

승인 된 버전. 승인 된 날짜 : 2020 년 1 월 24 일 10:12

# 배포 패키지를 사용하여 크로스 컴파일하는 방법

## 내용


- 1 기사 목적
- 2 전제 조건
- 3 커널 수정
  - 3.1 서문
  - 3.2 커널 설정 수정
  - 3.3 리눅스 커널 장치 트리 수정
  - 3.4 내장 Linux 커널 장치 드라이버 수정
  - 3.5 외부 리눅스 커널 모듈 수정 / 추가
- 4 외부 트리 외부 Linux 커널 모듈 추가
- 5 U-부팅 수정
- 6 TF-A 수정
- 7 "hello world"사용자 공간 예제 추가
- 8 가지 팁
  - 8.1 마운팅 포인트 생성


## 1 기사 목적

이 기사는 [devtool](#) 및 [BitBake](#) 도구를 사용한 크로스 컴파일을 설명하는 OpenSTLinux 배포판의 배포 패키지에 대한 간단한 예를 제공 합니다.

- Linux ® 커널로 수정 (구성, 장치 트리, 드라이버 등)
- 외부 인트 리 리눅스 커널 모듈의 수정
- U-Boot 수정
- TF-A의 수정
- 소프트웨어 추가

이 예제는 또한 호스트 컴퓨터에 대한 네트워크 연결을 통해 대상에 크로스 컴파일 결과를 배포하는 방법을 보여줍니다.

 이 페이지에 설명 된 모든 예제는 OpenEmbedded의 devtool 및 / 또는 비트 베이크를 사용합니다. 자세한 내용은 [OpenEmbedded-devtool](#) 을 참조하십시오.

 동일한 결과를 얻는 방법에는 여러 가지가 있습니다. 이 기사는 예제 당 하나 이상의 솔루션을 제공하는 것을 목표로합니다. 개발 제약 조건에 더 적합한 다른 방법을 탐색 할 자유가 있습니다.

## 전제 조건 2 개

[OpenSTLinux 배포 설치](#) 의 전제 조건을 실행해야 합니다.

보드와 호스트 시스템은 이더넷 링크를 통해 연결되며 호스트 시스템에서 원격 터미널 프로그램이 시작됩니다. [터미널을 얻는 방법](#) 을 참조하십시오 .

대상이 시작되고 해당 IP 주소 (<board ip address>)가 알려져 있습니다.

## 3 커널로 수정

### 3.1 서문

모듈로 수정을 시작하려면 배포 패키지 환경을 초기화해야 합니다.

```
PC $> cd <작업 분배 경로>
PC $> DISTRO = openstlinux-weston MACHINE = stm32mp1 소스 레이어 /meta-st/scripts/envsetup.sh
```

이제 다음 단락에서 <build dir>로 식별된 빌드 디렉토리에 있습니다.

커널 구성 요소에 대한 devtool을 초기화하십시오.

```
PC $> devtool 수정 가상 / 커널
참고 : 비트 베이크 서버 시작 중 ...
참고 : / mnt / internal_storage / oetest / oe_openstlinux_rocko / build-openstlinuxweston-stm32mp1 / workspace에서 작업 영역 레
참고 : bblayers.conf에서 작업 영역 레이어 활성화
구분 분석 레시피 : 100 % | ##### || 시간 : 0:00:54
2401 .bb 파일 구분 분석이 완료되었습니다 (0 캐시, 2401 구분 분석). 타겟 3282 개, 건너 뛰기 88 개, 마스크 0 개, 오류 0 개
참고 : 가상 / 커널을 linux-stm32mp에 매핑
참고 : 누락된 작업 대기열 종속성 해결
...
```

virtual / <something> 컴포넌트의 경우 가상 컴포넌트와 연관된 레시피 간의 매핑 이름을 가져와야 합니다. 이 예에서 커널 레시피의 이름은 추적에 표시되지  
만 devtool status를 호출하여 얻을 수도 있습니다.

