[1]

**SDK 설치**

**CCS 설치**

[2]

**DSP 보드에 연결 (준비작업)**

-보드의 EVM 케이블 연결

-이더넷 케이블 연결

-살 때 받은, SD카드 삽입

-5V 전원선 연결

모두 연결 했다면, 전원 인가! (보드에서 Power 버튼을 찾아서 누른다)

보드의 터치패드에서 > Settings > Network Settings > eth0 의 inet addr : 192.168.0.17 을

웹 브라우저에 입력해보면, 터치패드와 동일하게 사용해볼 수 있다

[3]

**SDK 에 있는 내장 GCC Tool Chain 살펴보고, 환경설정 Script를 사용하여 이 Tool Chain을 사용할수 있도록 구성하는 방법 학습**

크로스 컴파일러는 SDK설치 디렉토리의 {개인별 설정한 디렉토리}/linux-devkit/sysroots/ <Arago Linux>/usr/bin 에 있다

따라서, 먼저 터미널을 열고, 다음을 입력 (해당하는 디렉토리로 이동하는것)

>cd /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit

/sysroots/x86\_64-arago-linux/usr/bin

아래와 같이 Cross-Compile Tool 목록을 확인한다

>ls

2to3 ldattach abs6x libinfo6x

acp6x libtoolize acpia6x linux32

acpll6x linux64 addpart lnk6x

ap6x logger ap6x3 logger.util-linux

ar6x look arm-linux-gnueabihf-addr2line lsblk

arm-linux-gnueabihf-ar lsipc arm-linux-gnueabihf-as lslocks

arm-linux-gnueabihf-c++filt lslogins arm-linux-gnueabihf-cpp mesg

arm-linux-gnueabihf-g++ mesg.util-linux arm-linux-gnueabihf-gcc mkfs.bfs

arm-linux-gnueabihf-gcc-5.3.1 mkfs.ubifs arm-linux-gnueabihf-gcc-ar namei

arm-linux-gnueabihf-gcc-nm nm6x arm-linux-gnueabihf-gcc-ranlib nologin

arm-linux-gnueabihf-gcov nsenter arm-linux-gnueabihf-gdb ofd6x

arm-linux-gnueabihf-gprof omps2s6x arm-linux-gnueabihf-ld openssl

arm-linux-gnueabihf-ld.bfd opkg arm-linux-gnueabihf-nm opkg-check-config arm-linux-gnueabihf-objcopy opt6x arm-linux-gnueabihf-objdump pdd6x arm-linux-gnueabihf-ranlib pg

arm-linux-gnueabihf-readelf pkg-config arm-linux-gnueabihf-size pprof6x

arm-linux-gnueabihf-strings prelink6x arm-linux-gnueabihf-strip prlimit

asm6x pydoc blkdiscard qt5

cal raw cg6x rename

cg6x3 renice chcpu renice.util-linux

chrt resizepart chrt.util-linux rev

ci6x rtcwake cl6x script

clacc scriptreplay clist6x setarch

clocl setsid clt6x setsid.util-linux

cmp6x setterm col smtpd.py

colcrt strip6x colrm swaplabel

column tailf delpart taskset

dem6x ubiattach dis6x ubiblock

eject ubicrc32 eject.util-linux ubidetach

embed6x ubiformat fdformat ubimkvol

findmnt ubinfo flock ubinize

flock.util-linux ubirename fsfreeze ubirmvol

git ubirsvol git.real ubiupdatevol

git-receive-pack ul git-shell uname26

git-upload-archive unshare git-upload-pack update-alternatives hex6x update-ca-certificates hexdump utmpdump hexdump.util-linux utmpdump.util-linux i386 wall idle wall.util-linux ilk6x wdctl ionice whereis ipcmk wipefs ipcrm write ipcs x86\_64 isosize x86\_64-arago-linux-gnu-pkg-config last x86\_64-arago-linux-libtool lastb xref6x lastb.util-linux zramctl last.util-linux

이 중 , 사전에 Build 된 ARM Library를 찾기 위해 , 다음을 터미널에 입력한다

>cd /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit

/sysroots/x86\_64-arago-linux/usr/lib

아래와 같이 Library 목록을 확인할 수 있다(일부는 개별 하위 디렉토리에 포함되어 있다)

armv7ahf-neon-linux-gnueabi libhistory.so.6.3 libQt5Sql.so.5.6.2

gcc liblzo2.so.2 libQt5Test.so.5

libarchive.so.13 liblzo2.so.2.0.0 libQt5Test.so.5.6

libarchive.so.13.1.2 libmount.so.1 libQt5Test.so.5.6.2

libblkid.so.1 libmount.so.1.1.0 libQt5Xml.so.5

libblkid.so.1.1.0 libopkg.so.1 libQt5Xml.so.5.6

libbz2.so.1 libopkg.so.1.0.0 libQt5Xml.so.5.6.2

libbz2.so.1.0.6 libpanelw.so.5 libreadline.so.6

libcrypto.so.1.0.0 libpanelw.so.5.9 libreadline.so.6.3

libcurl.so.4 libpython2.7.so.1.0 libsmartcols.so.1

libcurl.so.4.4.0 libQt5Concurrent.so.5 libsmartcols.so.1.1.0

libelf-0.164.so libQt5Concurrent.so.5.6 libsqlite3.so.0

libelf.so.1 libQt5Concurrent.so.5.6.2 libsqlite3.so.0.8.6

libexpat.so.1 libQt5Core.so.5 libssl.so.1.0.0

libexpat.so.1.6.0 libQt5Core.so.5.6 libstdc++.so.6

libfdisk.so.1 libQt5Core.so.5.6.2 libstdc++.so.6.0.21

libfdisk.so.1.1.0 libQt5DBus.so.5 libuuid.so.1

libffi.so.6 libQt5DBus.so.5.6 libuuid.so.1.3.0

libffi.so.6.0.4 libQt5DBus.so.5.6.2 libz.so.1

libgdbm\_compat.so.4 libQt5Network.so.5 libz.so.1.2.8

libgdbm\_compat.so.4.0.0 libQt5Network.so.5.6 python2.7

libgdbm.so.4 libQt5Network.so.5.6.2 ssl

libgdbm.so.4.0.0 libQt5Sql.so.5

libhistory.so.6 libQt5Sql.so.5.6

Cross-Compilations를 보다 쉽게 하고,

Host System Library 대신, 적절하게 Cross Compile 된 Library와 연결되도록 하기 위해  
환경설정 Script 가 **linux-devkit** 디렉토리에 생성되어 있다

이 Script는 CC와 같은 많은 표준변수를 구성하여 Cross-Compile Tool Chain을 사용하고, PATH에 Tool Chain을 추가하고 Library 위치에 대한 경로를 구성한다.

(환경설정 Script에서 제공하는 설정을 사용하려면 Script 를 제공해야 한다)

따라서, 환경설정 Script를 제공하고 CC변수의 변경사항을 관찰하기 위해 ,

아래내용을 터미널에 입력하자

(해당 폴더로 이동)

> cd /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit

(source 하기 - setup 하는 건가..)

> source environment-setup

(그러면 다음과 같은 내용이 뜬다)

[linux-devkit]:~/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit>

(이 뒤에 echo $CC 입력하기)

echo $CC

(그러면 다음과 같은 내용이 뜬다)

arm-linux-gnueabihf-gcc

이 변경의 목적은 환경설정 Script 가 언제 제공되었는지를 쉽게 식별할 수 있게 하는것이다

**환경설정 Script를 제공하지 않으려는 경우**가 있으므로, 이는 중요한 사항이다

완벽한 예를 들자면, **Linux Kernel을 Build 할 때** 이다.

Kernel 을 Build 하는 동안 Kernel Build Process를 지원하기 위해 Host 에서 실행될 일부 Application이 Compile 된다

또한, 특정 Build 중에 source를 원하지만 기본환경에는 영향을 미치지 않는점을 참고하자.

SDK에서 이 작업을 수행하는 방법은, Build Process 중에만 사용되도록 Project Makefile의 환경설정 Script를 제공하는 것이다.

이를 확인하기 위해, 아래 작업을 수행하자.

-이미 환경설정 Script가 source 되었으므로, 기존 터미널을 닫고, 새 터미널을 연다

-다음을 터미널에 입력한다

(해당하는 디렉토리로 이동하기)

> cd /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/

example-applications/matrix-gui-browser-2.0/

(에디터 열기)

> gedit Makefile.build

어떤 창이 뜬다. (에디터창) 그럼 확인해보자.

**ENV\_SETUP ?= ../../linux-devkit/environment-setup**

을 통해, 환경설정 Script를 가리키는 ENV\_SETUP이라는 변수를 설정했음을 확인할 수 있다

**qmake : matrix\_browser.pro**

**@.** ${ENV\_SETUP}; \

**qmake CONFIG+=release matrix\_browser.pro**

을 통해, "@.${ENV\_SETUP}” **action** 을 수행한다.

위의 **“** @ **”** 기호는 Shell 명령을 실행함을 의미한다

위의 **“ . “** 기호를 통해, qmake2 명령을 호출하기 전에, 파일을 sourcing하는 또 다른 방법으로 환경 설정 스크립트를 제공하는 부분임.

이 작업을 보려면, matrix-gui-browser 프로젝트를 cross-compile 한다

따라서, 아래를 터미널에 입력하자.

(Makefile 빌드하는것)

> make -f Makefile.build

arm-linux-gnueabihf-g ++ 컴파일러가 Build 하는 동안 호출된 출력에 주목하라.   
또한 프롬프트에 sitara @ ubuntu 가 표시되어 환경 설정 스크립트가 Build를 위해 제공된 동안 make가 끝난 후에도 효과가 사라졌다는 것을 알 수 있다.

"Nothing to be done for ..." 출력이 표시되면 이전에 make 명령을 사용하여 이 File 을 Build 했음을 의미한다. 이 때, 명시된 출력을 보려면 make clean first 명령을 실행한다.

빌드된 matrix\_browser 실행 파일이 실제로 교차 컴파일 되었는지 확인하려면 다음 명령을 사용할 수 있다: 다음을 터미널에 입력

> file matrix\_browser

입력하면, 다음과 같이 확인할 수 있다

> matrix\_browser: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=312124e535844fd51f9a3508f067344872ffd524, not stripped

\*\*ARM 용, dynamically linked : 공유 라이브러리 사용

[4]

여기서 Processor Linux SDK 의 루트 내에 최상위 Makefile 에 대해 설명한다.

어떻게 사용되는지 주요 변수에 대해 논의하고 목표를 세운다.

Processor Linux SDK 최상위 Makefile 에서 더 많은 정보를 찾을 수 있다

[키 포인트]

\* 최상위 Makefile 의 위치

\* Rules.make 파일의 추가 설정

\* 사용 가능한 Build Targets 결정

\* 개별 Target 구축

먼저, 터미널에 입력 - 위치 이동 하기

> cd sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05

최상위 Makefile 은 **Rules.make** 라는 보완 파일과 함께 SDK 루트에 있다.

Rules.make 파일은 최상위 Makefile 과 개별 프로젝트 Makefile 에서 사용하는 변수 및 설정을 제공한다.

터미널에 아래 명령을 입력하여, Rules.make 파일을 연다

> gedit Rules.make

Rules.make 파일에서 PLATFORM 및 ARCH 와 같은 Target System 을 설명하는 변수가 있다. 또한 SDK 설치의 루트, linux-devkit 디렉토리의 위치 및 Cross-Compiler 를 가리키는 변수를 찾을 수 있다.

확인 했다면, Rules.make 에디터를 닫고,   
터미널에 아래의 명령을 입력하여, Makefile 파일을 연다

>gedit Makefile

여기에서 주의해야할 중요한 항목이 있다

1. 맨 윗줄 -include Rules.make
2. 모든 대상은 항상 모든 것을 구축하는 대신 개별 구성 요소를 작성하기 위해 호출할 수 있는 개별 대상을 표시한다.

