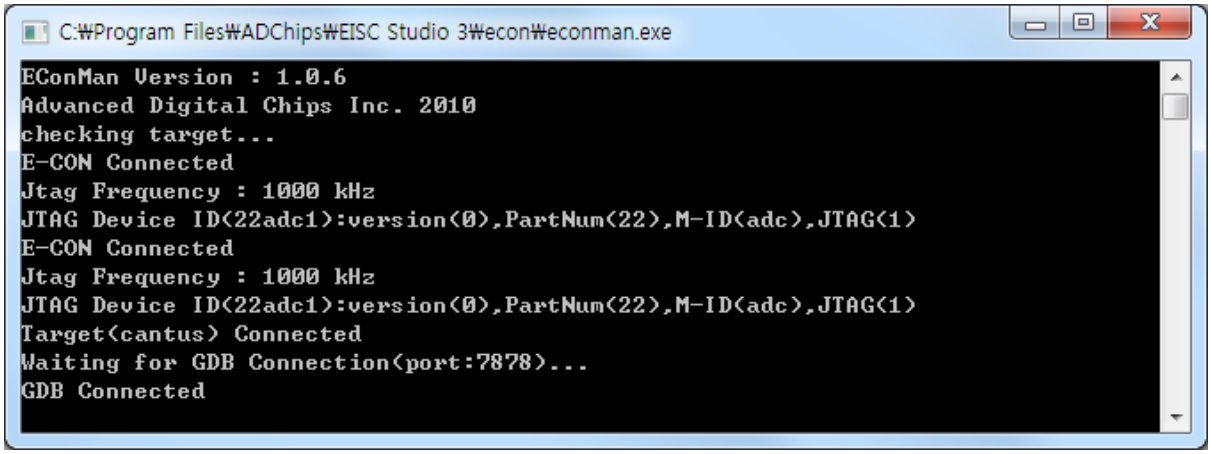


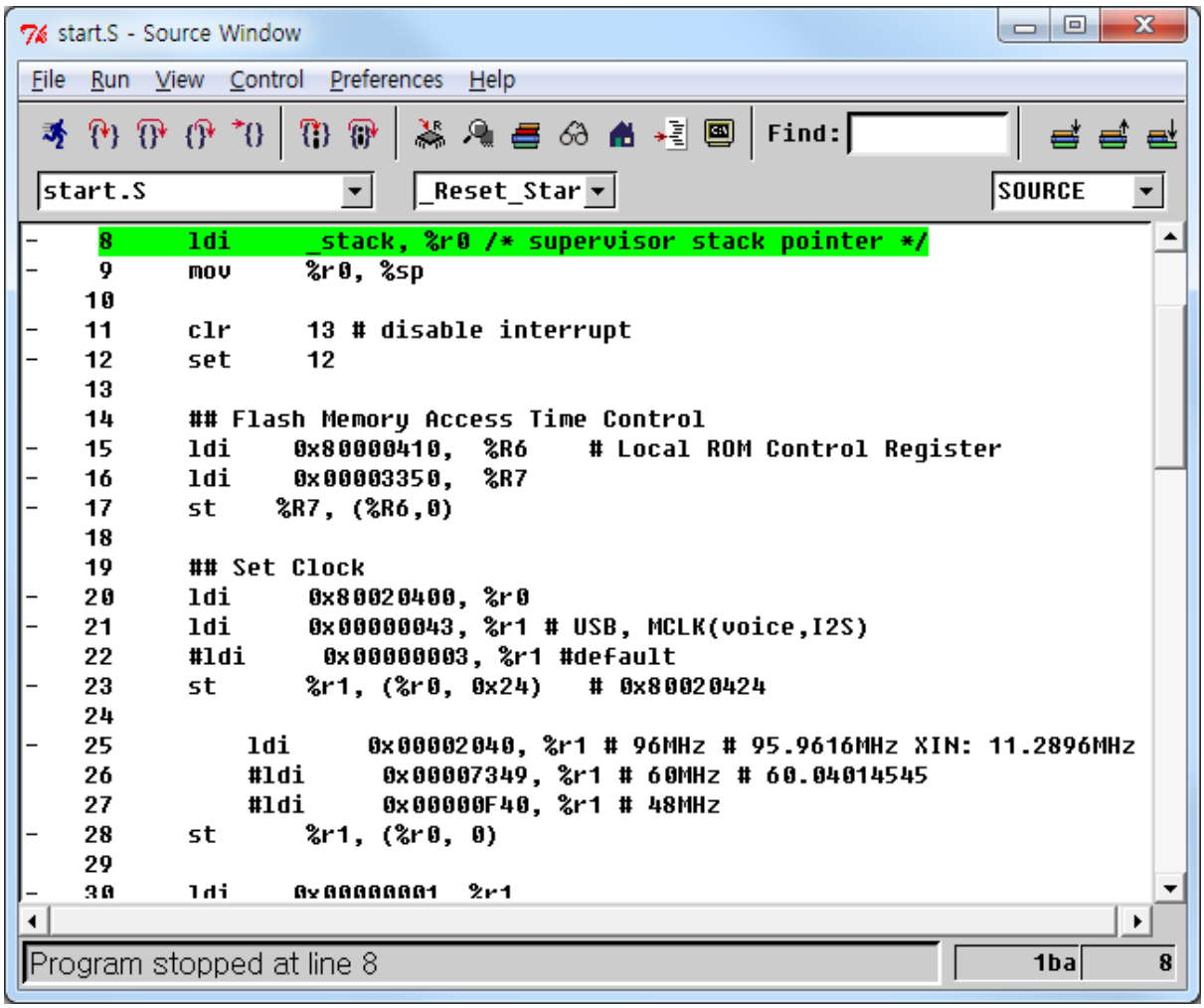
Using gdb server : Debugger 가 실행할 때 가장 먼저 실행되는 command 로, EConMan 을 사용하여, target 에 연결하고, Port 7878 로 gdbserver 를 열어 둔다.
사용하고자 하는 target 명을 반드시 입력해야 한다.
JTAG 은 1Mhz SWD 는 200Khz 가 기본 연결 속도이다.
JTAG/SWD 동작 주파수는 반드시 시스템 동작 주파수의 1/4 보다 낮아야 한다. 해당 칩의 최대 안정 속도가 이 보다 빠르다면 JTAG/SWD 속도를 올릴 수 있다. 하지만 프로그램이 동작 중에 PLL 을 사용해서 시스템 속도를 올리는 경우라면 PLL 이 동작된 이후에 JTAG/SWD 속도를 올릴 수 있으므로 처음부터 속도를 올리면 안된다.

Use Project Option : 현재 Project 에서만 유효한 Debugger Type 과 Argument 를 설정한다.

Use Global Option : 모든 Project 에서 유효한 Debugger Type 과 Argument 를 설정한다.
기본적으로 Argument 가 채워져 있다. 특별한 경우가 아니라면 별도 수정은 필요는 없을 것이다.

B. 위 "Debugger Options"에서 상단의 "Using gdb server"가 가장 먼저 실행되어, EConMan 을 실행하여 target 을 설정하고, Port 7878 로 gdbserver 를 열어둔다. 그 후에 "Use Global Option"에서 설정한 Insight 가 실행되어 접속한다.





"Select function name to disassemble"이라고 나타나면 위 start.S 부분을 click 하여 start.S 를 선택한다.