소수의 devtool 명령은 devtool modify와 같은 virtual / <something> 컴포넌트를 지원하므로 virtual / component와 연관된 레시피 이름을 가져와야 합니다.  
이 예에서 커널 레시피 이름은 추적 (linux-stm32mp)에 표시됩니다.

커널의 소스 코드는 <build dir> / workspace / sources에 있습니다. devtool modify 명령을 사용자 정의하여 소스 코드가 추출되는 경로를 변경할 수 있습니다

커널과 관련된 모든 작업을 위해 바이너리를 보드에 배포하는 데 약간의 사용을 권장합니다.

- 커널 이미지, 장치 트리 : bitbake deploy 명령 및 scp 사용 ([#kernel 구성 수정 참조](#))
- 커널 모듈 : 임시 디렉토리를 전달하여 devtool deploy-target을 사용하십시오 ([# 외부 Linux 커널 모듈 수정 / 추가 참조](#))

사용법의 차이는 보드에 배포 할 파일 수에서 비롯됩니다. 하나 또는 두 개의 파일 만 보드에 넣을 경우 가장 쉬운 방법은 원하는 파일 만 배포하는 것입니다.

### 3.2 커널 설정 수정

이 간단한 예는 CMA 크기에 대해 menuconfig를 통해 커널 구성을 수정합니다.

- 대상 부팅 로그의 분석을 통해 CMA 크기 (여기서는 128MB)의 현재 값을 가져옵니다.

```
보드 $> dmesg | grep -i cma
[0.000000] cma : 0xf0000000에서 128MiB 예약
```

- Linux 커널 구성 메뉴를 시작하십시오. [Menuconfig](#) 또는 커널 구성 [방법](#)을 참조하십시오.
- "장치 드라이버-일반 드라이버 옵션"으로 이동하십시오.
  - "메가 바이트 크기"를 선택하십시오
  - 값을 256으로 수정
  - 새 구성을 종료하고 저장하십시오.
- 구성 파일 (.config)이 수정 되었는지 확인

```
PC $> grep -i CONFIG_CMA_SIZE_MBYTES <빌드 디렉토리> / workspace / sources / <커널 레시피 이름> /. config.new
CONFIG_CMA_SIZE_MBYTES = 256
```

- Linux 커널 크로스 컴파일 : [Menuconfig](#) 또는 커널 구성 [방법](#)을 참조하십시오.
- 보드의 Linux 커널 이미지 업데이트 : [Menuconfig](#) 또는 커널 구성 [방법](#)을 참조하십시오.
- 보드를 재부팅합니다 : [Menuconfig](#) 또는 커널 구성 [방법](#)을 참조하십시오.
- 대상 부트 로그 분석을 통해 CMA 크기 (256MB)의 새로운 값을 얻습니다.

```
보드 $> dmesg | grep -i cma
[0.000000] CMA : 소유 256 0xe0000000에서 MiB 크기
```

### 3.3 리눅스 커널 장치 트리 수정

이 간단한 예는 사용자 LED의 기본 상태를 수정합니다.

- 보드가 시작된 상태에서; 사용자 LED (LD3)가 비활성화되어 있는지 확인

- <빌드 디렉토리> / workspace / sources / <커널 레시피 이름> / 디렉토리로 이동하십시오.

```
PC $> cd <빌드 디렉토리> / workspace / sources / <커널 레시피 이름> /
```

- 평가 보드의 arch / arm / boot / dts / stm32mp157c-ed1.dts 장치 트리 소스 파일을 편집하거나

감지 보드의 arch / arm / boot / dts / stm32mp157c-dk2.dts 장치 트리 소스 파일을 편집하거나

- "stm32mp : green : user"의 상태를 "okay"로 변경하고 기본 상태를 "on"으로 설정하십시오.

```
주도 {
    호환 가능 = "gpio-leds";

    상태 = "괜찮아 ";
    빨간색 {
        레이블 = "stm32mp : red : status";
        gpios = <& gpioa 13 GPIO_ACTIVE_LOW>;

        상태 = "비활성화 됨";
    };
    초록 {
        레이블 = "stm32mp : green : user";
        gpios = <& gpioa 14 GPIO_ACTIVE_LOW>;
        default-state = "on";

        상태 = "괜찮아 ";
    };
};
```