위의 설정된 개별 구성 요소를 작성하는 방법으로,   
u-boot 및 SPL 부트 로더 빌드를 위해 아래 명령을 터미널에 입력한다

> make u-boot-spl

그러면 상당한 양의 다음 내용들이 나옴

|  |
| --- |
| Building the Linux Kernel DTBs  =====================================  make -C /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8 ARCH=arm CROSS\_COMPILE=/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86\_64-arago-linux/usr/bin/arm-linux-gnueabihf- tisdk\_am57xx-evm\_defconfig  make[1]: Entering directory '/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8'  HOSTCC scripts/basic/fixdep  HOSTCC scripts/kconfig/conf.o  SHIPPED scripts/kconfig/zconf.tab.c  SHIPPED scripts/kconfig/zconf.lex.c  SHIPPED scripts/kconfig/zconf.hash.c  HOSTCC scripts/kconfig/zconf.tab.o  HOSTLD scripts/kconfig/conf  #  # configuration written to .config  #  make[1]: Leaving directory '/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8'  make -j 4 -C /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8 ARCH=arm CROSS\_COMPILE=/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86\_64-arago-linux/usr/bin/arm-linux-gnueabihf- am57xx-beagle-x15.dtb am57xx-beagle-x15-revb1.dtb am57xx-evm.dtb am57xx-evm-reva3.dtb am571x-idk.dtb am572x-idk.dtb am571x-idk-lcd-osd.dtb am572x-idk-lcd-osd.dtb  make[1]: Entering directory '/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8'  scripts/kconfig/conf --silentoldconfig Kconfig  HOSTCC scripts/basic/bin2c  WRAP arch/arm/include/generated/asm/bitsperlong.h  WRAP arch/arm/include/generated/asm/cputime.h  WRAP arch/arm/include/generated/asm/current.h  WRAP arch/arm/include/generated/asm/emergency-restart.h  WRAP arch/arm/include/generated/asm/exec.h  WRAP arch/arm/include/generated/asm/ioctl.h  WRAP arch/arm/include/generated/asm/errno.h  WRAP arch/arm/include/generated/asm/ipcbuf.h  WRAP arch/arm/include/generated/asm/irq\_regs.h  WRAP arch/arm/include/generated/asm/local.h  WRAP arch/arm/include/generated/asm/kdebug.h  WRAP arch/arm/include/generated/asm/local64.h  WRAP arch/arm/include/generated/asm/mm-arch-hooks.h  WRAP arch/arm/include/generated/asm/msgbuf.h  WRAP arch/arm/include/generated/asm/msi.h  WRAP arch/arm/include/generated/asm/param.h  WRAP arch/arm/include/generated/asm/parport.h  WRAP arch/arm/include/generated/asm/poll.h  WRAP arch/arm/include/generated/asm/resource.h  WRAP arch/arm/include/generated/asm/preempt.h  WRAP arch/arm/include/generated/asm/rwsem.h  WRAP arch/arm/include/generated/asm/sections.h  WRAP arch/arm/include/generated/asm/segment.h  WRAP arch/arm/include/generated/asm/seccomp.h  WRAP arch/arm/include/generated/asm/sembuf.h  WRAP arch/arm/include/generated/asm/serial.h  WRAP arch/arm/include/generated/asm/siginfo.h  WRAP arch/arm/include/generated/asm/shmbuf.h  WRAP arch/arm/include/generated/asm/simd.h  WRAP arch/arm/include/generated/asm/sizes.h  WRAP arch/arm/include/generated/asm/socket.h  WRAP arch/arm/include/generated/asm/sockios.h  WRAP arch/arm/include/generated/asm/termbits.h  WRAP arch/arm/include/generated/asm/termios.h  WRAP arch/arm/include/generated/asm/timex.h  WRAP arch/arm/include/generated/asm/unaligned.h  WRAP arch/arm/include/generated/asm/trace\_clock.h  HOSTCC scripts/kallsyms  HOSTCC scripts/dtc/dtc.o  HOSTCC scripts/genksyms/genksyms.o  CC scripts/mod/empty.o  HOSTCC scripts/mod/mk\_elfconfig  HOSTCC scripts/dtc/flattree.o  CC scripts/mod/devicetable-offsets.s  MKELF scripts/mod/elfconfig.h  GEN scripts/mod/devicetable-offsets.h  HOSTCC scripts/mod/sumversion.o  HOSTCC scripts/dtc/fstree.o  SHIPPED scripts/genksyms/parse.tab.c  SHIPPED scripts/genksyms/lex.lex.c  SHIPPED scripts/genksyms/keywords.hash.c  SHIPPED scripts/genksyms/parse.tab.h  HOSTCC scripts/genksyms/parse.tab.o  HOSTCC scripts/dtc/data.o  HOSTCC scripts/mod/modpost.o  HOSTCC scripts/dtc/livetree.o  HOSTCC scripts/genksyms/lex.lex.o  HOSTCC scripts/dtc/treesource.o  HOSTCC scripts/dtc/srcpos.o  HOSTCC scripts/dtc/checks.o  HOSTCC scripts/dtc/util.o  HOSTCC scripts/conmakehash  SHIPPED scripts/dtc/dtc-lexer.lex.c  SHIPPED scripts/dtc/dtc-parser.tab.h  SHIPPED scripts/dtc/dtc-parser.tab.c  HOSTCC scripts/dtc/dtc-lexer.lex.o  HOSTLD scripts/genksyms/genksyms  HOSTCC scripts/mod/file2alias.o  HOSTCC scripts/sortextable  HOSTCC scripts/dtc/dtc-parser.tab.o  HOSTLD scripts/dtc/dtc  HOSTLD scripts/mod/modpost  DTC arch/arm/boot/dts/am57xx-beagle-x15-revb1.dtb  DTC arch/arm/boot/dts/am57xx-beagle-x15.dtb  DTC arch/arm/boot/dts/am57xx-evm.dtb  DTC arch/arm/boot/dts/am57xx-evm-reva3.dtb  DTC arch/arm/boot/dts/am571x-idk.dtb  DTC arch/arm/boot/dts/am572x-idk.dtb  DTC arch/arm/boot/dts/am571x-idk-lcd-osd.dtb  DTC arch/arm/boot/dts/am572x-idk-lcd-osd.dtb  make[1]: Leaving directory '/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8'  ===================================  Building U-boot  ===================================  make -j 4 -C /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-\* CROSS\_COMPILE=/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86\_64-arago-linux/usr/bin/arm-linux-gnueabihf- am57xx\_evm\_config  make[1]: Entering directory '/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc'  HOSTCC scripts/basic/fixdep  HOSTCC scripts/kconfig/conf.o  SHIPPED scripts/kconfig/zconf.tab.c  SHIPPED scripts/kconfig/zconf.hash.c  SHIPPED scripts/kconfig/zconf.lex.c  HOSTCC scripts/kconfig/zconf.tab.o  HOSTLD scripts/kconfig/conf  #  # configuration written to .config  #  make[1]: Leaving directory '/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc'  make -j 4 -C /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-\* CROSS\_COMPILE=/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86\_64-arago-linux/usr/bin/arm-linux-gnueabihf- DTC=/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8/scripts/dtc/dtc  make[1]: Entering directory '/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc'  scripts/kconfig/conf --silentoldconfig Kconfig  CHK include/config.h  UPD include/config.h  GEN include/autoconf.mk  GEN include/autoconf.mk.dep  GEN spl/include/autoconf.mk  CHK include/config/uboot.release  CHK include/generated/timestamp\_autogenerated.h  UPD include/generated/timestamp\_autogenerated.h  CFG u-boot.cfg  UPD include/config/uboot.release  CHK include/generated/version\_autogenerated.h  UPD include/generated/version\_autogenerated.h  CC lib/asm-offsets.s  CC arch/arm/lib/asm-offsets.s  CHK include/generated/generic-asm-offsets.h  CHK include/generated/asm-offsets.h  UPD include/generated/generic-asm-offsets.h  UPD include/generated/asm-offsets.h  HOSTCC tools/gen\_eth\_addr  HOSTCC tools/img2srec  HOSTCC tools/mkenvimage.o  LDS u-boot.lds  HOSTCC tools/os\_support.o  WRAP tools/lib/crc32.c  HOSTCC tools/aisimage.o  HOSTCC tools/atmelimage.o  WRAP tools/common/bootm.c  HOSTCC tools/default\_image.o  WRAP tools/lib/fdtdec\_common.c  WRAP tools/lib/fdtdec.c  HOSTCC tools/fit\_common.o  HOSTCC tools/fit\_image.o  HOSTCC tools/gpimage.o  HOSTCC tools/gpimage-common.o  WRAP tools/common/image-fit.c  WRAP tools/common/image.c  HOSTCC tools/image-host.o  HOSTCC tools/imagetool.o  HOSTCC tools/imximage.o  HOSTCC tools/kwbimage.o  WRAP tools/lib/md5.c  HOSTCC tools/lpc32xximage.o  HOSTCC tools/mxsimage.o  HOSTCC tools/omapimage.o  HOSTCC tools/pblimage.o  HOSTCC tools/pbl\_crc32.o  WRAP tools/lib/rc4.c  HOSTCC tools/rkcommon.o  HOSTCC tools/rkimage.o  HOSTCC tools/rksd.o  HOSTCC tools/rkspi.o  HOSTCC tools/socfpgaimage.o  WRAP tools/lib/sha1.c  WRAP tools/lib/sha256.c  WRAP tools/common/hash.c  HOSTCC tools/ublimage.o  HOSTCC tools/zynqimage.o  WRAP tools/lib/libfdt/fdt.c  WRAP tools/lib/libfdt/fdt\_ro.c  WRAP tools/lib/libfdt/fdt\_rw.c  WRAP tools/lib/libfdt/fdt\_strerror.c  WRAP tools/lib/libfdt/fdt\_wip.c  WRAP tools/lib/libfdt/fdt\_region.c  WRAP tools/lib/libfdt/fdt\_sw.c  HOSTCC tools/dumpimage.o  HOSTCC tools/mkimage.o  HOSTCC tools/proftool  HOSTCC tools/fdtgrep.o  HOSTCC tools/lib/crc32.o  HOSTCC tools/common/bootm.o  HOSTCC tools/lib/fdtdec\_common.o  HOSTCC tools/lib/fdtdec.o  HOSTCC tools/common/image-fit.o  HOSTCC tools/common/image.o  HOSTCC tools/lib/md5.o  HOSTCC tools/lib/rc4.o  HOSTCC tools/lib/sha1.o  HOSTCC tools/lib/sha256.o  HOSTCC tools/common/hash.o  HOSTCC tools/lib/libfdt/fdt.o  HOSTCC tools/lib/libfdt/fdt\_ro.o  HOSTCC tools/lib/libfdt/fdt\_rw.o  HOSTCC tools/lib/libfdt/fdt\_strerror.o  HOSTCC tools/lib/libfdt/fdt\_wip.o  HOSTCC tools/lib/libfdt/fdt\_region.o  HOSTCC tools/lib/libfdt/fdt\_sw.o  HOSTLD tools/mkenvimage  HOSTLD tools/dumpimage  HOSTLD tools/mkimage  HOSTLD tools/fdtgrep  LD arch/arm/cpu/built-in.o  CC board/ti/am57xx/board.o  CC arch/arm/cpu/armv7/cache\_v7.o  CC board/ti/common/board\_detect.o  AS arch/arm/lib/vectors.o  AS arch/arm/lib/crt0.o  AS arch/arm/lib/relocate.o  CC arch/arm/lib/bootm-fdt.o  AS arch/arm/cpu/armv7/cache\_v7\_asm.o  CC arch/arm/lib/bootm.o  CC arch/arm/cpu/armv7/cpu.o  LD board/ti/common/built-in.o  CC arch/arm/cpu/armv7/cp15.o  CC cmd/boot.o  CC arch/arm/lib/sections.o  LD board/ti/am57xx/built-in.o  CC arch/arm/lib/stack.o  CC arch/arm/cpu/armv7/syslib.o  CC cmd/bootm.o  CC common/init/board\_init.o  CC arch/arm/lib/interrupts.o  AS arch/arm/cpu/armv7/lowlevel\_init.o  CC arch/arm/cpu/armv7/omap-common/reset.o  LD common/init/built-in.o  CC common/main.o  CC arch/arm/cpu/armv7/omap-common/timer.o  CC arch/arm/lib/reset.o  CC cmd/help.o  CC common/exports.o  CC arch/arm/cpu/armv7/omap-common/utils.o  CC arch/arm/lib/cache.o  CC cmd/version.o  CC common/hash.o  CC cmd/source.o  CC arch/arm/cpu/armv7/omap-common/hwinit-common.o  CC arch/arm/lib/cache-cp15.o  CC cmd/bdinfo.o  CC common/cli\_hush.o  CC arch/arm/cpu/armv7/omap-common/clocks-common.o  AR arch/arm/lib/lib.a  CC arch/arm/lib/eabi\_compat.o  CC cmd/bootefi.o  LD arch/arm/lib/built-in.o  CC disk/part.o  CC cmd/console.o  CC cmd/echo.o  CC disk/part\_dos.o  CC arch/arm/cpu/armv7/omap-common/emif-common.o  CC cmd/elf.o  CC disk/part\_iso.o  CC cmd/exit.o  CC common/autoboot.o  CC arch/arm/cpu/armv7/omap-common/vc.o  CC cmd/ext4.o  CC common/board\_f.o  CC disk/part\_efi.o  CC arch/arm/cpu/armv7/omap-common/abb.o  CC cmd/ext2.o  CC arch/arm/cpu/armv7/omap-common/pipe3-phy.o  CC cmd/fat.o  CC common/board\_r.o  CC arch/arm/cpu/armv7/omap-common/sata.o  CC cmd/fdt.o  CC arch/arm/cpu/armv7/omap-common/omap-cache.o  CC common/board\_info.o  CC arch/arm/cpu/armv7/omap-common/boot-common.o  CC common/bootm.o  LD disk/built-in.o  CC common/bootm\_os.o  AS arch/arm/cpu/armv7/omap-common/lowlevel\_init.o  CC arch/arm/cpu/armv7/omap-common/mem-common.o  CC common/env\_attr.o  LD arch/arm/cpu/armv7/omap-common/built-in.o  CC cmd/fs.o  CC arch/arm/cpu/armv7/omap5/boot.o  CC common/env\_callback.o  CC arch/arm/cpu/armv7/omap5/hwinit.o  CC cmd/gpio.o  CC common/env\_flags.o  CC common/env\_fat.o  CC arch/arm/cpu/armv7/omap5/emif.o  CC cmd/i2c.o  CC common/fdt\_support.o  CC cmd/itest.o  CC arch/arm/cpu/armv7/omap5/sdram.o  CC arch/arm/cpu/armv7/omap5/prcm-regs.o  CC arch/arm/cpu/armv7/omap5/hw\_data.o  CC arch/arm/cpu/armv7/omap5/abb.o  CC cmd/load.o  CC arch/arm/cpu/armv7/omap5/fdt.o  CC arch/arm/cpu/armv7/omap5/dra7xx\_iodelay.o  AS arch/arm/cpu/armv7/omap5/sec\_entry\_cpu1.o  AS arch/arm/cpu/armv7/start.o  CC cmd/mem.o  CC common/miiphyutil.o  LD arch/arm/cpu/armv7/omap5/built-in.o  LD arch/arm/cpu/armv7/built-in.o  CC cmd/mii.o  LD drivers/adc/built-in.o  CC drivers/block/disk-uclass.o  CC cmd/mdio.o  CC drivers/block/dwc\_ahci.o  CC common/usb.o  CC cmd/misc.o  CC cmd/mmc.o  CC drivers/block/ahci.o  CC cmd/net.o  CC cmd/part.o  CC common/usb\_hub.o  CC drivers/core/device.o  CC cmd/pcmcia.o  CC drivers/block/libata.o  CC cmd/pxe.o  LD drivers/block/built-in.o  CC drivers/crypto/fsl/sec.o  CC common/usb\_storage.o  