

승인 된 버전. 승인 된 날짜 : 2020 년 1 월 29 일 15:26

STM32MP1 배포 패키지

이 문서 얻고 사용하는 방법에 대해 설명 배포 패키지 의 소프트웨어를 내장 STM32MPU 의 모든 개발 플랫폼 STM32MP1 제품군 (STM32MP15 보드를 대상에 대한 수정 또는 소프트웨어의 조각을 추가하고 바로 리눅스 배포판을 생성하기 위해,) 생성물.

지식 및 개발 환경 측면에서 몇 가지 전제 조건 을 나열 하고이 패키지에 대한 STM32MPU 내장 소프트웨어 패키지를 다운로드하고 설치하는 단계별 접근 방식 을 제공합니다 .

마지막으로 소프트웨어의 일부를 업그레이드 (추가, 제거, 구성, 개선 ...)하기위한 지침을 제안합니다.

내용

- 1 배포 패키지 내용
- 2 배포 패키지 단계별 개요
- 3 전제 조건 확인
 - 3.1 지식
 - 3.2 개발 설정
- 4 스타터 패키지 설치
- 5 OpenSTLinux 배포판 설치
 - 5.1 OpenEmbedded 빌드 환경 초기화
- 6 OpenSTLinux 배포판 구축
- 7 내장 된 이미지 깜박임
- 8 부팅 순서 확인
- 9 Arm Cortex-A에서 실행되는 소프트웨어 수정
 - 9.1 레이어 생성
 - 9.2 devtool 유틸리티
 - 9.3 개발자 패키지로 개발 통합
- 10 Arm Cortex-M에서 실행되는 소프트웨어 수정
- 11 자신 만의 리눅스 배포판 만들기
- 12 자신 만의 스타터 및 개발자 패키지 생성
- 13 필수 명령에 대한 빠른 링크
- 14 참조

1 배포 패키지 내용

STM32MPU 임베디드 소프트웨어 배포판 및 해당 패키지에 아직 익숙하지 않은 경우 다음 기사를 읽으십시오.

- [귀하의 요구에 가장 적합한 STM32MPU 임베디드 소프트웨어 패키지](#) (특히 배포 패키지 장)
- [STM32MPU 임베디드 소프트웨어 배포](#)

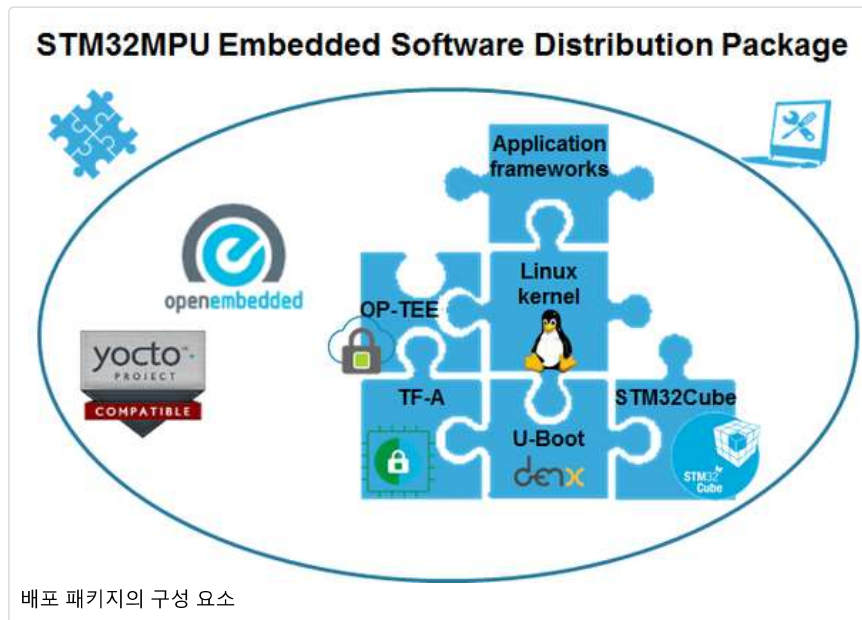
STM32MPU 임베디드 소프트웨어 배포 용 배포 패키지에 이미 익숙한 경우 [필수 명령에 대한 빠른 링크](#)가 유용 할 수 있습니다.