- 빌드 디렉토리로 이동

```
PC $> cd <빌드 디렉토리>
```

이 업데이트가 승인되면이 페이지의 다른 단락에보고하십시오}}

- 장치 트리 Blob 생성 ( \*.dtb )

```
PC $> 비트 베이크 가상 / 커널 -C 컴파일
```

- 보드의 장치 트리 모양 업데이트

```
PC $> scp <빌드 디렉토리> / tmp-glibc / deploy / images / <컴퓨터 이름> / *. dtb root @ <보드 IP 주소> : / boot
```



는 IF / 부팅 설치 지점이 아직 존재하지 않는, 참조하시기 바랍니다 장치 지점을 만드는 방법

- 보드를 재부팅

```
보드 $> cd / boot; 동조; systemctl 재부팅
```

- 사용자 LED (LD3)가 활성화되어 있는지 확인하십시오 (녹색)

### 3.4 내장 Linux 커널 장치 드라이버 수정

이 간단한 예제는 디스플레이 드라이버가 프로브 될 때 무조건 로그 정보를 추가합니다.

- 디스플레이 드라이버가 프로브 될 때 로그 정보가 없는지 확인하십시오

```
보드 $> dmesg | grep -i stm_drm_platform_probe
보드 $>
```

- <빌드 디렉토리> / workspace / sources / <커널 레시피 이름> /으로 이동하십시오.

```
PC $> cd <빌드 디렉토리> / workspace / sources / <커널 레시피 이름> /
```

- ./drivers/gpu/drm/stm/drv.c 소스 파일을 편집 하십시오.
- stm\_drm\_platform\_probe 함수 에 로그 정보 추가

```
정적 int stm_drm_platform_probe (struct platform_device * pdev)
{
    struct device * dev = &pdev->dev;
    struct drm_device * ddev;
    ret;
    [...]

    DRM_INFO ( "간단한 예-%s \n", __func__ );

    리턴 0;
    [...]
}
```

- 빌드 디렉토리로 이동

```
PC $> cd <빌드 디렉토리>
```

- 리눅스 커널 크로스 컴파일

```
PC $> 비트 베이크 가상 / 커널 -c 컴파일
```

- 온보드 Linux 커널 이미지 업데이트

```
PC $> scp <빌드 디렉토리> / tmp-glibc / deploy / images / <컴퓨터 이름> uImage root @ <보드 IP 주소> : / boot
```



는 IF / 부팅 설치 지점이 아직 존재하지 않는, 참조하시기 바랍니다 장착 지점을 만드는 방법

- 보드를 재부팅

```
보드 $> cd / boot; 동조; systemctl 재부팅
```

- 디스플레이 드라이버가 프로브 될 때 로그 정보가 있는지 확인하십시오.

```
보드 $> dmesg | grep -i stm_drm_platform_probe
[5.005833] [drm] 간단한 예 -stm_drm_platform_probe
```

### 3.5 외부 Linux 커널 모듈 수정 / 추가

Linux 커널의 대부분의 장치 드라이버 (모듈)는 커널 자체 (내장 / 내부 모듈) 또는 / 아래의 루트 파일 시스템에 배치해야하는로드 가능한 커널 모듈 (LKM / 외부 모듈)로 컴파일 할 수 있습니다. lib / modules 디렉토리 외부 모듈은 트리 내부 (커널 트리 구조) 또는 트리 외부 (커널 트리 구조 외부) 일 수 있습니다.

이 간단한 예는 가상 비디오 테스트 드라이버 (vivid) 커널 모듈이 프로브되거나 제거 될 때 무조건 로그 정보를 추가합니다.