LD drivers/crypto/fsl/built-in.o  LD drivers/crypto/rsa\_mod\_exp/built-in.o  LD drivers/crypto/built-in.o  CC drivers/dfu/dfu.o  CC drivers/core/lists.o  CC drivers/core/root.o  CC cmd/sf.o  CC drivers/dfu/dfu\_mmc.o  CC drivers/core/uclass.o  CC common/flash.o  CC common/splash.o  CC cmd/scsi.o  CC drivers/dfu/dfu\_ram.o  CC common/menu.o  CC drivers/core/util.o  CC drivers/dfu/dfu\_sf.o  CC drivers/core/device-remove.o  CC common/env\_common.o  CC cmd/spi.o  LD drivers/dfu/built-in.o  CC cmd/time.o  CC drivers/core/simple-bus.o  CC cmd/test.o  CC common/console.o  CC cmd/usb.o  CC drivers/core/dump.o  CC cmd/ximg.o  LD drivers/core/built-in.o  LD drivers/hwmon/built-in.o  CC common/dlmalloc.o  CC drivers/input/input.o  CC cmd/spl.o  CC drivers/dma/ti-edma3.o  CC cmd/dfu.o  CC cmd/gpt.o  CC drivers/input/key\_matrix.o  LD drivers/dma/built-in.o  CC drivers/gpio/gpio-uclass.o  CC common/malloc\_simple.o  LD drivers/input/built-in.o  LD drivers/memory/built-in.o  LD drivers/misc/built-in.o  CC drivers/pcmcia/tqm8xx\_pcmcia.o  CC cmd/regulator.o  CC common/image.o  LD drivers/pcmcia/built-in.o  LD drivers/pwm/built-in.o  CC drivers/rtc/date.o  LD drivers/rtc/built-in.o  LD drivers/soc/built-in.o  LD drivers/sound/built-in.o  LD drivers/spmi/built-in.o  LD drivers/thermal/built-in.o  LD drivers/timer/built-in.o  CC cmd/nvedit.o  LD drivers/tpm/built-in.o  LD drivers/twserial/built-in.o  CC drivers/gpio/omap\_gpio.o  LD drivers/video/bridge/built-in.o  LD drivers/video/built-in.o  LD drivers/watchdog/built-in.o  LD drivers/built-in.o  CC drivers/i2c/i2c-uclass.o  CC common/image-fdt.o  LD drivers/gpio/built-in.o  CC drivers/mmc/mmc-uclass.o  CC drivers/mmc/mmc.o  CC drivers/i2c/i2c-uclass-compat.o  CC common/image-fit.o  CC cmd/disk.o  CC drivers/i2c/omap24xx\_i2c.o  LD cmd/built-in.o  CC drivers/mtd/mtdcore.o  LD drivers/i2c/built-in.o  LD drivers/mtd/onenand/built-in.o  CC drivers/mtd/spi/sf-uclass.o  CC common/memsize.o  CC drivers/mtd/spi/sf\_probe.o  CC drivers/mtd/mtd\_uboot.o  CC common/stdio.o  CC drivers/mtd/spi/spi\_flash.o  LD drivers/mtd/built-in.o  CC drivers/net/cpsw.o  CC common/cli\_simple.o  CC drivers/mmc/omap\_hsmmc.o  CC common/cli.o  CC drivers/mtd/spi/sf\_params.o  CC common/cli\_readline.o  CC drivers/mtd/spi/sf.o  LD drivers/mtd/spi/built-in.o  CC common/dfu.o  CC drivers/net/cpsw-common.o  CC common/command.o  CC common/s\_record.o  LD drivers/net/built-in.o  CC drivers/net/phy/phy.o  CC drivers/mmc/mmc\_write.o  CC common/xyzModem.o  LD drivers/mmc/built-in.o  LD drivers/pci/built-in.o  CC drivers/power/palmas.o  LD drivers/power/battery/built-in.o  LD drivers/power/fuel\_gauge/built-in.o  LD drivers/power/mfd/built-in.o  CC drivers/power/pmic/pmic-uclass.o  LD common/built-in.o  LD drivers/power/built-in.o  LD drivers/net/phy/built-in.o  CC drivers/power/regulator/regulator-uclass.o  LD drivers/rapidio/built-in.o  CC drivers/spi/spi-uclass.o  CC drivers/serial/serial-uclass.o  CC drivers/power/pmic/palmas.o  LD drivers/power/pmic/built-in.o  LD drivers/usb/common/built-in.o  CC drivers/usb/dwc3/core.o  CC drivers/serial/ns16550.o  CC drivers/power/regulator/palmas\_regulator.o  CC drivers/spi/ti\_qspi.o  LD drivers/serial/built-in.o  LD drivers/usb/emul/built-in.o  LD drivers/usb/eth/built-in.o  LD drivers/power/regulator/built-in.o  CC drivers/usb/gadget/epautoconf.o  CC drivers/usb/gadget/udc/udc-core.o  LD drivers/spi/built-in.o  CC drivers/usb/gadget/config.o  CC drivers/usb/dwc3/gadget.o  CC drivers/usb/gadget/usbstring.o  CC drivers/usb/gadget/g\_dnl.o  LD drivers/usb/gadget/udc/built-in.o  CC drivers/usb/host/xhci.o  CC drivers/usb/gadget/f\_dfu.o  CC drivers/usb/dwc3/ep0.o  CC drivers/usb/host/xhci-mem.o  LD drivers/usb/gadget/built-in.o  LD drivers/usb/musb-new/built-in.o  LD drivers/usb/musb/built-in.o  CC drivers/usb/phy/omap\_usb\_phy.o  CC drivers/usb/dwc3/dwc3-omap.o  CC drivers/usb/host/xhci-ring.o  CC drivers/usb/dwc3/ti\_usb\_phy.o  LD drivers/usb/phy/built-in.o  LD drivers/usb/ulpi/built-in.o  CC fs/fs.o  LD drivers/usb/dwc3/dwc3.o  CC drivers/usb/host/xhci-dwc3.o  LD drivers/usb/dwc3/built-in.o  CC lib/efi\_loader/efi\_image\_loader.o  CC drivers/usb/host/xhci-omap.o  CC fs/ext4/ext4fs.o  CC fs/fat/fat\_write.o  LD drivers/usb/host/built-in.o  CC lib/efi\_loader/efi\_boottime.o  CC net/checksum.o  CC net/arp.o  CC fs/ext4/ext4\_common.o  CC net/bootp.o  CC lib/efi\_loader/efi\_runtime.o  CC lib/efi\_loader/efi\_console.o  CC net/eth-uclass.o  CC lib/efi\_loader/efi\_memory.o  CC net/eth\_common.o  CC lib/efi\_loader/efi\_disk.o  CC fs/fat/file.o  CC net/net.o  LD lib/efi\_loader/built-in.o  CC lib/libfdt/fdt.o  CC fs/ext4/dev.o  LD fs/fat/built-in.o  CC fs/ext4/ext4\_write.o  CC lib/libfdt/fdt\_ro.o  CC fs/ext4/ext4\_journal.o  CC net/nfs.o  CC lib/libfdt/fdt\_rw.o  CC fs/ext4/crc16.o  LD test/built-in.o  CC test/dm/cmd\_dm.o  LD fs/ext4/built-in.o  LD fs/built-in.o  CC lib/zlib/zlib.o  CC net/ping.o  LD test/dm/built-in.o  CC net/tftp.o  CC lib/libfdt/fdt\_strerror.o  CC lib/libfdt/fdt\_sw.o  CC lib/libfdt/fdt\_wip.o  CC lib/libfdt/fdt\_empty\_tree.o  LD net/built-in.o  CC lib/crc7.o  CC lib/libfdt/fdt\_addresses.o  CC lib/libfdt/fdt\_region.o  CC lib/crc8.o  CC lib/crc16.o  CC lib/errno\_str.o  CC lib/fdtdec\_common.o  CC lib/fdtdec.o  CC lib/gunzip.o  LD lib/libfdt/built-in.o  CC lib/initcall.o  CC lib/lmb.o  CC lib/ldiv.o  CC lib/md5.o  CC lib/net\_utils.o  CC lib/qsort.o  LD lib/zlib/built-in.o  CC lib/rc4.o  CC lib/sha1.o  CC lib/sha256.o  CC lib/strmhz.o  CC lib/list\_sort.o  CC lib/hashtable.o  CC lib/errno.o  CC lib/display\_options.o  CC lib/crc32.o  CC lib/ctype.o  CC lib/div64.o  CC lib/hang.o  CC lib/linux\_compat.o  CC lib/linux\_string.o  CC lib/membuff.o  CC lib/slre.o  CC lib/string.o  CC lib/time.o  CC lib/uuid.o  CC lib/vsprintf.o  CC lib/panic.o  CC lib/strto.o  LD lib/built-in.o  CC examples/standalone/stubs.o  CC examples/standalone/hello\_world.o  LD examples/standalone/libstubs.o  LD examples/standalone/hello\_world  OBJCOPY examples/standalone/hello\_world.srec  OBJCOPY examples/standalone/hello\_world.bin  LD u-boot  OBJCOPY u-boot-nodtb.bin  OBJCOPY u-boot.srec  SYM u-boot.sym  DTC arch/arm/dts/am57xx-beagle-x15.dtb  DTC arch/arm/dts/am572x-idk.dtb  DTC arch/arm/dts/am571x-idk.dtb  SHIPPED dts/dt.dtb  MKIMAGE u-boot.img  CAT u-boot-dtb.bin  COPY u-boot.dtb  MKIMAGE u-boot-dtb.img  COPY u-boot.bin  LD spl/arch/arm/cpu/built-in.o  CC spl/board/ti/am57xx/board.o  CC spl/board/ti/common/board\_detect.o  CC spl/arch/arm/cpu/armv7/cache\_v7.o  AS spl/arch/arm/lib/vectors.o  AS spl/arch/arm/lib/crt0.o  CC spl/arch/arm/lib/spl.o  CC spl/arch/arm/lib/sections.o  AS spl/arch/arm/cpu/armv7/cache\_v7\_asm.o  CC spl/arch/arm/cpu/armv7/cpu.o  CC spl/arch/arm/lib/stack.o  LD spl/board/ti/common/built-in.o  CC spl/arch/arm/cpu/armv7/cp15.o  CC spl/arch/arm/lib/interrupts.o  CC spl/common/spl/spl.o  LD spl/board/ti/am57xx/built-in.o  CC spl/common/init/board\_init.o  CC spl/arch/arm/cpu/armv7/syslib.o  CC spl/arch/arm/lib/reset.o  LD spl/common/init/built-in.o  CC spl/common/env\_common.o  AS spl/arch/arm/cpu/armv7/lowlevel\_init.o  CC spl/common/spl/spl\_fit.o  CC spl/arch/arm/lib/cache.o  CC spl/arch/arm/cpu/armv7/omap-common/reset.o  CC spl/arch/arm/cpu/armv7/omap-common/timer.o  CC spl/arch/arm/lib/cache-cp15.o  CC spl/common/console.o  CC spl/common/spl/spl\_mmc.o  CC spl/arch/arm/cpu/armv7/omap-common/utils.o  AR spl/arch/arm/lib/lib.a  CC spl/arch/arm/lib/eabi\_compat.o  CC spl/arch/arm/cpu/armv7/omap-common/hwinit-common.o  CC spl/common/dlmalloc.o  CC spl/common/spl/spl\_fat.o  LD spl/arch/arm/lib/built-in.o  CC spl/cmd/nvedit.o  CC spl/common/spl/spl\_ext.o  CC spl/arch/arm/cpu/armv7/omap-common/clocks-common.o  CC spl/cmd/disk.o  LD spl/common/spl/built-in.o  CC spl/disk/part.o  LD spl/cmd/built-in.o  CC spl/drivers/dma/ti-edma3.o  CC spl/common/malloc\_simple.o  CC spl/disk/part\_dos.o  CC spl/common/image.o  CC spl/arch/arm/cpu/armv7/omap-common/emif-common.o  LD spl/drivers/dma/built-in.o  CC spl/drivers/gpio/omap\_gpio.o  CC spl/disk/part\_iso.o  LD spl/drivers/gpio/built-in.o  CC spl/drivers/i2c/i2c-uclass.o  CC spl/arch/arm/cpu/armv7/omap-common/vc.o  CC spl/common/image-fdt.o  CC spl/disk/part\_efi.o  CC spl/arch/arm/cpu/armv7/omap-common/abb.o  CC spl/arch/arm/cpu/armv7/omap-common/pipe3-phy.o  CC spl/drivers/i2c/i2c\_core.o  CC spl/common/memsize.o  CC spl/common/stdio.o  CC spl/arch/arm/cpu/armv7/omap-common/sata.o  CC spl/drivers/i2c/omap24xx\_i2c.o  CC spl/arch/arm/cpu/armv7/omap-common/omap-cache.o  CC spl/common/cli\_simple.o  CC spl/arch/arm/cpu/armv7/omap-common/boot-common.o  LD spl/disk/built-in.o  CC spl/drivers/mmc/mmc-uclass.o  AS spl/arch/arm/cpu/armv7/omap-common/lowlevel\_init.o  LD spl/drivers/i2c/built-in.o  CC spl/common/cli.o  CC spl/drivers/mtd/spi/spi\_spl\_load.o  CC spl/arch/arm/cpu/armv7/omap-common/mem-common.o  CC spl/drivers/mmc/mmc.o  LD spl/arch/arm/cpu/armv7/omap-common/built-in.o  CC spl/common/cli\_readline.o  CC spl/drivers/mtd/spi/sf\_probe.o  CC spl/arch/arm/cpu/armv7/omap5/boot.o  CC spl/arch/arm/cpu/armv7/omap5/hwinit.o  CC spl/drivers/mtd/spi/spi\_flash.o  CC spl/common/dfu.o  CC spl/arch/arm/cpu/armv7/omap5/emif.o  CC spl/drivers/mtd/spi/sf\_params.o  CC spl/common/command.o  CC spl/arch/arm/cpu/armv7/omap5/sdram.o  CC spl/drivers/mtd/spi/sf.o  LD spl/drivers/mtd/spi/built-in.o  CC spl/drivers/power/palmas.o  CC spl/arch/arm/cpu/armv7/omap5/prcm-regs.o  CC spl/common/s\_record.o  LD spl/drivers/power/built-in.o  CC spl/arch/arm/cpu/armv7/omap5/hw\_data.o  CC spl/drivers/power/pmic/pmic-uclass.o  CC spl/drivers/mmc/omap\_hsmmc.o  CC spl/common/xyzModem.o  LD spl/drivers/power/pmic/built-in.o  LD spl/drivers/power/regulator/built-in.o  CC spl/arch/arm/cpu/armv7/omap5/abb.o  CC spl/drivers/serial/serial.o  CC spl/arch/arm/cpu/armv7/omap5/fdt.o  LD spl/common/built-in.o  LD spl/drivers/mmc/built-in.o  LD spl/dts/built-in.o  CC spl/drivers/spi/spi.o  CC spl/fs/ext4/ext4fs.o  CC spl/drivers/serial/serial\_ns16550.o  CC spl/arch/arm/cpu/armv7/omap5/dra7xx\_iodelay.o  CC spl/drivers/spi/ti\_qspi.o  CC spl/drivers/serial/ns16550.o  CC spl/fs/ext4/ext4\_common.o  AS spl/arch/arm/cpu/armv7/omap5/sec\_entry\_cpu1.o  LD spl/arch/arm/cpu/armv7/omap5/built-in.o  AS spl/arch/arm/cpu/armv7/start.o  LD spl/arch/arm/cpu/armv7/built-in.o  LD spl/drivers/serial/built-in.o  LD spl/drivers/spi/built-in.o  CC spl/drivers/usb/phy/omap\_usb\_phy.o  CC spl/fs/ext4/dev.o  CC spl/lib/libfdt/fdt.o  CC spl/fs/ext4/ext4\_write.o  LD spl/drivers/usb/phy/built-in.o  LD spl/drivers/built-in.o  LDS spl/u-boot-spl.lds  CC spl/lib/libfdt/fdt\_ro.o  CFG spl/u-boot-spl.cfg  CC spl/fs/fat/fat\_write.o  CC spl/lib/libfdt/fdt\_rw.o  CC spl/fs/ext4/ext4\_journal.o  CC spl/fs/ext4/crc16.o  CC spl/lib/libfdt/fdt\_strerror.o  CC spl/lib/libfdt/fdt\_sw.o  CC spl/lib/libfdt/fdt\_wip.o  CC spl/lib/libfdt/fdt\_empty\_tree.o  LD spl/fs/ext4/built-in.o  CC spl/lib/libfdt/fdt\_addresses.o  CC spl/lib/libfdt/fdt\_region.o  CC spl/lib/hashtable.o  CC spl/lib/errno.o  CC spl/lib/display\_options.o  LD spl/fs/fat/built-in.o  LD spl/lib/libfdt/built-in.o  LD spl/fs/built-in.o  CC spl/lib/crc32.o  CC spl/lib/ctype.o  CC spl/lib/div64.o  CC spl/lib/hang.o  CC spl/lib/linux\_compat.o  CC spl/lib/linux\_string.o  CC spl/lib/membuff.o  CC spl/lib/slre.o  CC spl/lib/string.o  CC spl/lib/time.o  CC spl/lib/uuid.o  CC spl/lib/vsprintf.o  CC spl/lib/panic.o  CC spl/lib/strto.o  LD spl/lib/built-in.o  LD spl/u-boot-spl  OBJCOPY spl/u-boot-spl-nodtb.bin  COPY spl/u-boot-spl.bin  MKIMAGE MLO  make[1]: Leaving directory '/home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc' |