요약하면이 배포 패키지 는 다음을 제공합니다.

- **OpenEmbedded 기반 의 빌드 프레임 워크** (일명 배포 빌더)
- 에 대한 **OpenSTLinux 배포** (ARM 코어 텍스-A 프로세서의 개발), 소프트웨어의 모든 조각 소스 코드를 BSP를 (리눅스 커널, U-부팅, TF-A, 선택적으로 OP-TEE), 애플리케이션 프레임 워크 (WAYLAND - 웨스턴 : , GStreamer, ALSA ...)
- 에 대한 **STM32Cube MPU 패키지** (ARM 코어 텍스 M 프로세서에 대한 개발), 소프트웨어의 모든 조각 소스 코드 : BSP, HAL, 미들웨어 및 애플리케이션

① 그주의 분배 패키지 내에 **STM32CubeMP1 패키지 OpenSTLinux 분포에 통합** 큐브 프로젝트에 대한 구체적인 제법을 통해 ([조리법-확장](#) / [m4projects / m4projects-stm32mp1.bb](#) STM32는 MPU 장치 용 마이크로 일렉트로닉스 층 (들) [메타 세인트 -stm32mp](#))

- 필요에 따라 시스템을 조정하고 빌드 된 이미지를 처리하기위한 툴셋 (예 : 빌드 된 이미지를 보드에 설치하기위한 [STM32CubeProgrammer](#))



2 배포 패키지 단계별 개요

개발을 위해 STM32MPU 내장 소프트웨어 패키지를 준비하는 단계는 다음과 같습니다.

- ☐ 전제 조건 확인
- ☐ 보드 용 스타터 패키지 설치
- ☐ OpenSTLinux 배포 설치
- ☐ OpenSTLinux 부품을 구축
- ☐ 내장 된 이미지를 감박임
- ☐ 부팅 순서를 확인을

이러한 단계가 완료되면 다음을 수행 할 수 있습니다.

- Arm Cortex-A에서 실행되는 소프트웨어 수정
- Arm Cortex-M에서 실행되는 소프트웨어 수정
- 나만의 리눅스 배포판 만들기
- 자신의 스타터 및 개발자 패키지 생성

3 전제 조건 확인

3.1 지식

STM32MP1 배포 패키지는 대상 제품에 대한 Linux 배포 작성을 목표로합니다.이 패키지를 최대한 활용하려면 Linux에 대한 확실한 지식이 권장됩니다.

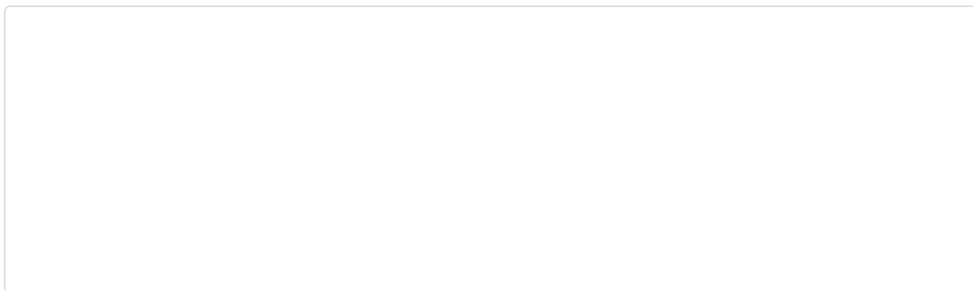
OpenSTLinux 배포판은 임베디드 Linux 용 OpenEmbedded (/ Yocto Project) 빌드 프레임 워크를 기반으로하는 Linux 배포판입니다. OpenEmbedded에 대한 간단한 소개와 표준 문서 및 교육에 대한 링크는 [OpenEmbedded](#) 기사 에서 확인할 수 있습니다 .