- <빌드 디렉토리> / workspace / sources / <커널 레시피 이름> /으로 이동하십시오.

```
PC $> cd <빌드 디렉토리> / workspace / sources / <커널 레시피 이름> /
```

- ./drivers/media/platform/vivid/vivid-core.c 소스 파일을 편집 하십시오.
- vivid\_probe 및 vivid\_remove 함수 에 로그 정보 추가

```
static int vivid_probe (struct platform_device * pdev)
{
    const struct font_desc * font = find_font ( "VGA8x16");
    int ret = 0, i;
    [...]

    / * n_devs는 할당 된 실제 장치 수를 반영합니다 * /
    n_devs = i;

    pr_info ( "간단한 예-% s \ n", __func__ ) ;

    리트 윗을 반환;
}
```

```
static int vivid_remove (struct platform_device * pdev)
{
    struct vivid_dev * dev;
    부호없는 int i, j;
    [...]

    pr_info ( "간단한 예-% s \ n", __func__ ) ;

    리턴 0;
}
```

- 빌드 디렉토리로 이동

```
PC $> cd <빌드 디렉토리>
```

- 리눅스 커널 모듈 크로스 컴파일

```
PC $> 비트 베이킹 가상 / 커널 -c 컴파일
```

- 보드에서 생성한 커널 모듈 업데이트

```
PC $> devtool deploy-target -Ss <커널 레시피 이름> root @ <board ip address> : /
```

- 로드 가능한 커널 모듈에 대한 종속성 설명을 업데이트하고 디스크의 데이터를 메모리와 동기화

```
보드 $> / sbin / depmod -a
보드 $> 동기화
```

- 생성한 커널 모듈을 Linux 커널에 삽입

```
보드 $> 모드 프로브 생성한
[...]
[3412.784638] 간단한 예 -vivid_probe
```

- Linux 커널에서 생성한 커널 모듈 제거

```
보드 $> rmmod 생성한
[...]
[3423.708517] 간단한 예 -vivid_remove
```

## 4 외부 트리 외부 Linux 커널 모듈 추가

이 간단한 예제는 "Hello World"외부 트리 외부 Linux 커널 모듈을 Linux 커널에 추가합니다.

- 이 커널 모듈 예제의 디렉토리를 만듭니다

```
PC $> mkdir kernel_module_example
PC $> cd 커널 _module_example
```

- 이 커널 모듈 예제의 소스 코드 파일을 작성하십시오. *kernel\_module\_example.c*

```
// SPDX 식별자 : GPL-2.0
/ *
 * Copyright (C) STMicroelectronics SA 2018
 *
 * 저자 : Jean-Christophe Troatin <jean-christophe.troatin@st.com>
 *
 * /

모든 커널 모듈에 대해 #include <linux / module.h> / * * /
KERN_INFO의 경우 #include <linux / kernel.h> / * * /
__init 및 __exit 매크로의 경우 #include <linux / init.h> / * * /

static int __init kernel_module_example_init ( void )
{
    printk ( KERN_INFO "커널 모듈 예 : STMicroelectronics의 hello world \ n " );
    리턴 0 ;
}

static void __exit kernel_module_example_exit ( void )
```

- 이 커널 모듈 예제에 대한 makefile을 작성하십시오. *Makefile*



makefile의 모든 들어 쓰기는 표입니다

```
# 간단한 외부 트리 외부 리눅스 커널 모듈 예제를위한 Makefile

# 빌드 할 오브젝트 파일
obj-m := kernel_module_example.o

# 리눅스 커널 소스 코드를 포함하는 디렉토리 경로
# 및 구성 파일 (.config)
KERNEL_DIR? = <리눅스 커널 경로>

# 컴파일 할 소스 파일이 들어있는 디렉토리 경로
PWD := $ (셀 암호)

기본:
$ (MAKE) -C $ (KERNEL_DIR) M = $ (PWD) 모듈

깨끗한:
$ (MAKE) -C $ (KERNEL_DIR) M = $ (PWD) 청소
```