[참고] 이 Lab 에서 시간 제약으로 모든 대상을 빌드하지 않지만 make 를 입력하여 모든 구성요소를 빌드할 수도 있다

[5]

1. 현재 디렉토리에서 아래의 디렉토리로 이동한다.

>cd sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/

2. 디렉토리의 내용을 확인하기 위해 ls 를 입력한다.

>ls

[디렉토리의 내용들]

|  |
| --- |
| extra-drivers  linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8  prebuilt-images  u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc |

3. 위 목록 중 , Prebuilt-Images 디렉토리의 내용만 확인해본다

>ls prebuilt-images/

[디렉토리의 내용들]

|  |
| --- |
| am571x-idk.dtb am57xx-evm.dtb  am571x-idk-lcd-osd101t2587.dtb am57xx-evm-reva3.dtb  am571x-idk-lcd-osd.dtb MLO-am57xx-evm  am572x-idk.dtb README  am572x-idk-lcd-osd101t2587.dtb u-boot-am57xx-evm.img  am572x-idk-lcd-osd.dtb u-boot-spl.bin-am57xx-evm  am572x-idk-pru-excl-uio.dtb uEnv.txt  am57xx-beagle-x15.dtb zImage-am57xx-evm.bin  am57xx-beagle-x15-revb1.dtb |

이러한 Image 는 사전 컴파일되어 대상 장치에서 작동하도록 테스트되었다.

이것은 좋은 시작점으로 알려져 있다.

4. Linux System 을 구축할 때 작동하는 것으로 알려진 커널 구성으로 시작한 다음,

조각(구성모듈) 을 추가 및 제거하여 System 을 최적화하는 것이 유용하다.

Processor Linux SDK 에서 prebuilt-images 디렉토리 안에

pre-built kernel image 를 Build 하는데 사용된 Kernel 구성을 제공한다.

이 구성은 아래 명령을 사용하여 볼 수 있다(버전을 실제 커널 버전으로 바꾼다)

Kernel 구성을 보기 위하여, 다음 명령을 터미널에 입력한다

>ls linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8/arch/arm/configs/tisdk\_\*

그러면 아래 내용이 뜬다 (tisdk\_<machine>\_defconfig 형식의 파일 확인하면 됨)

linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8/arch/arm/configs/tisdk\_am57xx-evm\_defconfig

이 구성을 자신의 Kernel Build에 대한 기본 구성을 사용할수 있다.

5. 최신 SDK 에는 더 이상 MLO Bootloader 를 생성하는 별도의 코드 기반이 없다.

대신 이 bootloader 는 이제 u-boot Code 기반의 일부이며 u-boot 의 두 번째 Program Loader(SPL) 기능을 사용하여 생성된다.

이전의 [4]번 Lab 에서 make u-boot 명령을 사용하여 u-boot 를 컴파일했다.

아래 명령을 사용하여 u-boot 디렉토리의 내용을 보면 u-boot.img 파일뿐만 아니라 MLO 파일도 생긴것을 확인할 수 있다. 이것들은 모두 u-boot 빌드 작업 중에 빌드되었다.

터미널에 다음을 입력해보자

> ls u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc/

그러면 다음 목록이 뜬다

|  |
| --- |
| api drivers MAINTAINERS spl u-boot-dtb.bin  arch dts MAKEALL System.map u-boot-dtb.img  board examples Makefile test  **u-boot.img**  cmd fs  **MLO** TISDK-README u-boot.lds  common include net tools u-boot.map  config.mk Kbuild post u-boot u-boot-nodtb.bin  configs Kconfig README u-boot.bin u-boot.srec  disk lib scripts u-boot.cfg u-boot.sym  doc Licenses snapshot.commit u-boot.dtb |

위 이미지에서 MLO 및 u-boot.img 파일이 강조 표시되었다.

이 두 파일은 아까 [4]번 LAB에서 u-boot Build 작업 중에 만들어졌다.

이점은 우리가 동일한 u-boot Code 를 공유하여 초기 Bootloader 를 Build 할 수 있을 뿐 아니라, 완전한 기능을 갖춘 u-boot bootloader 를 Build 할 수 있다는 것이다.

6. Tree Module 이 없는 Device Drivers 는 extra-drivers 디렉토리에 있다.

이는 Kernel 구성이 단일 디렉토리인 board-support 디렉토리로 변경될 때

재구성이 필요한 항목을 그룹화하기 위해 수행된다.

아래 내용을 터미널에 입력하여 Tree Driver 를 확인하자.

> ls extra-drivers/

(내용 확인하기)

|  |
| --- |
| cmem-mod-4.12.00.00+gitAUTOINC+d6d3e84c84  cryptodev-module-1.8  debugss-module-drv-1.4.0+gitAUTOINC+0aedcabdbb  gdbserverproxy-module-drv-1.1.0+gitAUTOINC+df0b8f6f4e  ti-gc320-driver-5.0.11.p7  ti-sgx-ddk-km-1.14.3699939  uio-module-drv-2.1.0.0+gitAUTOINC+1976149588 |

[6]

이 절에서 Processor Linux SDK 의 setup.sh Script 를 사용하여, TFTP 및 NFS 를 사용하여 개발한 Board 를 구성하는 방법에 대해 설명한다. ***(겁나 설치한다. 정신바짝 차리자)***

[키포인트]

\* 설치 환경에 필요한 SW 설치

\* NFS 파일 공유 생성

\* 커널 전송을 위한 TFTP 설정

\* minicom Script 를 사용하여 EVM 구성

다음 단계는 Target Board 와의 직렬 연결이 유효한지 확인하는데 도움이 된다.

(이단계는 FTDI USB-to-Serial 어댑터를 사용하는 EVM-SK 및 Beaglebone 과 같은 Board 용으로 작성됨)

(직선형 UART 연결을 가진 Board 의 경우 /dev/ttyUSBx 에 대한 단계를 무시하고,

대신 Board 에 연결된 Linux Host 의 직렬 Port 를 사용하면 된다.)

**1. Linux Host 와 Target Board 사이에 USB 케이블을 연결한다.**[중요사항]  
AM335x EVM-SK 에서 USB 케이블을 사용하여 **USB-to-Serial 인터페이스**를 사용할 수 있다.   
(가상 머신을 사용하는 경우 AM335x EVM-SK 가 선택된 USB 장치인지 확인이 필요)  
(Windows 7 에서 VMWare 이미지를 실행하는 경우 일부 USB-to-Serial 어댑터가 Windows 7 에서 제대로 작동하지 않는다. 따라서 어댑터에서 Windows 7 을 지원한다고 명시해야함. 어댑터에 대한 추가 드라이버를 설치해야 할 수도 있음)

**2. 터미널을 열고 다음 명령을 입력**하여   
System 에서 사용 가능한 USB Serial 어댑터를 확인해보자  
> ls /dev/ttyUSB\*  
(출력되는 내용)  
  **/dev/ttyUSB0**   
 /dev/ttyUSB1 (이것은 표시되지 않을수도 있다)  
  
Board 에 사용된 FTDI Chip 은 Linux Host 에 2 개의 Serial 인터페이스를 제공한다.  
첫 번째 인터페이스는 JTAG 인터페이스이고   
두 번째 인터페이스는 Serial 콘솔 인터페이스이다.  
([참고] VMWare 를 사용하는 경우 "Future Technology Devices" 를  
 VMWare 이미지로 전달해야할 수 있음)

3. Board 의 직렬 콘솔을 **/dev/ttyUSB0** 으로 열고자 한다

직렬 연결 전에, **터미널에 입력**하여, minicom 을 설치해야 한다.  
> sudo apt install minicom

설치가 완료되었다면, 이제 직렬포트를 설정하자. 다음을 터미널에 입력한다 :  
> **sudo** minicom **-w -s**

|  |  |
| --- | --- |
| (그러면 이러한 창이 뜬다)  +-----[configuration]------+  | Filenames and paths |  | File transfer protocols |  | Serial port setup |  | Modem and dialing |  | Screen and keyboard |  | Save setup as dfl |  | Save setup as.. |  | Exit |  | Exit from Minicom |  +--------------------------+ | (아니면 이러한 창이 뜬다)  Welcome to minicom 2.7    OPTIONS: I18n  Compiled on Feb 7 2016, 13:37:27.  Port **/dev/tty8**, 18:20:36    Press CTRL-A Z for help on special keys  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  /dev/ttyUSB0 또는 /dev/ttyUSB1 로 뜬다면, **다음 LAB 을 진행**하면 된다. 그러나 나는, 참 재수없게도 -\_- **/dev/tty8** 이기 때문에 , **따로 시리얼 포트 설정을 해주어야 한다** |

|  |  |
| --- | --- |
| **시리얼 포트 설정**  **1**. minicom 만 입력시 : CTRL-A Z 를 누른 후 O 를 눌러서 cOnfiguration 메뉴로 들어오거나, 직접 CTRL-A O 를 누르면 설정 메뉴로 들어갈수 있다.  **2**.minicom -s -w 입력시 : 바로 설정메뉴가 뜬다.  **3**.설정메뉴에서 (세번째) Serial port setup 를 선택한다  (그러면 다음이 뜬다)  +-----------------------------------------------------------------------+  | A - Serial Device : /dev/tty8 |  | B - Lockfile Location : /var/lock |  | C - Callin Program : |  | D - Callout Program : |  | E - Bps/Par/Bits : 115200 8N1 |  | F - Hardware Flow Control : Yes |  | G - Software Flow Control : No |  | |  | Change which setting? |  +-----------------------------------------------------------------------+  **A** 키를 눌러 이동한 뒤 =>  **/dev/tty8** 부분을 - **/dev/ttyUSB0** 로 수정한다   |  | | --- | | 주의!  (단, 위에서 **터미널을 열고 다음 명령을 입력하여 System 에서 사용 가능한 USB Serial 어댑터를 확인한 바와 같이,**   > ls /dev/ttyUSB\* (출력되는 내용)   **/dev/ttyUSB0** <<== 이것을 확인했던 것으로 수정해야 한다. ) |   설정을 완료했으므로, 저장해야 한다. 따라서, **Save setup as dfl** 을 선택하여 엔터친다.  (현재 설정한 값을 **기본값**으로 지정하는 것이다)  설정후 나가는 방법 : ctrl A Z 누르고 , Q (리셋하지 않고 미니컴 나가기)  그리고 다시 minicom을 실행하여 맞게 설정이 되었는지 확인하고 , 다음 LAB을 진행한다.  (이 아래 부터는 동일하게 뜰 것이다) |

|  |
| --- |
| Welcome to minicom 2.7    OPTIONS: I18n  Compiled on Feb 7 2016, 13:37:27.  Port **/dev/ttyUSB0**, 18:20:37    Press CTRL-A Z for help on special keys |

4. 이제 Board 의 전원을 켰을때, Board 에 따라 부팅 시퀀스 혹은 간단히 "ccccc" 가 minicom 창에 출력된 것을 **확인**할 수 있다. 이는 직렬 연결이 예상대로 작동함을 의미한다.   
(또한, 이 콘솔을 백그라운드에서 실행 상태로 둘 수 있다.)

([참고] Beaglebone 의 경우 USB 케이블이 연결되면 Board 의 전원이 켜진다. 예상되는 출력을 제공하는 리셋 버튼을 눌러 보드를 리셋할 수 있다.)