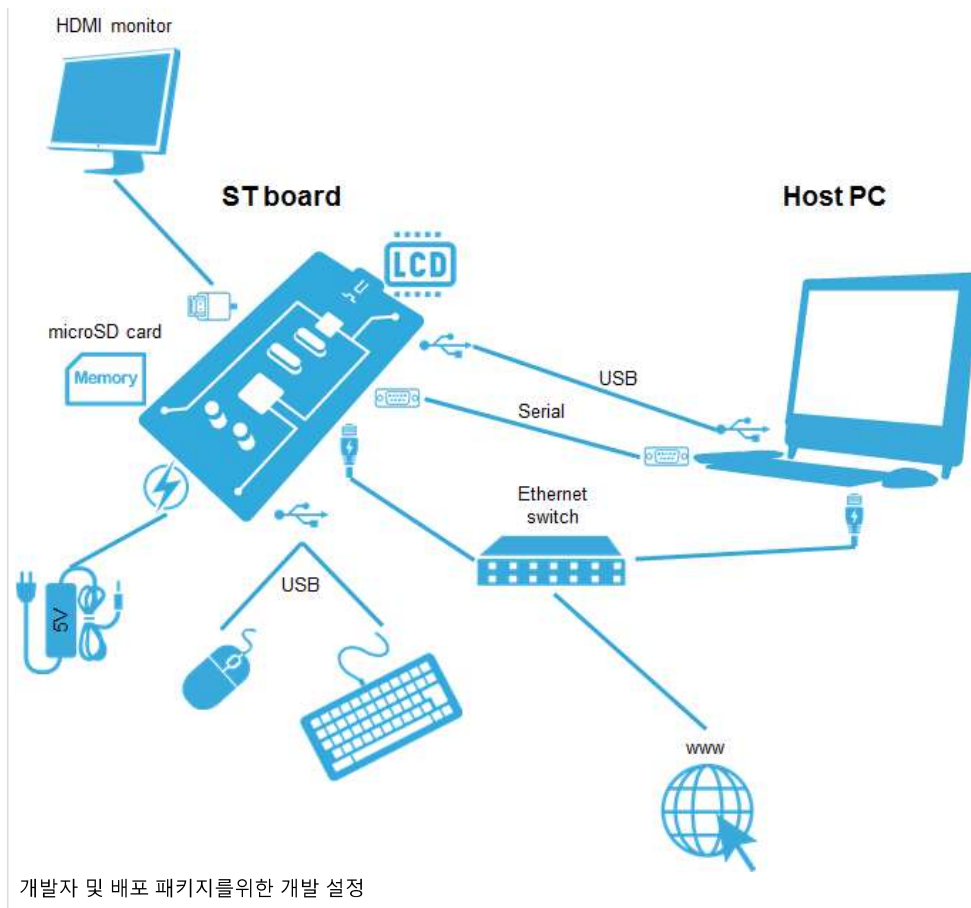
[STM32MPU 임베디드 소프트웨어 아키텍처 개요](#) 도 살펴 보는 것이 좋습니다.

3.2 개발 설정

개발 PC (호스트)에 권장되는 설정은 [PC 전제 조건](#) 문서에서 지정됩니다 .

개발 플랫폼 (보드) 및 개발 PC (호스트)가 무엇이든, 가능한 개발 설정 범위는 아래 그림과 같습니다.





다음 구성 요소는 **필수**입니다 .

- 위에서 지정한대로 크로스 컴파일 및 크로스 디버깅을위한 호스트 PC
- 관련 스타터 패키지 기사에 지정된 보드 조립 및 구성
- 소프트웨어 이미지 (바이너리)를로드하고 업데이트하는 대용량 저장 장치 (예 : microSD 카드)

다음 구성 요소는 **선택 사항**이지만 **권장**됩니다 .