- 작업 공간에 새로운 레시피 추가

```
PC $> cd <빌드 디렉토리>
PC $> devtool add mymodule kernel_module_example /
```

- 커널 모듈 빌드에 레시피를 적용

```
PC $> devtool 편집 레시피 mymodule
```

다음 변경 사항에 따라 레시피를 수정하십시오 (강조 표시된 라인 참조).

```
1 # recipetool로 만든 레시피
2 # 이것은 레시피의 기초이며 완전히 기능하기 위해서는 추가 편집이 필요할 수 있습니다.
3 # (편집 할 때 설명을 자유롭게 제거하십시오.)
4
5 # 라이선스 명세서처럼 보이는 파일을 찾을 수 없습니다. 첨부 된
6 개의 설명서 및 소스 헤더를
7 확인 하고 이에 따라 LICENSE 및 LIC_FILES_CHKSUM을 설정하십시오. #
8 # 참고 : 라이선스는 적어도 시작 건물을 할 수 있도록 "CLOSED"로 설정을되고있는 - 경우
9 이 내장되는 소프트웨어의 라이선스에 대한 정확하지 않은 # (그
10이 사용하기 전에 올바른 값을 지정해야 합니다) 대부분의 경우되지 않습니다
11 초기 테스트 / 개발 이외의 아무것도 # 레시피!
12 LICENSE = "CLOSED"
13 LIC_FILES_CHKSUM = ""
14
15 # SRC_URI에 대한 정보가 아직 없습니다 (외부 소스 트리 만 지정됨)
16 SRC_URI = ""
17
18 # 참고 : 이것은 Makefile 전용 소프트웨어입니다.
19 # 레시피의
```

- 빌드 디렉토리로 이동

```
PC $> cd <빌드 디렉토리>
```

- 생성 된 커널 모듈 예제

```
PC $> devtool 빌드 mymodule
```

- 이 커널 모듈 예제를 보드에 푸시

```
PC $> devtool deploy-target -Ss mymodule root @ <보드 IP 주소>
```

- 로드 가능한 커널 모듈에 대한 종속성 설명을 업데이트하고 디스크의 데이터를 메모리와 동기화

```
보드 $> /sbin/depmod -a
보드 $> 동기화
```

- 커널 모듈 예제를 Linux 커널에 삽입

```
보드 $> modprobe kernel_module_example
[18167.821725] 커널 모듈 예 : STMicroelectronics의 hello world
```

- Linux 커널에서 커널 모듈 예제를 제거하십시오.

```
보드 $> rmmod kernel_module_example
[18180.086722] 커널 모듈 예 : STMicroelectronics의 작별
```

## 5 U-Boot 수정

이 간단한 예는 U-Boot가 시작될 때 무조건 로그 정보를 추가합니다. 신뢰할 수 있는 부팅 체인 범위 내에서 U-Boot는 2 단계 부팅 로더 (SSBL)로 사용됩니다.

- 보드가 재부팅 될 때 U-Boot 로그 정보를 확인하십시오

```

보드 $> 재부팅
[...]
U- 부트 <U- 부트 버전>

CPU : STM32MP1 rev1.0
모델 : STMicroelectronics STM32MP157C [...]
보드 : 트러스트 모드의 stm32mp1
[...]

```

- 빌드 디렉토리로 이동

```
PC $> cd <빌드 디렉토리>
```

- U-boot 레시피 검색

```

PC $> devtool 검색 u-boot *
u-boot-stm32mp-extlinux U-boot 용 'extlinux.conf'파일 생성
u-boot-stm32mp stm32mp 용 내장 장치 용 범용 부트 로더

```

이 예에서 레시피 이름은 **u-boot-stm32mp**입니다.