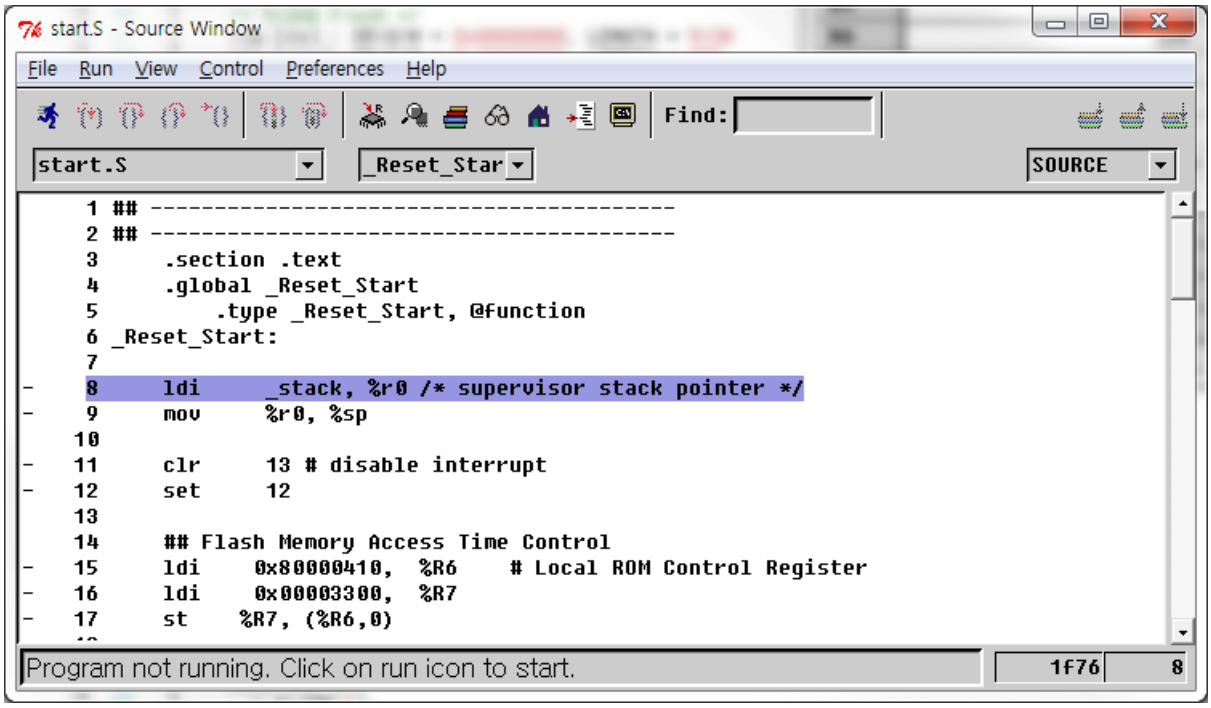
4.3. Debugging 시 주의할 점

A. Debugging 할 Project 는 –g 옵션을 이용하여 compile 되어야 한다.

B. Compile 된 Program 은 Download 되어 있어야 한다.

C. Insight 가 실행되면 프로그램이 실행준비 완료 상태이기 때문에 "Run"을 사용해서는 안 된다.

D. 아래와 같이 Code Line 이 보라색으로 나타나면, 정상적으로 연결된 상태가 아님을 뜻한다. E-CON 과 Target 의 연결 상태를 점검하고, EConMan 이 올바르게 실행 되었는지 확인해 본다.



좀 더 자세한 GDB 명령어는 아래 링크 주소에서 얻을 수 있다.

http://sourceware.org/gdb/

5. Serial Flash 지원

연결된 serial flash 의 정보가 E-Con 이 자체적으로 갖고 있는 정보에 없다면 사용자가 추가 할 수 있다. (version 1.4.25 이후)

E-Con 이 설치된 폴더의 config/serialflash.db 파일을 아래와 같이 생성 해서 사용하면 된다.

정보가 중복 될 경우 아래와 같은 우선순위를 가진다.