- 호스트 PC ([터미널 프로그램](#)을 통해)와 트레이스 용 보드 (초기 부팅 트레이스까지) 및 원격 PC에서 보드에 대한 직렬 링크 (명령 줄)
- 로컬 네트워크를 통한 교차 개발 및 교차 디버깅을 위해 호스트 PC와 보드 간의 이더넷 링크. 이것은 직렬 (또는 USB) 링크를 대체하거나 보완하는 것입니다
- 보드에서 사용 가능한 기술에 따라 보드에 연결된 디스플레이 : DSI LCD 디스플레이, HDMI 모니터 (또는 TV) 등
- USB 포트를 통해 연결된 마우스 및 키보드

카메라, 디스플레이, JTAG, 센서, 액추에이터 등 보드의 연결 기능을 통해 **추가 옵션** 구성 요소를 추가 할 수 있습니다.

4 스타터 패키지 설치

배포 패키지를 설치하고 사용하기 전에 **Starter Package 덕분에 보드에 익숙해 져야합니다** . 스타터 패키지와 관련된 모든 기사는 [Category : Starter Package](#) : 보드에 해당하는 기사를 찾아서 설치 지침을 따르십시오 (아직 완료되지 않은 경우).

간단히 말해 다음을 의미합니다.

- 보드가 성공적으로 부팅됩니다
- 플래시 된 이미지는이 기사에서 다운로드 할 OpenSTLinux 배포판과 동일한 STM32MPU 임베드드 소프트웨어 배포판에서 가져온 것입니다.

스타터 패키지 덕분에 **모든 플래시 파티션** 이 채워집니다 .

그런 다음 배포 패키지를 사용하면 이러한 파티션을 서로 독립적으로 수정하거나 업그레이드 할 수 있습니다.

5 OpenSTLinux 배포판 설치

- STM32MP1 OpenSTLinux 배포판은 매니페스트 저장소 위치 및 매니페스트 개정판 ([openstlinux-20-02-19](#))을 통해 제공됩니다.
- 설치 는 `repo` 명령 에 의존합니다 . 호스트 PC에 Repo 도구 (Git에서 실행되는 Google 빌드 저장소 관리 도구)가 아직 설치 및 구성되어 있지 않은 경우 [PC 전제 조건](#) 기사를 참조하십시오.
- OpenSTLinux 배포판은 다양한 오픈 소스 리포지토리에서 다운로드되는 오픈 소스 소프트웨어 (OSS) 패키지를 사용하고 있습니다. 따라서 IT 인프라 프록시는 이러한 액세스를 금지하지 않아야합니다. 프록시 관련 문제가 의심되는 경우 [프록시 문제를 방지](#) 하는 [방법](#) 문서를 참조하십시오.

- STM32MP1 OpenSTLinux 배포판 설치

	STM32MP1 배포 패키지 OpenSTLinux 배포-STM32MP15-Ecosystem-v1.2.0 릴리스
설치	<ul style="list-style-type: none"> 배포 패키지를 설치할 호스트 PC 디렉토리로 이동하십시오 (<배포 패키지 설치 디렉토리>). 예를 들어, 제안을 따라 작업 디렉토리를 구성하는 경우 : <pre>\$ cd <작업 디렉토리 경로> / Distribution-Package</pre>
	<ul style="list-style-type: none"> OpenSTLinux 분배 설치 서브 디렉토리를 작성하십시오. <pre>\$ mkdir openstlinux-20-02-19 \$ cd openstlinux-20-02-19</pre>
	<ul style="list-style-type: none"> 현재 디렉토리에서 repo를 초기화하십시오 ('repo init'에 대한 자세한 내용은 here).
	<p>Ubuntu 16.04의 경우 레거시 저장소를 사용하여 여기 에 설치해야 합니다.</p> <pre>\$ repo init -u https://github.com/STMicroelectronics/oe-manifest.git -b 심판 / 태그 / openstlinux-20-02-19</pre>
	<p>참고 : 프로세스에 영향을 주지 않으면서 "repo init" 명령 중에 "ERROR 404"가 나타날 수 있습니다.</p> <ul style="list-style-type: none"> 로컬 프로젝트 디렉토리를 매니페스트에 지정된 원격 저장소와 동기화하십시오 ('repo sync'에 대한 자세한 내용은 here). <pre>\$ 저장소 동기화</pre>
	<p>참고 : 배포 패키지를 설치하려면 약 140MB가 필요합니다 (배포 패키지가 컴파일 되면 약 25GB).</p>
릴리스 노트	<p>이 소프트웨어 패키지의 내용에 대한 자세한 내용은 관련 STM32MP15 에코 시스템 릴리스 노트를 참조하십시오 .</p> <p> 이전 릴리스에 관심이있는 경우 에코 시스템 릴리스 노트의 아카이브를 살펴보세요.</p>