- u-boot로 작업 시작

```
PC $> devtool 수정 u-boot-stm32mp
```

```

예 :
PC $> cd <build dir> / workspace / sources / u-boot-stm32mp

```

- `./board/st/stm32mp1/stm32mp1.c` 소스 파일을 편집 하십시오.
- *바독판* 기능 예 로그 정보 추가

```

int checkboard (void)
{
    문자 * 모드;

    [...]

    printf ( "보드 : % s 모드에서 stm32mp1 \ n", 모드);
    printf ( "U-Boot 간단한 예 \ n");

    리턴 0;
}

```

- U-Boot 크로스 컴파일 : 신뢰할 수 있는 부팅

```

PC $> devtool build u-boot-stm32mp
PC $> bitbake u-boot-stm32mp -c 배포

```

- 컴파일 결과가 저장된 디렉토리로 이동

```
PC $> cd <빌드 디렉토리> / tmp-glibc / deploy / images / <컴퓨터 이름> /
```

- 보드를 재부팅하고 U-boot 셸에서 멈추려면 아무 키나 누르십시오.



```

보드 $> 재부팅
[...]
자동 부팅을 중지하려면 아무 키나 누르십시오 : 0
STM32MP>

```

- USB OTG 포트를 통해 호스트 시스템과 보드 사이에 USB 케이블을 연결하십시오
- U-Boot 셸에서 USB 대용량 저장 기능을 호출하십시오.

```
STM32MP> 음 0 mmc 0
```



U-Boot UMS 기능 사용에 대한 자세한 내용은 U-Boot [에서 USB 대용량 저장소를 사용하는 방법을](#) 참조하십시오.

- 호스트 시스템에서 2 차 스테이지 부트 로더 ( *ssbl* ) 와 연관된 파티션을 확인하십시오 . *sd3* here

```

PC $> ls -l / dev / disk / by-partlabel /
총 0
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 bootfs-> ../../sd4
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 fsbl1-> ../../sd1
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 fsbl2-> ../../sd2
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 rootfs-> ../../sd5
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 ssbl-> ../../sd3
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 userfs-> ../../sd6

```

- 바이너리 (u-boot.stm32)를 전용 파티션에 복사

```
PC $> dd if = u-boot- <보드 이름> -trusted.stm32 of = / dev / sdc3 bs = 1M conv = fdatasync
```

(여기서 u-boot-stm32mp157c-ev1-trusted.stm32 또는 u-boot-stm32mp157c-dk2-trusted.stm32)

- U-Boot 셸 재설정

```
STM32MP> 리셋
```

- 보드가 재부팅 될 때 새로운 U-Boot 로그 정보를 살펴보십시오

```

[...]
U- 부트 <U- 부트 버전>

CPU : STM32MP1 rev1.0
모델 : STMicroelectronics STM32MP157C [...]
보드 : 트러스트 모드의 stm32mp1
U-Boot 간단한 예
[...]

```

## 6 TF-A 수정

이 간단한 예는 TF-A가 시작될 때 무조건 로그 정보를 추가합니다. 신뢰할 수 있는 부팅 체인 의 범위 내에서 TF-A는 FSBL (First Stage Boot Loader)로 사용됩니다.

- 보드가 재부팅 될 때 TF-A 로그 정보를 살펴보십시오

```

보드 $> 재부팅
[...]
정보 : MPU (MPSYSRST)에 의해 생성 된 시스템 재설정
정보 : SDMMC 사용
[...]

```

- 빌드 디렉토리로 이동

```
PC $> cd <빌드 디렉토리>
```

- TF-A 레시피 검색

```
PC $> devtool 검색 tf-a *
babeltrace Babeltrace-추적 형식 바벨 타워
C 및 유니 코드 문자열을 조작하기위한 libunistring 라이브러리
lttng-tools Linux 추적 툴킷 제어
gettext 다국어 메시지 생성을위한 유틸리티 및 라이브러리
glibc GLIBC (GNU C 라이브러리)
STM32MP1 용 tf-a-stm32mp 신뢰할 수 있는 펌웨어 -A
gnutls GNU 전송 계층 보안 라이브러리
gststreamer1.0 GStreamer 1.0 멀티미디어 프레임 워크
harfbuzz 텍스트 셰이핑 라이브러리
glibc의 glibc-locale 로컬 데이터
kbd 키 테이블 파일 및 키보드 유틸리티
```

이 예에서 레시피 이름은 **tf-a-stm32mp**입니다.