5. **minicom 을 종료**한다

6. **다음을 터미널에 입력**하여, 설치한다

> cd /home/sitara/ti-processor-sdk-linux-<machine>-<version>

>  **sudo** ./setup.sh

|  |
| --- |
| Are you running this script using sudo? The detected username is 'root'.  Verify and enter your Linux username below  [ root ]  요게 뜨면, **username** 입력하세요  > yunk  (성공적으로 확인될 때 나타나는 메세지)  User 'yunk' is already apart of the 'dialout' group |

(패키지 스크립트 설치)

|  |
| --- |
| setup package script  This script will make sure you have the proper host support packages installed  This script requires administrator priviliges (sudo access) if packages are to be installed.  -------------------------------------------------------------------------------  System has required packages!  --------------------------------------------------------------------------------  Package verification and installation successfully completed **(성공~!)**  --------------------------------------------------------------------------------  In which directory do you want to install the target filesystem?(if this directory does not exist it will be created)  [ /home/{username}/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/**targetNFS** ] |

경로가 위와 같이 나온다면, **Enter ~!**

(타겟 파일시스템, 추출하기 및 관리자 권한 경고)

(권한은 깔끔히 무시해줘도 된다. sudo 입력하여 슈퍼유저임을 입증했기 때문이다)

|  |
| --- |
| This step will extract the target filesystem to /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/targetNFS  Note! This command requires you to have administrator priviliges (sudo access)  on your host.  Press return to continue |

위와 같이 나온다면, **Enter ~!**

(SDK 인스톨 및 관리자 권한 경고)

(권한은 깔끔히 무시해줘도 된다)

|  |
| --- |
| Successfully extracted tisdk-rootfs-image-am57xx-evm.tar.xz to  /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/targetNFS already exists  WARNING! Keeping the previous filesystem may cause compatibility problems if you are upgrading the SDK  --------------------------------------------------------------------------------  This step will set up the SDK to install binaries in to:  /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/targetNFS/home/root/am57xx-evm  The files will be available from /home/root/am57xx-evm on the target.  This setting can be changed later by editing Rules.make and changing the  EXEC\_DIR or DESTDIR variable (depending on your SDK).  Press return to continue |

위와 같이 나온다면, **Enter ~!**

참고] 루트 파일 시스템의 추출 시간을 절약하기 위해 파일 시스템을 이미 추출하였을 수 있다. 이 경우 기존 파일 시스템의 이름을 바꿀 것인지, 파일 시스템을 덮어 쓸 것인지, 파일 시스템 추출을 건너 뛸 것인지 묻는 메시지가 나타난다. **적절히 선택할 것..**

(r) rename existing filesystem (o) overwrite existing filesystem (s) skip filesystem extraction

(setup Script는 Rules.make 에 Binary설치를 위한 기본 경로가 구성되었음을 알려준다. 설정 성공했고, 방금 작성한 NFS 파일 시스템을 /etc/exports 파일에 추가하려면, 관리자 권한 필요하다 또 묻는다)

|  |
| --- |
| Rules.make edited successfully..  --------------------------------------------------------------------------------  This step will export your target filesystem for NFS access.  Note! This command requires you to have administrator priviliges (sudo access)  on your host.  Press return to continue |

위와 같이 나온다면, **Enter ~!**

(이로써, NFS 를 통해 대상 장치를 부팅하는 루트 파일 시스템으로 사용할 수 있게 됨

그리고 TFTP 루트 디렉토리를 어디로 할꺼냐 물음)

|  |
| --- |
| [ ok ] Stopping nfs-kernel-server (via systemctl): nfs-kernel-server.service. System 중지  [ ok ] Starting nfs-kernel-server (via systemctl): nfs-kernel-server.service. NFS 서버를 시작  --------------------------------------------------------------------------------File System 이 export 됨  Which directory do you want to be your tftp root directory?(if this directory does not exist it will be created for you)  [ /tftpboot ] |

디폴트 사용 할것이므로.. 그냥 **Enter~!**

|  |
| --- |
| --------------------------------------------------------------------------------  This step will set up the tftp server in the /tftpboot directory.  Note! This command requires you to have administrator priviliges (sudo access)  on your host.  Press return to continue  This step will set up minicom (serial communication application) for  SDK development  For boards that contain a USB-to-Serial converter on the board such as:  \* BeagleBone  \* Beaglebone Black  \* AM335x EVM-SK  \* AM57xx EVM  \* K2H, K2L, and K2E EVMs  the port used for minicom will be automatically detected. By default Ubuntu  will not recognize this device. Setup will add a udev rule to  /etc/udev/ so that from now on it will be recognized as soon as the board is  plugged in.  For other boards, the serial will default to /dev/ttyS0. Please update based  on your setup.  --------------------------------------------------------------------------------  NOTE: If your using any of the above boards simply hit enter and the correct port will be determined automatically at a later step. For all other boards select the serial port  that the board is connected to.  Which serial port do you want to use with minicom?  [ /dev/ttyS0 ] |

위와 같이 나온다면, **Enter ~!**

|  |
| --- |
| Copied existing /home/yunk/.minirc.dfl to /home/yunk/.minirc.dfl.old  Configuration saved to /home/yunk/.minirc.dfl. You can change it further from inside  minicom, see the Software Development Guide for more information.  --------------------------------------------------------------------------------  This step will set up the u-boot variables for booting the EVM.  --------------------------------------------------------------------------------  Autodetected the following ip address of your host, correct it if necessary HOST IP 입력  [ 192.168.0.12  192.168.0.22 ] |

(HOST System 에서 Image 를 부팅하기 위해 u-boot 를 구성하는데 필요하다. 기본 IP 주소를 사용하자)

위와 같이 나온다면, **Enter ~!**

(이제 리눅스 커널을 얻을 수 있는 곳을 선택하게 될 것이다. 이 실습에서는 **TFTP** 에서 부팅하도록 구성할 것이다.)

|  |
| --- |
| Select Linux kernel location:  1: TFTP  2: SD card  [ 1 ] |

1 입력 후, **Enter ~!**

(다음은 루트 파일 시스템의 출처를 묻는다. 이 실습에서는 **NFS** 를 사용한다.)

|  |
| --- |
| Select root file system location:  1: NFS  2: SD card  [ 1 ] |

1 입력 후, **Enter ~!**

(이제 /tftpboot 디렉토리에서 사용할 수 있는 Kernel Image 목록이 표시된다. uImage-am335x-evm.bin 의 TFTP Image 에 기본 Image 를 사용해야 한다)

|  |
| --- |
| Available kernel images in /tftproot:  am571x-idk-lcd-osd.dtb  am571x-idk-lcd-osd101t2587.dtb  am571x-idk.dtb  am572x-idk-lcd-osd.dtb  am572x-idk-lcd-osd101t2587.dtb  am572x-idk-pru-excl-uio.dtb  am572x-idk.dtb  am57xx-beagle-x15-revb1.dtb  am57xx-beagle-x15.dtb  am57xx-evm-reva3.dtb  am57xx-evm.dtb  zImage-am57xx-evm.bin  Which kernel image do you want to boot from TFTP?  **[ zImage-am57xx-evm.bin ]** 헉.. zlmage 이다.. uImage 가 아니다  Would you like to create a minicom script with the above parameters (y/n)?  [ y ] |

y 입력 후, **Enter ~!**

|  |
| --- |
| Successfully wrote /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/bin/setupBoard.minicom  Detecting connection to board... /dev/ttyUSB0  --------------------------------------------------------------------------------  Would you like to run the setup script now (y/n)?  Please connect the ethernet cable as described in the Quick Start Guide.  Once answering 'y' on the prompt below, you will have 300 seconds to connect  the board and power cycle it before the setup times out  After successfully executing this script, your EVM will be set up. You will be  able to connect to it by executing 'minicom -w' or if you prefer a windows host  you can set up Tera Term as explained in the Software Developer's Guide.  If you connect minicom or Tera Term and power cycle the board Linux will boot.  [ y ] |

위와 같이 나온다면, **그냥 Enter ~!   
(수동으로 Board 를 재부팅하자.)**

|  |
| --- |
| Device /dev/ttyUSB0 is locked.  /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  You can manually run minicom in the future with this setup script using: minicom -S ./bin/setupBoard.minicom  -------------------------------------------------------------------------------  TISDK setup completed!  Please continue reading the Software Developer's Guide for more information on  how to develop software on the EVM |

**TISDK setup completed! 완료! 이제 보드를 재부팅하여 , 다음 LAB을 수행하자**

|  |
| --- |
| **[에러발생시 참고사항]**  [1] 다음과 같이 셋업 실패 뜨고 중지될때 :  Copied existing /home/beacon/.minirc.dfl to /home/beacon/.minirc.dfl.old  tee: /home/beacon/.minirc.dfl: Permission denied  Failed setup, aborting..  Failed setup, aborting..  **보드의 EVM 선이 컴의 USB포트에 잘 연결되었는지 확인하고,  터미널에 다음을 다시 입력 > sudo ./setup.sh**  [2] 다음과 같이, 설치 중지될때 :  E: Package 'uboot-mkimage' has no installation candidate  **터미널에 다음을 입력하여 설치 : > sudo apt-get install u-boot-tools**  **그리고 다시 셋업을 실행합니다(sudo꼭 넣기) : > sudo ./setup.sh** |

7. Target Device 가 이제 SDK 설치의 targetNFS 디렉토리에 있는 NFS 루트 파일 시스템을 사용하여, HOST System 의 TFTP Server 에서 Linux Kernel 을 부팅한다.

다음 명령을 사용하여 장치가 NFS 에서 부팅되었는지 확인할 수 있다:

7.1. ENTER 를 눌러 Arago Project 프롬프트를 보고 root 를 사용하여 로그인한다.

7.2. cat /proc/cmdline 출력 결과를 /dev/nfs 로 표시하고 nfsroot 를 방금 작성한 NFS 공유로 구성해야 한다.

(mount 명령을 사용하여 루트 파티션이 NFS 를 통해 mount 되었는지 확인할 수도 있다.)

8. 이제 System 이 구성되었다.

언제나 setup.sh Script 를 다시 실행하여 Board 구성을 변경할 수 있다.

9. u-boot 환경을 다시 기본값으로 재설정하려면 minicom 에서 아래를 수행한다:

9.1. root 를 사용하여 Board 에 로그인한다.

9.2. init 6 을 사용하여 보드를 재부팅한다.

9.3. 자동 부팅을 중지하기 위해 아무 키나 누르라는 메시지가 나타나면 부팅 프로세스를 중지하고 u-boot 프롬프트로 들어간다.

9.4. env default -f -a 를 사용하여 기본 부트 환경을 재설정한다.

9.5. 이제 boot 명령을 사용하여 기본 부팅 인수로 Board 를 부팅할 수 있다.

이러한 변경 사항은 다음 부팅때 손실된다.

영구히 변경하려면 uEnv.txt 파일을 삭제하거나 setup.sh Script 를 다시 실행해야 한다.

[7]

Linux 명령줄을 사용하여 예제 프로그램을 사용하는 방법과,

크로스 컴파일을 쉽게하기 위해 예제에서 사용되는 몇 가지 기술을 살펴보자

[키포인트]

\* 예제에는 CCS / Eclipse Project 파일이 있다.

\* 명령행에서 예제 빌드하기

\* 실행 파일이 ARM 용으로 Build 되었는지 확인

1. Makefiles 를 사용하여 개별 예제를 다시 Build 할 수 있다.

예로 arm-benchmarks 예제를 다시 작성하려면 **Linux 터미널을 열고 아래로 이동**한다:

> cd /home/sitara/ti-processor-sdk-linux-<machine>-<version>/example-applications/ arm-benchmarks-<version>

2. 아래 명령을 **터미널에 입력**하자

> make clean

> make

3. Build 가 끝나면, file 명령을 사용하여 실행 파일이 ARM Device 용 Build 인지 확인해보자.

다음을 **터미널에 입력**하자

> file dhrystone/Release/dhrystone

(그러면, 이러한 메세지가 나온다)

|  |
| --- |
| dhrystone/Release/dhrystone: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=c6eb2ac26e8039fb094c5abb8090de5180d2db60, not stripped |

출력은 ARM 장치용 dhrystone 실행 파일 임을 확인 할 수 있다.

4. 개별 benchmarks 에는 또한 CCS / Eclipse Project File 이 존재하며 CCS 를 사용하여 이러한 Project 를 구축할 수 있다:

다음을 **터미널에 입력**하자

> cd dhrystone

> ls -la

(그러면, 이러한 메세지가 나온다)

|  |
| --- |
| total 84  drwxr-xr-x 4 yunk yunk 4096 5월 22 16:04 .  drwxr-xr-x 6 yunk yunk 4096 5월 22 16:06 ..  -rwxr-xr-x 1 yunk yunk 11275 5월 18 16:15 **.cproject**  drwxrwxr-x 2 yunk yunk 4096 5월 22 16:04 Debug  -rw-r--r-- 1 yunk yunk 14092 5월 18 16:15 dhry\_1.c  -rw-r--r-- 1 yunk yunk 6968 5월 18 16:15 dhry\_2.c  -rwxr-xr-x 1 yunk yunk 20245 5월 18 16:15 dhry.h  -rw-r--r-- 1 yunk yunk 5429 5월 18 16:15 Makefile  -rw-r--r-- 1 yunk yunk 2392 5월 18 16:15 **.project**  drwxrwxr-x 2 yunk yunk 4096 5월 22 16:04 Release |

.project 와 .cproject 와 같은 파일이 보여짐을 확인하면 된다.

이 파일을 사용하면 **CCS** 를 사용하여 dhrystone Project 를 Build 할 수 있다

[8]

CCS / Eclipse Project 파일을 사용하여 예제를 CCS 로 가져와 Build 하는 방법 터득하기

[키포인트]

\* 예제에서 가져올 수 있는 CCS / Eclipse Project 파일이 있다.

\* CCSv6 를 사용하여 예제 프로그램을 컴파일 할 수 있다.

1. CCS를 켠다

2. Project Explorer 창에서 마우스오른쪽클릭(팝업메뉴) > Import > Import > General > **Existing Projects into Workspace** 옵션 선택한다.

3. 그러면, **Import Projects 창**이 나온다. 여기에서,  
Browse 클릭 > /home/yunk/sitara/ti-어쩌구저쩌구/**example-applications** 폴더 - OK

그러면, 발견된 프로젝트들이 나열되어 채워진다. 채워진 후 목록에서,

dhrystone, linpack, mem\_util, oprofile\_example, whetstone 만 체크하여 > Finish 클릭

이 예제에서는 mem\_util 만 사용해본다

4. mem\_util 마우스오른쪽클릭(팝업메뉴) 한 후, 사용할 Build 구성을 선택한다.

\* Release Build 의 경우: Build Configurations -> Set Active -> Release

\* Debug Build 의 경우: Build Configurations -> Set Active -> Debug

그리고, 맨위의 메뉴줄 에서 Project > Build Project 선택 하여, **프로젝트를 빌드**한다

이때, ARM 용으로 Compile 된 것은, **[arm/le] 로 표시**된다

(참고 : Project -> Build All 을 사용하여 Project Explorer 의 모든 Project 를 Build 할 수 있음)

5. mem\_util 의 Debug 폴더내 (벌레모양아이콘의) mem\_util 내용을 보면,

=> ELF 내용을 볼 수 있다.