1. 외부 지정 목록 리스트(econ/config/serialflash.db)
2. E-Con 내부에서 이미 선언되어 있는 리스트
3. JEDEC Standard JESD216 SFDP 지원하는 serial flash

config/serialflash.db 파일에 새로운 메모리 정보를 추가하기 위해서는 serialflash.db 파일에 편집기를 이용해서 새로운 정보를 입력 하면 된다.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<SerialFlash version="1">

    <Flash Name = "SPANSION" ID1 ="0x1" ID2="0x40" ID3="0x15"
SectorSize="0x1000" BlockSize="0x10000" Size="0x400000" Quad="true"/>

<Flash Name = "SPANSION" ID1 ="0x1" ID2="0x40" ID3="0x16" SectorSize="0x1000"
BlockSize="0x10000" Size="0x800000" Quad="true"/>

</SerialFlash>
```

Name : 제품명

ID1 : Manufacturer

ID2 : memory type

ID3 : capacity

SectorSize = sector size

BlockSize = block size

Size = memory size

해당 정보는 serial flash 의 datasheet 를 참고하여 입력 하면된다.

SectorSize 는 Sector Erase 명령어(0x20h)로 지워지는 크기를 의미한다.

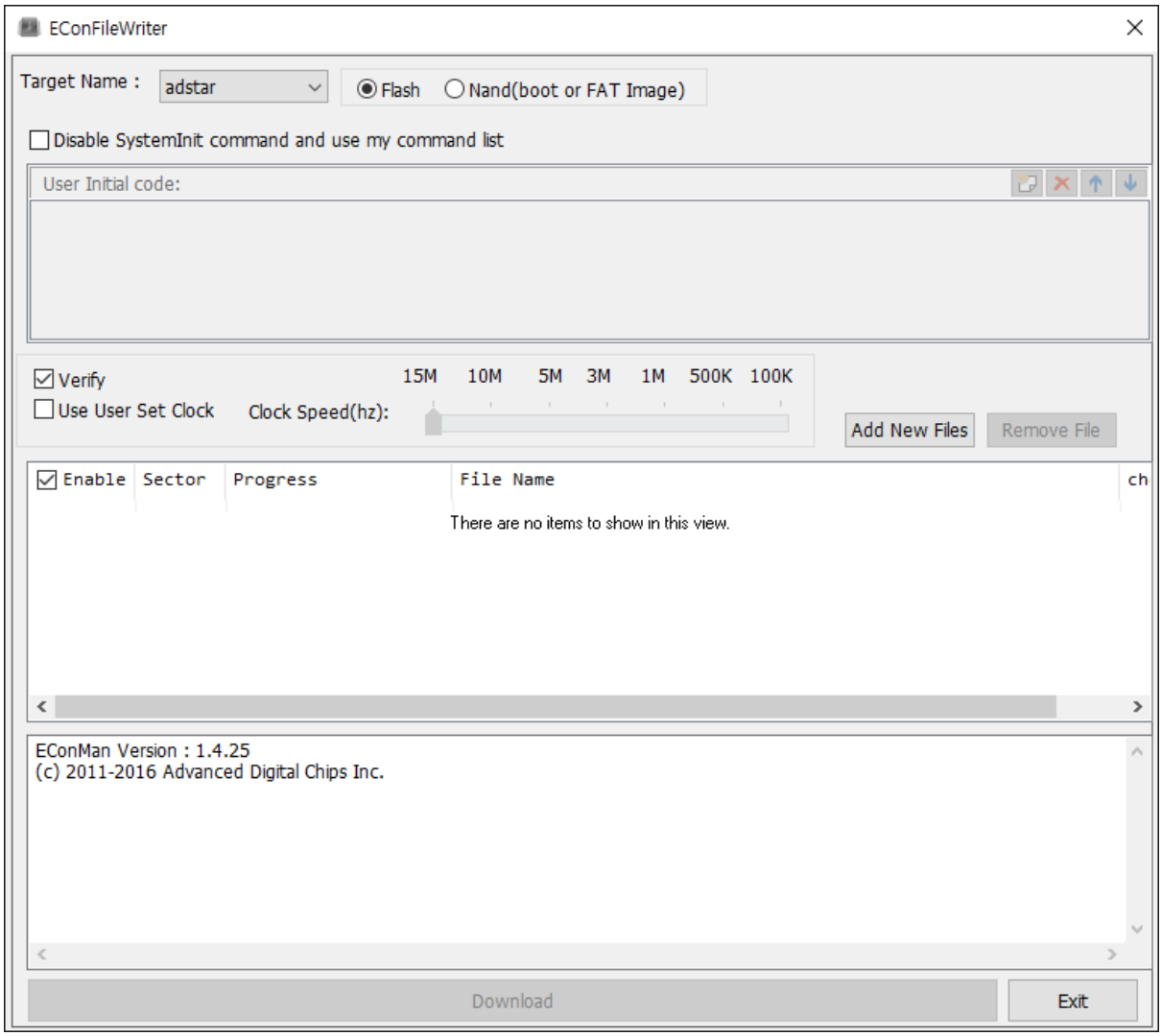
BlockSize 는 block Erase 명령어(0xD8h) 로 지워지는 크기를 의미한다.

만약 해당 serial flash 가 Sector Erase 명령어가 0x20 이 아니거나 없을 경우 SectorSize 를 BlockSize 와 동일한 값을 입력해서 사용 할 수 있다.

어떤 경우에도 반드시 Block Erase 명령어는 반드시 0xD8h 이어야만 한다.

6. EConFileWriter

EConFileWriter 는 E-Con 을 이용하여 Serial Flash 또는 Nand Flash 에 파일을 프로그래밍할 수 있는 기능만을 위한 GUI 환경의 Application 이다.



"Disable SystemInit command and use my command list" : 파일다운로드 하기 이전에 수행되는 systeminit 대신 사용자의 명령어를 이용하여 초기화 할 수 있다. 명령어 종류 및 사용법은 EConMan 명령어와 동일하다.

"Verify" : 검증 수행 여부

"Use User Set Clock" : systeminit 시 설정되는 JTAG/SWD clock 대신 사용자가 직접 선택 할 수 있다.