- tf-a로 작업 시작

```
PC $> devtool 수정 tf-a-stm32mp
```

- <build dir> / workspace / sources / tf-a-stm32mp로 이동하십시오

```
PC $> cd <빌드 디렉토리> / workspace / sources / tf-a-stm32mp
```

- ./plat/stm32mp1/bl2\_io\_storage.c 소스 파일을 편집 하십시오.
- stm32mp1\_io\_setup 함수 에 로그 정보 추가

```
무효 stm32mp1_io_setup (void)
{
    int io_result;
    [...]

    / * 재설정 이유에 대한 추적 추가 * /
    print_reset_reason ();

    INFO ( "TF-A 간단한 예");

    [...]
}
```

- TF-A 크로스 컴파일

```
PC $> devtool 빌드 tf-a-stm32mp
PC $> bitbake tf-a-stm32mp -c 배포
```

- 컴파일 결과가 저장된 디렉토리로 이동

```
PC $> cd <빌드 디렉토리> / tmp-glibc / deploy / images / <컴퓨터 이름> /
```

- 보드를 재부팅하고 U-boot 셸에서 멈추려면 아무 키나 누르십시오.

```
보드 $> 재부팅
[...]
자동 부팅을 중지하려면 아무 키나 누르십시오 : 0
STM32MP>
```

- USB OTG 포트를 통해 호스트 시스템과 보드 사이에 USB 케이블을 연결하십시오
- U-Boot 셸에서 USB 대용량 저장 기능을 호출하십시오.

```
STM32MP> 음 0 mmc 0
```



U-boot um 기능 사용에 대한 자세한 내용은 [U-Boot에서 USB 대용량 저장소를 사용하는 방법을 참조하십시오](#).

- 호스트 시스템에서 첫 번째 단계 부트 로더와 연관된 파티션 (백업으로 *fsbl1* 및 *fsbl2*) 과 관련된 파티션을 확인하십시오 . *sdcl* 및 *sdcl2* (백업으로) 여기

```
PC $> ls -l / dev / disk / by-partlabel /
총 0
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 bootfs-> ../../sdcl4
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 sfsbl1-> ../../sdcl
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 sfsbl2-> ../../sdcl2
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 rootfs-> ../../sdcl5
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 ssbl-> ../../sdcl3
lrwxrwxrwx 1 루트 루트 10 1 월 17 일 18:05 userfs-> ../../sdcl6
```

- 바이너리 (tf-a-stm32mp157c-ev1-trusted.stm32)를 전용 파티션에 복사하십시오. 새 TF-A 바이너리를 테스트하려면 백업 FSBL (*fsbl2*)에 이전 TF-A 바이너리를 유지하는 것이 유용 할 수 있습니다

```
PC $> dd if = tf-a- <보드 이름> -trusted.stm32 of = / dev / sdcl bs = 1M conv = fdatasync
```

(여기서 tf-a-stm32mp157c-ev1-trusted.stm32 또는 tf-a-stm32mp157c-dk2-trusted.stm32)

- U-Boot 셸 재설정

```
STM32MP> 리셋
```

- 보드가 재부팅 될 때 새로운 TF-A 로그 정보를 살펴보십시오

```
[...]
정보 : MPU (MPSYSRST)에 의해 생성 된 시스템 재설정
정보 :      TF-A 간단한 예
정보 : SDMMC 사용
[...]
```

## 7 "hello world"사용자 공간 예제 추가

이 장에서는 간단한 "hello world"예제를 컴파일하고 실행하는 방법을 보여줍니다.