(elf명령어로 확인하려면, readelf -S 입력)

4.1. ELF 섹션

아래 ELF 파일은 34개의 섹션이 존재하고   
 .interp 라는 이름의 섹션은 VMA가 00010134, 오프셋은 00000134이다.

|  |
| --- |
| Sections:  Idx Name Size VMA LMA File off Algn  0 .interp 00000019 00010134 00010134 00000134 2\*\*0  CONTENTS, ALLOC, LOAD, READONLY, DATA  1 .note.ABI-tag 00000020 00010150 00010150 00000150 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  2 .note.gnu.build-id 00000024 00010170 00010170 00000170 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  3 .hash 00000094 00010194 00010194 00000194 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  4 .dynsym 00000120 00010228 00010228 00000228 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  5 .dynstr 0000009f 00010348 00010348 00000348 2\*\*0  CONTENTS, ALLOC, LOAD, READONLY, DATA  6 .gnu.version 00000024 000103e8 000103e8 000003e8 2\*\*1  CONTENTS, ALLOC, LOAD, READONLY, DATA  7 .gnu.version\_r 00000020 0001040c 0001040c 0000040c 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  8 .rel.dyn 00000018 0001042c 0001042c 0000042c 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  9 .rel.plt 00000078 00010444 00010444 00000444 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  10 .init 0000000c 000104bc 000104bc 000004bc 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, CODE  11 .plt 000000c8 000104c8 000104c8 000004c8 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, CODE  12 .text 00000508 00010590 00010590 00000590 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, CODE  13 .fini 00000008 00010a98 00010a98 00000a98 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, CODE  14 .rodata 00000134 00010aa0 00010aa0 00000aa0 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  15 .ARM.exidx 00000008 00010bd4 00010bd4 00000bd4 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  16 .eh\_frame 00000004 00010bdc 00010bdc 00000bdc 2\*\*2  CONTENTS, ALLOC, LOAD, READONLY, DATA  17 .init\_array 00000004 00020be0 00020be0 00000be0 2\*\*2  CONTENTS, ALLOC, LOAD, DATA  18 .fini\_array 00000004 00020be4 00020be4 00000be4 2\*\*2  CONTENTS, ALLOC, LOAD, DATA  19 .jcr 00000004 00020be8 00020be8 00000be8 2\*\*2  CONTENTS, ALLOC, LOAD, DATA  20 .dynamic 000000e8 00020bec 00020bec 00000bec 2\*\*2  CONTENTS, ALLOC, LOAD, DATA  21 .got 0000004c 00020cd4 00020cd4 00000cd4 2\*\*2  CONTENTS, ALLOC, LOAD, DATA  22 .data 00000008 00020d20 00020d20 00000d20 2\*\*2  CONTENTS, ALLOC, LOAD, DATA  23 .bss 0000000c 00020d28 00020d28 00000d28 2\*\*3  ALLOC  24 .comment 0000002d 00000000 00000000 00000d28 2\*\*0  CONTENTS, READONLY  25 .ARM.attributes 00000035 00000000 00000000 00000d55 2\*\*0  CONTENTS, READONLY  26 .debug\_aranges 000000d0 00000000 00000000 00000d90 2\*\*3  CONTENTS, READONLY, DEBUGGING  27 .debug\_info 000007c3 00000000 00000000 00000e60 2\*\*0  CONTENTS, READONLY, DEBUGGING  28 .debug\_abbrev 00000271 00000000 00000000 00001623 2\*\*0  CONTENTS, READONLY, DEBUGGING  29 .debug\_line 00000a80 00000000 00000000 00001894 2\*\*0  CONTENTS, READONLY, DEBUGGING  30 .debug\_frame 0000007c 00000000 00000000 00002314 2\*\*2  CONTENTS, READONLY, DEBUGGING  31 .debug\_str 0000d160 00000000 00000000 00002390 2\*\*0  CONTENTS, READONLY, DEBUGGING  32 .debug\_loc 000000f3 00000000 00000000 0000f4f0 2\*\*0  CONTENTS, READONLY, DEBUGGING  33 .debug\_ranges 00000068 00000000 00000000 0000f5e8 2\*\*3  CONTENTS, READONLY, DEBUGGING  34 .debug\_macro 000037f8 00000000 00000000 0000f650 2\*\*0  CONTENTS, READONLY, DEBUGGING |

4.2. readelf -l

|  |
| --- |
| Program Header:  0x70000001 off 0x00000bd4 vaddr 0x00010bd4 paddr 0x00010bd4 align 2\*\*2  filesz 0x00000008 memsz 0x00000008 flags r--  PHDR off 0x00000034 vaddr 0x00010034 paddr 0x00010034 align 2\*\*2  filesz 0x00000100 memsz 0x00000100 flags r-x  INTERP off 0x00000134 vaddr 0x00010134 paddr 0x00010134 align 2\*\*0  filesz 0x00000019 memsz 0x00000019 flags r--  LOAD off 0x00000000 vaddr 0x00010000 paddr 0x00010000 align 2\*\*16  filesz 0x00000be0 memsz 0x00000be0 flags r-x  LOAD off 0x00000be0 vaddr 0x00020be0 paddr 0x00020be0 align 2\*\*16  filesz 0x00000148 memsz 0x00000154 flags rw-  DYNAMIC off 0x00000bec vaddr 0x00020bec paddr 0x00020bec align 2\*\*2  filesz 0x000000e8 memsz 0x000000e8 flags rw-  NOTE off 0x00000150 vaddr 0x00010150 paddr 0x00010150 align 2\*\*2  filesz 0x00000044 memsz 0x00000044 flags r--  STACK off 0x00000000 vaddr 0x00000000 paddr 0x00000000 align 2\*\*4  filesz 0x00000000 memsz 0x00000000 flags rw- |

4.3. readelf -s

프로그램의 내용 하나하나를 테이블로 묶어서 관리하는 것이다. 때문에 여러 소스 코드를 묶을 때 extern 처리나 static 를 구분할 수 있다.

(명령어 실행시, Type에서 FUNC이면 함수, FILE이면 파일을 나타낸다.)

|  |
| --- |
| SYMBOL TABLE:  00010134 l d .interp 00000000 .interp  00010150 l d .note.ABI-tag 00000000 .note.ABI-tag  00010170 l d .note.gnu.build-id 00000000 .note.gnu.build-id  00010194 l d .hash 00000000 .hash  00010228 l d .dynsym 00000000 .dynsym  00010348 l d .dynstr 00000000 .dynstr  000103e8 l d .gnu.version 00000000 .gnu.version  0001040c l d .gnu.version\_r 00000000 .gnu.version\_r  0001042c l d .rel.dyn 00000000 .rel.dyn  00010444 l d .rel.plt 00000000 .rel.plt  000104bc l d .init 00000000 .init  000104c8 l d .plt 00000000 .plt  00010590 l d .text 00000000 .text  00010a98 l d .fini 00000000 .fini  00010aa0 l d .rodata 00000000 .rodata  00010bd4 l d .ARM.exidx 00000000 .ARM.exidx  00010bdc l d .eh\_frame 00000000 .eh\_frame  00020be0 l d .init\_array 00000000 .init\_array  00020be4 l d .fini\_array 00000000 .fini\_array  00020be8 l d .jcr 00000000 .jcr  00020bec l d .dynamic 00000000 .dynamic  00020cd4 l d .got 00000000 .got  00020d20 l d .data 00000000 .data  00020d28 l d .bss 00000000 .bss  00000000 l d .comment 00000000 .comment  00000000 l d .ARM.attributes 00000000 .ARM.attributes  00000000 l d .debug\_aranges 00000000 .debug\_aranges  00000000 l d .debug\_info 00000000 .debug\_info  00000000 l d .debug\_abbrev 00000000 .debug\_abbrev  00000000 l d .debug\_line 00000000 .debug\_line  00000000 l d .debug\_frame 00000000 .debug\_frame  00000000 l d .debug\_str 00000000 .debug\_str  00000000 l d .debug\_loc 00000000 .debug\_loc  00000000 l d .debug\_ranges 00000000 .debug\_ranges  00000000 l d .debug\_macro 00000000 .debug\_macro  00000000 l df \*ABS\* 00000000 /home/tcwg-buildslave/workspace/tcwg-make-release/label/tcwg-x86\_64-ex40/target/arm-linux-gnueabihf/\_build/builds/x86\_64-unknown-linux-gnu/arm-linux-gnueabihf/glibc.git~release-2.21-master/csu/abi-note.o  00010150 l .note.ABI-tag 00000000 $d  00000000 l df \*ABS\* 00000000 /home/tcwg-buildslave/workspace/tcwg-make-release/label/tcwg-x86\_64-ex40/target/arm-linux-gnueabihf/\_build/builds/x86\_64-unknown-linux-gnu/arm-linux-gnueabihf/glibc.git~release-2.21-master/csu/start.o  00010590 l .text 00000000 $t  00010bd4 l .ARM.exidx 00000000 $d  000105b4 l .text 00000000 $d  0000000c l .debug\_aranges 00000000 $d  00000000 l df \*ABS\* 00000000 init.c  00010aa0 l .rodata 00000000 $d  00020d20 l .data 00000000 $d  00000000 l df \*ABS\* 00000000 /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86\_64-arago-linux/usr/bin/../arm-linux-gnueabihf/libc/usr/lib/arm-linux-gnueabihf/5.3.1/crti.o  000105c0 l .text 00000000 $a  000105c0 l F .text 00000000 call\_weak\_fn  000105dc l .text 00000000 $d  000104bc l .init 00000000 $a  00010a98 l .fini 00000000 $a  00000044 l .debug\_aranges 00000000 $d  00000000 l df \*ABS\* 00000000 /home/yunk/sitara/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86\_64-arago-linux/usr/bin/../arm-linux-gnueabihf/libc/usr/lib/arm-linux-gnueabihf/5.3.1/crtn.o  000104c4 l .init 00000000 $a  00010a9c l .fini 00000000 $a  000000b4 l .debug\_aranges 00000000 $d  00000000 l df \*ABS\* 00000000 crtstuff.c  00020be8 l O .jcr 00000000 \_\_JCR\_LIST\_\_  000105e5 l F .text 00000000 deregister\_tm\_clones  000105e4 l .text 00000000 $t  00010604 l .text 00000000 $d  00010608 l .text 00000000 $t  00010609 l F .text 00000000 register\_tm\_clones  00020d24 l .data 00000000 $d  00010635 l F .text 00000000 \_\_do\_global\_dtors\_aux  00020d30 l O .bss 00000001 completed.9244  00020be4 l .fini\_array 00000000 $d  00020be4 l O .fini\_array 00000000 \_\_do\_global\_dtors\_aux\_fini\_array\_entry  0001064d l F .text 00000000 frame\_dummy  00020be0 l .init\_array 00000000 $d  00020be0 l O .init\_array 00000000 \_\_frame\_dummy\_init\_array\_entry  00020d30 l .bss 00000000 $d  00000000 l df \*ABS\* 00000000 mem\_util.c  00010aa4 l .rodata 00000000 $d  00010670 l .text 00000000 $a  00000010 l .debug\_frame 00000000 $d  00000000 l df \*ABS\* 00000000 elf-init.c  00010a54 l .text 00000000 $t  00010a8c l .text 00000000 $d  00010a94 l .text 00000000 $t  00000048 l .debug\_frame 00000000 $d  00000000 l df \*ABS\* 00000000 crtstuff.c  00010bdc l .eh\_frame 00000000 $d  00010bdc l O .eh\_frame 00000000 \_\_FRAME\_END\_\_  00020be8 l .jcr 00000000 $d  00020be8 l O .jcr 00000000 \_\_JCR\_END\_\_  00000000 l df \*ABS\* 00000000  00020be4 l .init\_array 00000000 \_\_init\_array\_end  00020bec l O .dynamic 00000000 \_DYNAMIC  00020be0 l .init\_array 00000000 \_\_init\_array\_start  00020cd4 l O .got 00000000 \_GLOBAL\_OFFSET\_TABLE\_  000104c8 l .plt 00000000 $a  000104d8 l .plt 00000000 $d  000104dc l .plt 00000000 $a  00010a95 g F .text 00000002 \_\_libc\_csu\_fini  00000000 w \*UND\* 00000000 \_ITM\_deregisterTMCloneTable  00020d20 w .data 00000000 data\_start  00000000 F \*UND\* 00000000 printf@@GLIBC\_2.4  00020d28 g .bss 00000000 \_\_bss\_start\_\_  00000000 F \*UND\* 00000000 fflush@@GLIBC\_2.4  00020d34 g .bss 00000000 \_bss\_end\_\_  00000000 F \*UND\* 00000000 tolower@@GLIBC\_2.4  00020d28 g .data 00000000 \_edata  00010a98 g F .fini 00000000 \_fini  00020d34 g .bss 00000000 \_\_bss\_end\_\_  00020d28 g O .bss 00000004 stderr@@GLIBC\_2.4  00020d20 g .data 00000000 \_\_data\_start  00000000 F \*UND\* 00000000 \_\_libc\_start\_main@@GLIBC\_2.4  00000000 F \*UND\* 00000000 strerror@@GLIBC\_2.4  00000000 w \*UND\* 00000000 \_\_gmon\_start\_\_  00000000 F \*UND\* 00000000 open@@GLIBC\_2.4  00020d24 g O .data 00000000 .hidden \_\_dso\_handle  00000000 F \*UND\* 00000000 exit@@GLIBC\_2.4  00010aa0 g O .rodata 00000004 \_IO\_stdin\_used  00000000 F \*UND\* 00000000 strtoul@@GLIBC\_2.4  00000000 F \*UND\* 00000000 mmap@@GLIBC\_2.4  00000000 F \*UND\* 00000000 fprintf@@GLIBC\_2.4  00010a55 g F .text 00000040 \_\_libc\_csu\_init  00000000 F \*UND\* 00000000 \_\_errno\_location@@GLIBC\_2.4  00020d34 g .bss 00000000 \_end  00010591 g F .text 00000000 \_start  00020d34 g .bss 00000000 \_\_end\_\_  00020d2c g O .bss 00000004 stdout@@GLIBC\_2.4  00020d28 g .bss 00000000 \_\_bss\_start  00010670 g F .text 000003e4 main  00000000 F \*UND\* 00000000 munmap@@GLIBC\_2.4  00000000 w \*UND\* 00000000 \_Jv\_RegisterClasses  00020d28 g O .data 00000000 .hidden \_\_TMC\_END\_\_  00000000 w \*UND\* 00000000 \_ITM\_registerTMCloneTable  00000000 F \*UND\* 00000000 abort@@GLIBC\_2.4  000104bc g F .init 00000000 \_init  00000000 F \*UND\* 00000000 close@@GLIBC\_2.4 |

[9]

CCS 에서 새 프로젝트를 만들고 ARM Device 용으로 Cross Compile 하는 방법에 대해 실습

[키포인트]

\* 새로운 Cross Compile Project를 만드는 방법

\* Project 에 Source 를 추가하고 Build 하는 방법

1. **Select a wizard 창** : New Project > C/C++ 선택 > C Project 선택 > **Next**

2. **C Project 창** :   
Project name : **helloworld** 입력,   
Project type: **Executable -> Empty Project** 선택,   
Toolchains: **Cross GCC** 선택   
> **Next**

3. **Select Configurations 창** : Debug 와 Release ………………….모두 선택 되었는지 확인 > **Next**

(여기에서 기본 Debug 및 Release 구성을 사용하거나, 원하는 경우 추가/제거 할 수 있다)

4. **Cross GCC Command 창** :   
Cross compiler prefix : **arm-linux-gnueabihf- (주의하여 입력!!)**

Tool command path: **/home/계정/ti-processor-어쩌구/linux-devkit/sysroots/<Arago Linux>/usr/bin**

( Cross compiler profix 의 접두사는 **"-"** 로 끝난다. 이것은 Tool 명령 경로를 설정할 때 볼 수 있는 Cross-Compiler Tool 의 접두사이다) ,

(Tool command path 의 실제경로: **/home/yunk/sitara/ti-processor-어쩌구/linux-devkit/ sysroots/x86\_64-arago-linux/usr/bin**)

5. 소스코드를 작성할 **C파일을 생성**한다

마우스오른쪽클릭(팝업메뉴) > New > Source File >   
Source File : **helloworld.c** 입력   
> **Finish**

6. helloworld.c 에 다음 코드를 넣는다

|  |
| --- |
| #include <stdio.h>  int main(){  printf(“Hello world from TI! \n”);  return 0;  } |

7. 위의 코드를 **컴파일**한다

7.1 **망치**누르거나

7.2 **Project -> Build Project 선택**

7.3 결과로 나온 실행파일은 *Debug 디렉토리* 에서 찾을수 있다.