"Add New Files" : 다운로드 할 파일을 선택 할 수 있다. Serial Flash 의 경우 Sector Number 를 지정할 수 있고 Nand Flash 의 경우 Block Number 를 지정 할 수 있다.

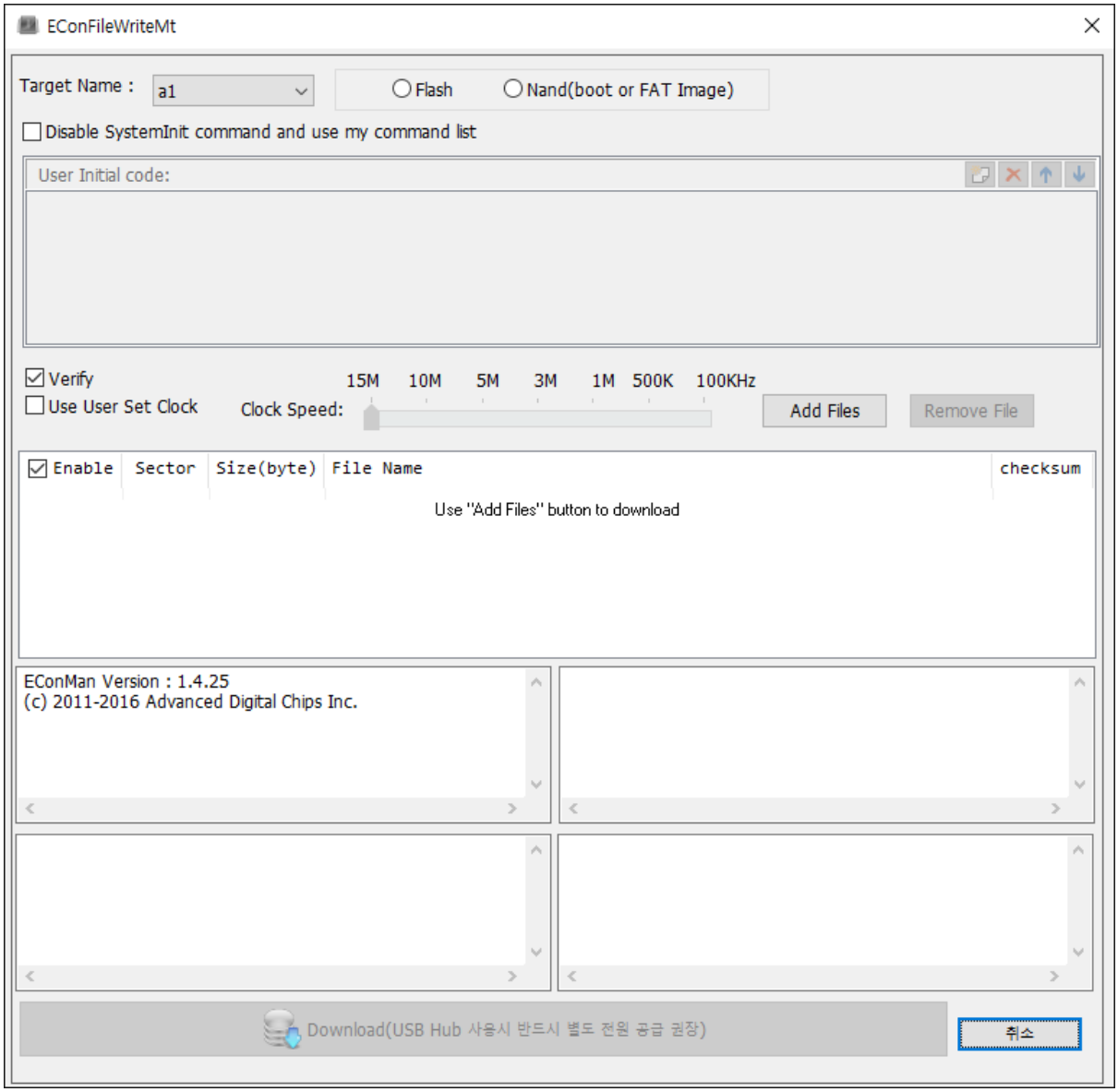
파일 추가와 모든 설정이 완료 되면 "Download" 버튼을 이용하여 파일을 다운로드 할 수 있다.

7. EConFileWriteMt

하나의 HOST PC 와 최대 4 개의 E-Con 을 연결하여 파일을 다운로드 할 수 있다.

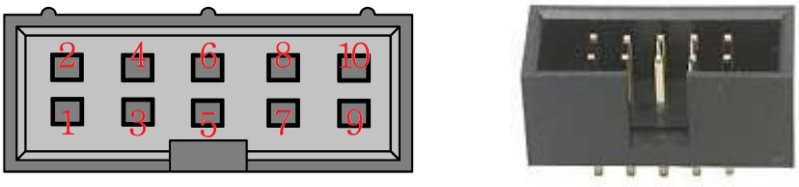
사용법은 EConFileWriter 와 동일하다.

**2 개를 연결하여 다운로드 한다고 해서 1 개씩 할 때 보다 2 배의 속도가 나오는 것은 아니다.*



8. E-CON 과 Target Board 연결 정보

E-Con 과 연결될 Target Board 의 Connector 정보는 다음과 같다.



<2.54mm Pitch 10-Pin Header Box>

JTAG (SWD)			
NO	**PIN**	**NO**	**PIN**
1	**TDI**	**2**	**V3P3D**
3	**TMS(SDA)**	**4**	**NC**
5	**TRST**	**6**	**TDO**
7	**TCK(SCK)**	**8**	**NC**
9	**GND**	**10**	**GND**

.