- 이 사용자 공간 예제에 대한 디렉토리를 작성하십시오.

```
PC $> mkdir hello_world_example
PC $> cd hello_world_example
```

- 이 사용자 공간 예에 대한 소스 코드 파일을 작성하십시오. *hello\_world\_example.c*

```
// SPDX 식별자 : GPL-2.0
/ *
 * Copyright (C) STMicroelectronics SA 2018
 *
 * 저자 : Jean-Christophe Trotin <jean-christophe.trotin@st.com>
 *
 * /

#include <stdio.h>
#include <unistd.h>

int main ( int argc , char ** argv )
{
    int i = 11 ;

    printf ( " \n 사용자 공간 예 : STMicroelectronics의 hello world \n " );
    setbuf ( stdout , NULL );
    반면 ( I - ) {
        의 printf ( "% I ' , I );
```

- 작업 공간에 새로운 레시피 추가

```
PC $> cd <빌드 디렉토리>
PC $> devtool add myhelloworld hello_world_example /
```

- 레시피 적응

```
PC $> devtool 편집 레시피 myhelloworld
```

다음 변경 사항에 따라 레시피를 수정하십시오 (강조 표시된 라인 참조).

```
1 # recipetool로 만든 레시피
2 # 이것은 레시피의 기초이며 완전히 기능하기 위해서는 추가 편집이 필요할 수 있습니다.
3 # (편집 할 때이 설명을 자유롭게 제거하십시오.)
4
5 # 라이선스 명세서처럼 보이는 파일을 찾을 수 없습니다. 첨부 된
6 개의 설명서 및 소스 헤더를
7 확인 하고 이에 따라 LICENSE 및 LIC_FILES_CHKSUM을 설정하십시오. #
8 # 참고 : 라이선스는 적어도 시작 건물을 할 수 있도록 "CLOSED"로 설정되고있는 - 경우
9 # 이 내장되는 소프트웨어의 라이선스에 대한 정확하지 않은 # (그
10 #이 사용하기 전에 올바른 값을 지정해야 합니다) 대부분의 경우되지 않습니다
11 초기 테스트 / 개발 이외의 아무것도 # 레시피!
12 LICENSE = "CLOSED"
13 LIC_FILES_CHKSUM = ""
14
15 # SRC_URI에 대한 정보가 아직 없습니다 (외부 소스 트리 만 지정됨)
16 SRC_URI = ""
17
18 # 참고 : Makefile을 찾을 수 없으므로 수행 할 작업을 결정할 수 없습니다.
19
```

- 이 바이너리를 보드에 밀어 넣습니다.

```
PC $> devtool deploy-target -s myhelloworld root @ <보드 IP 주소>
```

- 이 사용자 공간 예제 실행

```
보드 $> / usr / bin / hello_world_example

사용자 공간 예 : STMicroelectronics의 hello world
10 9 8 6 6 4 3 2 1 0
사용자 공간 예 : STMicroelectronics의 작별
```

## 8 가지 팁

### 8.1 마운팅 포인트 생성

부트 파일 시스템 (bootfs 파티션)의 마운트 지점을 만드는 것이 목적입니다.

- 부팅 파일 시스템과 관련된 파티션 레이블을 찾습니다

```
보드 $> ls -l / dev / disk / by-partlabel /  
총 0  
lrwxrwxrwx 1 루트 루트 15 12 월 13 일 12:31 bootfs-> ../../mmcblk0p4  
lrwxrwxrwx 1 루트 루트 15 12 월 13 일 12:31 fsbl1-> ../../mmcblk0p1  
lrwxrwxrwx 1 루트 루트 15 12 월 13 일 12:31 fsbl2-> ../../mmcblk0p2  
lrwxrwxrwx 1 루트 루트 15 12 월 13 일 12:31 rootfs-> ../../mmcblk0p5  
lrwxrwxrwx 1 루트 루트 15 12 월 13 일 12:31 ssbl-> ../../mmcblk0p3  
lrwxrwxrwx 1 루트 루트 15 12 월 13 일 12:31 userfs-> ../../mmcblk0p6
```

- /boot 디렉토리의 /dev/mmcblk0p4에 있는 부트 파일 시스템을 연결하십시오.

```
보드 $> 마운트 / dev / mmcblk0p4 / boot
```