[10]

Configuring Remote System Explorer (RSE)

위에서 컴파일한 프로젝트를 , 원격 시스템 탐색기(RSE)를 사용하여 파일을 전송하기 위해, RSE 를 구성해본다

자세한 내용은 ㅡ <http://processors.wiki.ti.com/index.php/How_to_setup_Remote_System_Explorer_plug-in>

1. **RSE perspective** 를 연다

맨위의 메뉴줄의 Window > Perspective > Open Perspective > Other...

> **Remote System Explore**r 선택 > **OK**

2. 그러면, **RSE View** 가 열렸을 것이다. 이 상태에서,

맨위의 메뉴줄의 File > New >Other…

**Select a wizard** 창 : **Remote System Explorer** > **Connection** 선택 > **Next**

> **Linux** > **Next**

> Host name : **192.168.0.14 (DHCP이므로, IP ADDR 확인 후 그 주소 넣기)**

Connection name : MyTarget EVM

**Verify host name 체크해제**

> **Next**

> **ssh.files 만 체크**하자 > **Next**

> **Processes.shell.linux 만 체크**하자 > **Next**

> **ssh.shells 만 체크**하자 > **Finish**

3. 이제 RSE view 에서 **EVM 구성**을 볼 수 있다.

그런데 RSE view 가 활성화되면 C/C++ view 가 닫힐 가능성이 높다. 따라서, **C/C++ view** **띄우자.**

맨위의 메뉴줄의 Window -> Show View -> **Other…**> C/C++ -> **C/C++ Projects** > **OK**

(C/C++ Projects view 의 위치가 마음에 들지 않으면 CCS 의 다른 위치로 드래그 앤 드롭하여 Tab 을 끌어다 놓을 수 있다)

4. New Connection Wizard 가 완료된 이후 Remote System Explorer view 가 열리면,   
**target EVM 과 통신하도록 새 연결을 구성**해야한다. 따라서,

MyTarget EVM 에서 마우스오른쪽클릭(팝업메뉴) > Properties   
> **Host** > Default User ID: **root 로 변경** > **OK**

5. 이제 Remote System Explorer 를 사용할 준비가 되었다.   
target EVM file system 이 처음으로 개인 키로 부팅되고 공개 키가 target file system 에 만들어진다.   
target EVM 에 처음으로 연결하기 전에 공개 키를 target EVM 에서 Linux host system 으로 export(내보내기) 해야 한다. 키를 구성하기 위해 다음을 설정하자.

MyTarget EVM 에서 마우스오른쪽클릭(팝업메뉴) > **Connect**   
> User ID : **root** , **Password(optional) : 비워둔다** > **OK**

> Authontication 관련된 창이 뜨면, **OK(승인)** 눌러준다.

(만약, /home/sitara/.ssh/known\_hosts 가 존재하지 않는다고 메세지 뜨면, 대화상자에서 Yes 를 선택한다)

[11]

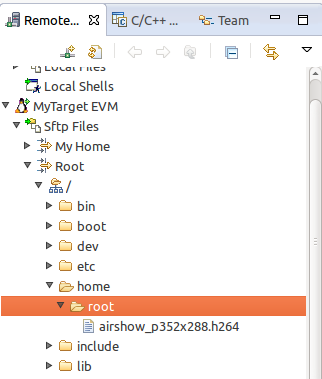
RSE(Remote System Explorer) 구성 했으니, Target System 에 파일을 전송해 보자.

[키포인트]

\* RSE 를 통하여, Host 와 Target Device 간에 File 을 전송하는데 사용할 수 있다.

\* RSE 를 통하여, Target Device 에 Terminal 을 열어 실행 파일을 실행할 수 있다.

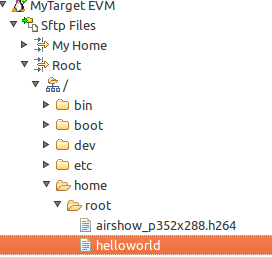
11.1. **Sftp Files -> Root -> /home -> root 를 연다**



11.2. C/C++Projects 창보기로 전환한 뒤, 아래의 **helloworld 실행파일을 Copy** 한다



11.3. RSE 창보기로 전환한 뒤, 위에서 열어둔 **root 폴더에 Paste** 한다



11.4. 이제 helloworld 실행 파일을 Target Device 로 전송해보자.

11.4.1 **Open Terminal 아이콘을 클릭**한다 (모니터 모양 아이콘)



11.4.2 **Launch Terminal 창** :   
 > Choose terminal : **SSH Terminal 선택**

> Host : **192.168.0.14** **(Target 의 IP Address 입력 - DHCP 이므로, 매번 IP ADDR 확인하여 설정해줘야 한다. 중요!!)**

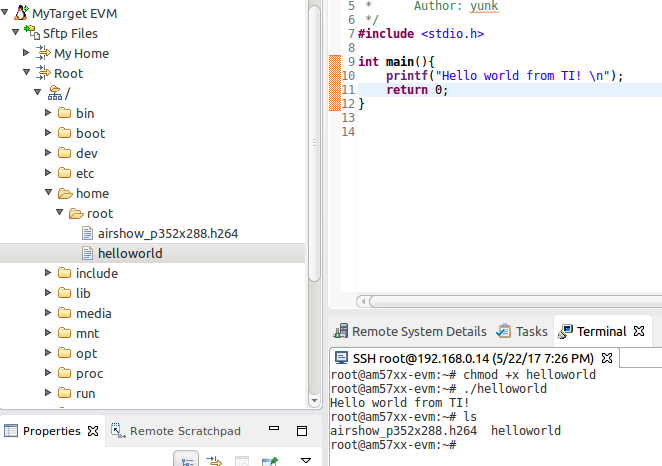
> User : **root**

11.4.3 그러면 **터미널 창**이 열린다. 이곳에 **다음 명령을 입력**한다

**> chmod +x helloworld**  (권한설정)

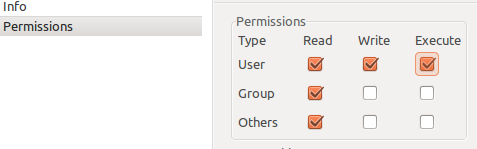
**> ./helloworld** (실행)

11.4.4 그러면 (하단에) 화면에 뿌려지는 것을 확인할 수 있다.



11.4.5 추가적으로, RSE 보기에서 파일을 마우스오른쪽클릭 > Properties > Permissions >

실행 권한을 활성화 하면,   
 RSE 를 사용하여 파일을 실행 가능하게 만들 수 있다.



[12]

helloworld 프로그램을 디버깅하도록 CCS 를 구성하는 방법 연습

[키포인트]

\* CCS 는 Project 파일을 자동으로 Build 하도록 구성할 수 있다.

\* RSE 는 Debug Host 와 Target 간에 파일을 전송할 때 유용하다.

\* CCS 에는 gdbserver 시작을 포함하여 Target 에서 명령을 실행하는데 사용할 수 있는 Terminal 이 있다.

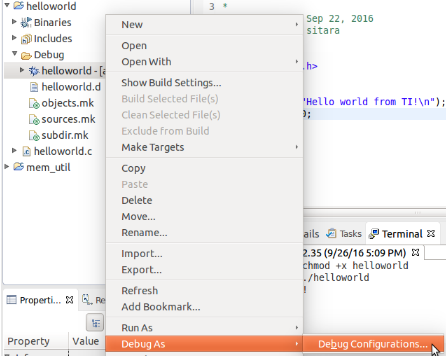
\* break points 를 사용하여 특정 지점에서 실행을 중지할 수 있다.

12.1. 먼저 **켜둔 CCS 의 터미널에 입력** 하여, Target 에서 gdbserver 를 시작해보자

**> gdbserver :10000 helloworld**

|  |
| --- |
|  |

12.2. **C/C++ Project 창모드로 전환**하고,   
 **(창이 없을 경우 맨윗줄 메뉴>view>Project Explorer 클릭)**  
Debug 폴더내 helloworld 실행파일 우클릭 > Debug As > **Debug Configurations...** 선택



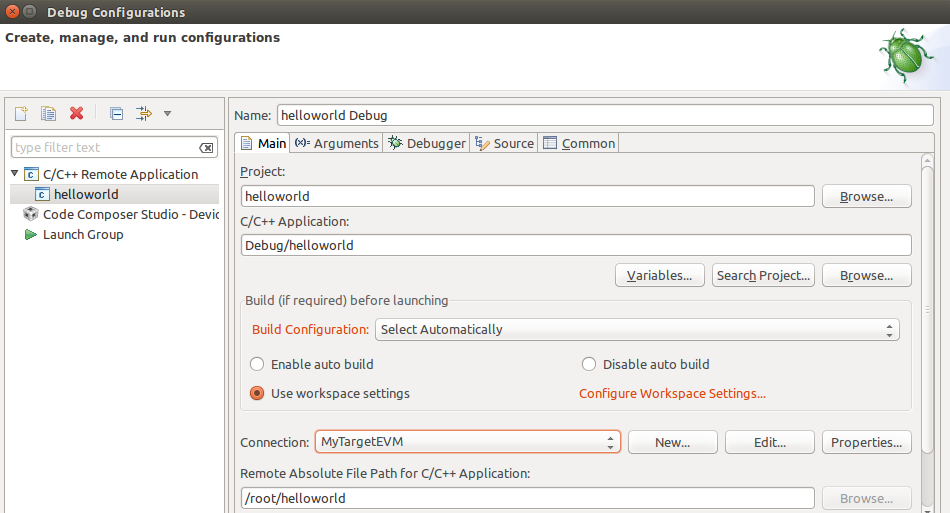
12.3. **C/C++ Remote Application** 을 우클릭하여 **New** 를 선택



12.4. **Debug Configuretions 창의 Main 탭**에서

Name 을 **helloworld Debug** 로 변경  
Remote Absolute File Path for C/C++ Application 을 **/home/root/helloworld** 로 변경

Connection 을 New 눌러서 > **MyTargetEVM** 으로 만들어서 변경



12.5. **Debug Configuretions 창의 Debugger 탭**에서

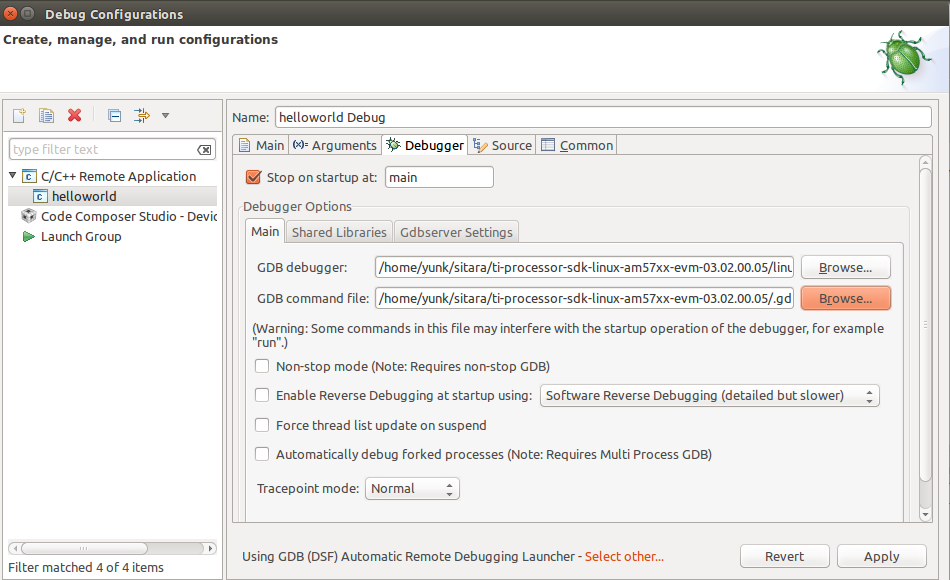
5.1 GDB debugger : **/home/sitara/ti-processor-sdk-linux-어쩌구/linux-devkit/sysroots**

**/<Arago Linux>/usr/bin/arm-linux-gnueabihf-gdb** 로 경로찾아 넣기

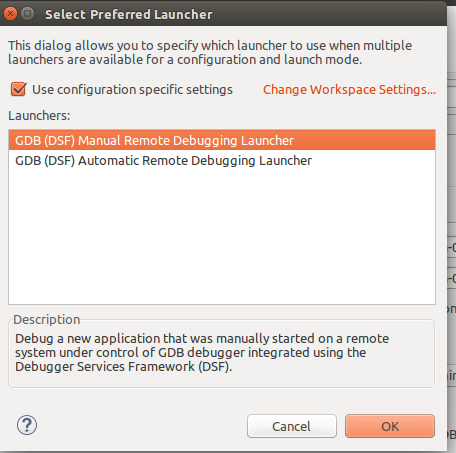
5.2 GDB command file : **ti-processor-sdk-linux-<machine>-<version>/.gdbinit** 로

경로찾아 넣기 **(*우클릭*하여 *Show hidden files 선택*하여 *.gdbinit 파일을 찾아서 클릭*)**

5.3 맨 밑 **Using GDB(DSF) 옆**에, **Select other..** 클릭



5.4 **GDB(DSF) Manual Remote Debugging Launcher** 를 선택하고, **OK** 를 클릭한다

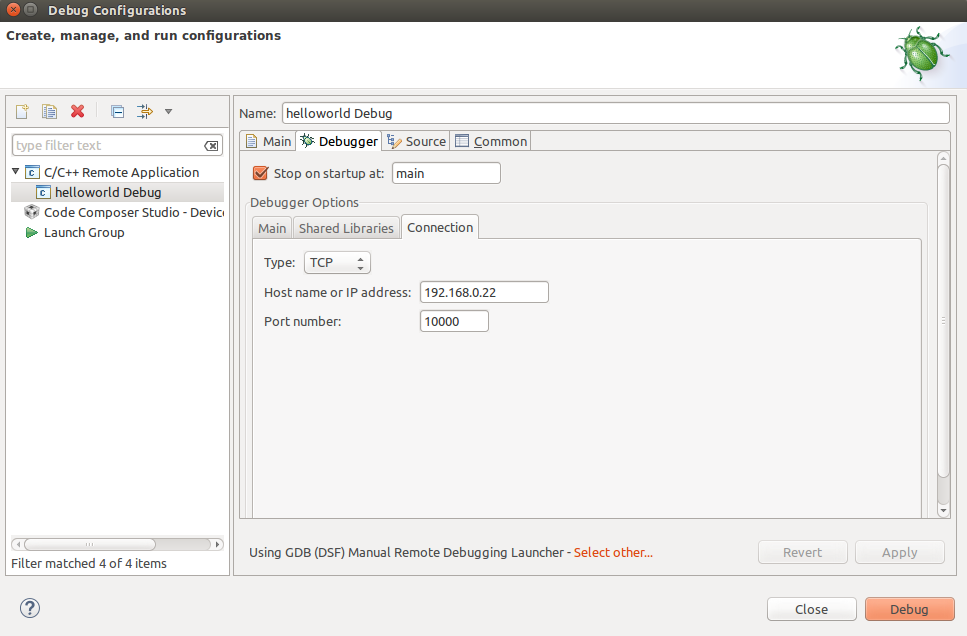


5.5 **Debug Configuretions 창**의 **Debugger** 탭에서 > **Connection** 에서

> Host name or IP address : **192.168.0.14**

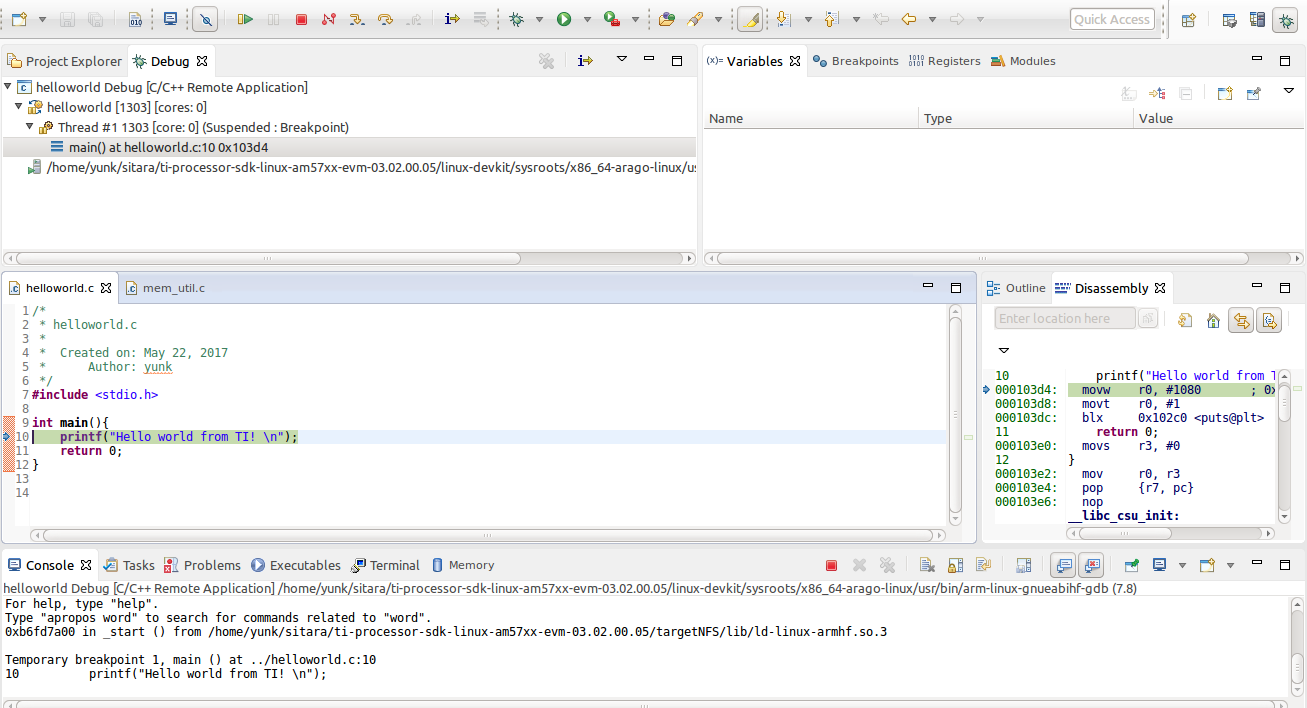
**(위에서부터 계속 설정하고 있는 그 IP ADDR !)**

> **Apply** 클릭



|  |
| --- |
| **디버깅이 안될 경우**  **1. DHCP의 경우 IP ADDR 가 바뀌기 때문에, Launch Terminal 재설정이 필요**하다  **11.4 이후부터~ 다시 시도**해본다  **(이 때, 현재 IP로 재설정이 필요)**  **2. C++과 ARM 의 크로스컴파일 관련 다운로드가 필요한 경우**(특히 NewPC)  2.1 다음 명령을 터미널에 입력하자  > sudo apt-get install -y gcc-arm-linux-gnueabihf  > sudo apt-get install -y libncurses-dev  > sudo apt-get install -y libqt4-dev pkg-config  > sudo apt-get install -y u-boot-tools  > sudo apt-get install -y device-tree-compiler  2.2 C ++의 크로스 컴파일을 위해, g ++을 설치하기  > sudo apt-get install g++-arm-linux-gnueabihf  **3.** **CCS 내의 터미널을 켜 둔 상태**에서 **디버깅을 실행**해야 한다 **(완전중요!)**  3.1 **[12.1]**을 참고하여, CCS의 터미널에 **>gdbserver :10000 helloworld 입력**한다  3.2 **[12.5]**를 참고하여, **디버깅**을 실행한다 |

**12.6 디버깅이 올바르게 실행된 상태**



12.6.1 현재 주기능의 첫번째 줄에서 System 이 정지해 있다. 확인하자

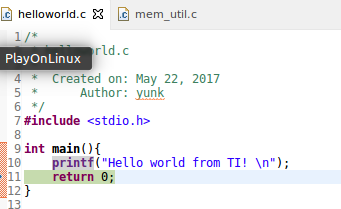
그리고, 이제 창의 맨 위에 있는 Control을 통하여 Debug 세션을 제어할 수 있게 되었다

12.6.2 **Step Over 버튼을 클릭**해보자.



그러면, 활성화된듯 색깔줄이(Active line) **return 0**로 이동하고,

Hello World 메시지가 콘솔에 출력된다

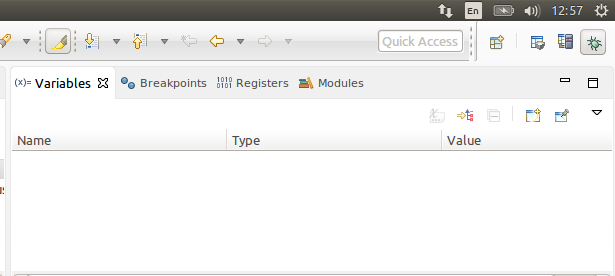


12.6.3 **Resume 버튼**을 클릭하여, 실행을 마치고, 디버깅상태를 종료한다

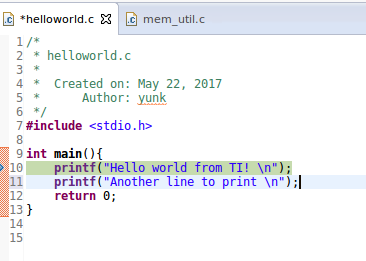


12.7 **[12.1]**을 참고하여, CCS의 터미널에 **>gdbserver :10000 helloworld 입력**한다

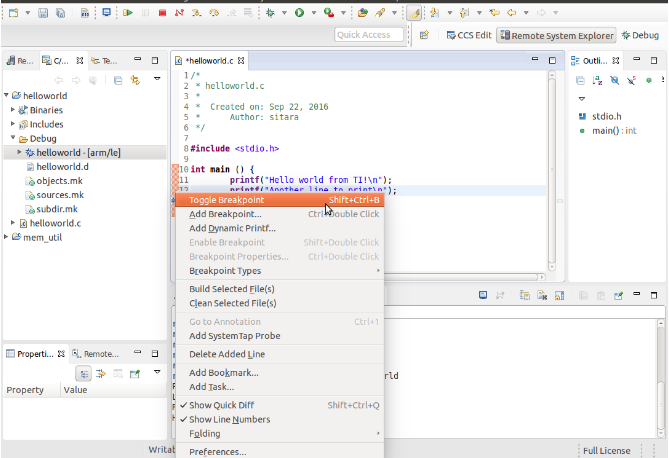
12.7.1 이제 창의 오른쪽 위 모서리에 있는 Remote System Explorer 를 선택하여, Debugger View를 종료한다



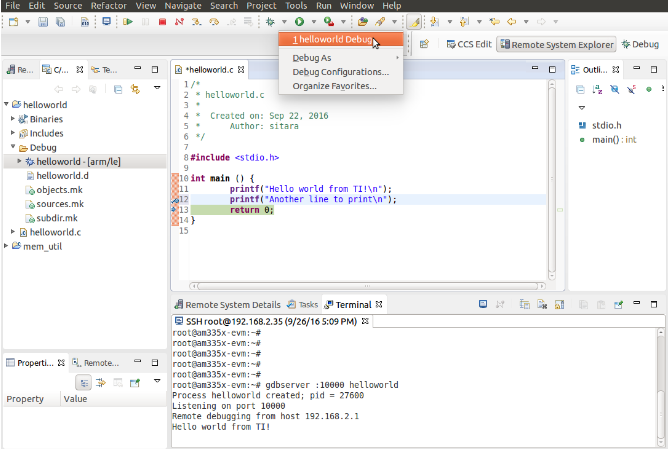
12.7.2 다음과 같이 C코드 **printf(“....”);** 한줄을 추가한다



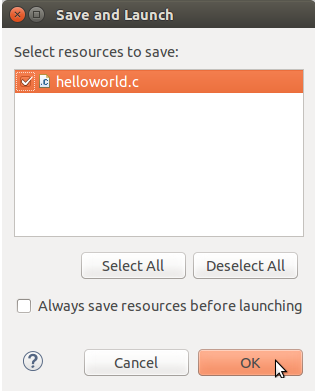
12.7.3 새로운 행 왼쪽에 있는 회색 열을 우클릭하여, **Breakpoint Toggle**(껏다켜기) 클릭



12.7.4 상단 Toolbar 에서 Bug 아이콘 옆에 있는 화살표를 클릭하고 **1 helloworld Debug** 구성을 선택한다.

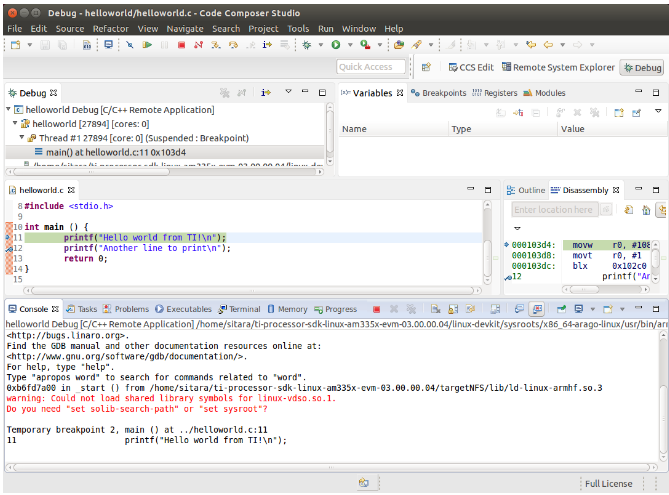


12.7.5 변경된 내용을 helloworld.c 에 저장하라는 메시지가 나타나면 **OK** 를 선택한다.

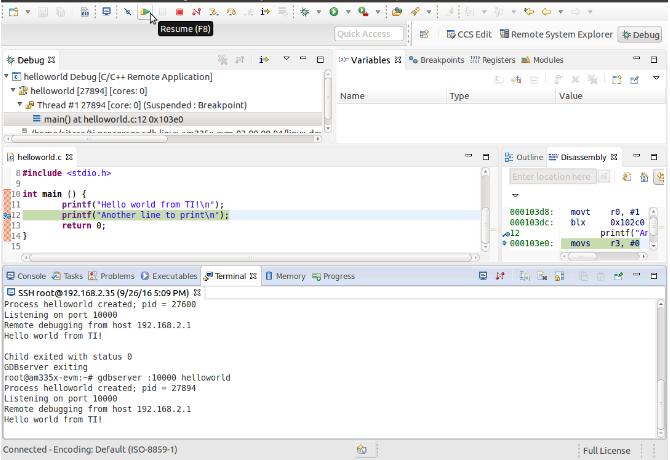


12.7.6 소스가 컴파일되고 Debug View 가 자동으로 열리는 것을 볼 수 있다.

다시 한 번 현재 위치가 함수의 첫 번째 줄이 될 것이지만 이번에는 소스에 표시된 breakpoint 가 있다. RSE 를 사용하여 실행 파일을 Target 으로 복사하는 것을 잊지말자.



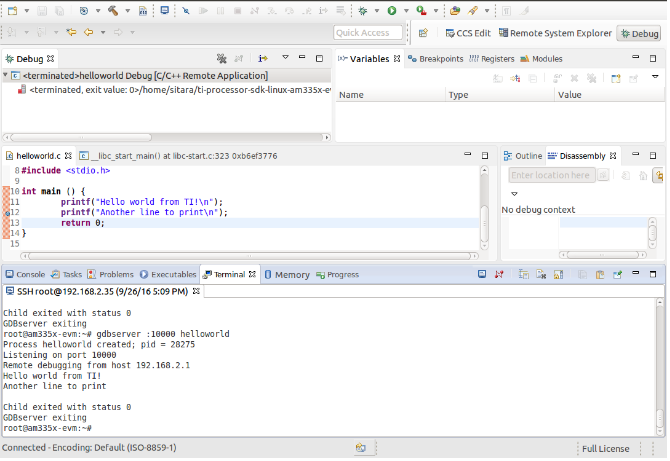
12.7.7 실행 아이콘을 클릭하고 프로그램이 breakpoint 에서 자동으로 중지되는지 확인한다. 콘솔 출력에 Hello World 가 표시된다



12.7.8 Step Over 아이콘을 클릭하면 방금 추가된 2 번째 printf 행이 실행되므로 Target File System 에 변경 사항이 Update 된다.



12.7.8 이제 Stop 버튼을 눌러 Debugger 를 중지할 수 있다.



12.7.9 Debugger 를 다시 시작하려면 상단 Toolbar 에서 Bug 아이콘을 클릭하여 프로그램을 다시 시작하여 Debug 한다.