- OpenSTLinux 분배 설치 디렉토리는 에 <배포 패키지 설치 디렉토리>, 그리고 이름 openstlinux-20-02-19 :

openstlinux-20-02-19	OpenSTLinux 배포
레이어	
메타 오픈 임베디드	OpenEmbedded-Core 유니버스 (OpenEmbedded 표준)의 레이어 모음
meta-qt5	OpenEmbedded (표준)의 QT5 레이어
메타 세인트	
메타 세인트 openstlinux의	OpenSTLinux 분배를 위한 프레임 워크와 이미지 설정이 포함 마이크로 일렉트로닉스
메타 세인트 stm32mp의	STM32는 MPU 대한 BSP의 설명이 들어 ST 마이크로 일렉트로닉스 총 장치
recipes-bsp	
alsa	대한 조리법 ALSA의 제어 구성
드라이버	Vivante GCNANO GPU 커널 드라이버에 대한 조리법
신뢰-펌웨어-A	TF-A에 대한 조리법
u-부팅	U를 위한 조리법 -Boot
레시피 확장	
linux-examples	STM32 MPU 장치에 대한 Linux 예제 레시피
m4coredump cortex의	코어 덤프를 관리하기 위한 스크립트 레시피 M4
m4projects의	코어 텍스 M4 펌웨어 사례에 대한 조리법
조리법 그래픽	
Vivante	라이브러리 OpenGL ES, OpenVG 및 EGL (멀티 백엔드)을 위한 gcnano-userland 레시피

5.1 OpenEmbedded 빌드 환경 초기화

OpenEmbedded 환경 설정 스크립트는 BitBake 또는 devtool 도구를 사용하는 각각의 새 작업 터미널에서 한 번만 실행해야 합니다 (나중 참조).

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

STM32MP1 용 BSP는 [소프트웨어 라이선스 계약\(SLA\)](#) 이 적용되는 패키지 및 펌웨어에 따라 다릅니다 . 이 EULA를 읽고 동의하라는 메시지가 표시됩니다.

참고 :

- *openstlinux - 웨스턴* (OpenSTLinux의 웨스턴 / WAYLAND을 갖춘 유통)과 *stm32mp1* (모든 STM32MP1 하드웨어 구성에 대한 시스템 구성)이있는 기본 값 배포판 및 기계
- 스타터 패키지 (이미지) 및 개발자 패키지 (SDK ...) 용 소프트웨어 패키지가 구성으로 구축되었습니다.
- *DISTRO* 및 *MACHINE*의 다른 값 은 [OpenSTLinux 배포](#) 에서 제안됩니다.

무엇보다도 환경 설정 스크립트는 *build- <distro>-<machine>* 디렉토리 (여기서 **build-openstlinuxweston-stm32mp1**) 라는 **빌드 디렉토리**를 작성합니다 . 스크립트가 실행 된 후 현재 작업 디렉토리가 빌드 디렉토리로 설정됩니다. 나중에 빌드가 완료되면 빌드 중에 작성된 모든 파일이 포함됩니다.

로컬 구성 파일 (*build- <distro>-<machine> /conf/local.conf*)에는 모든 로컬 사용자 설정이 포함됩니다. 레이어 구성 파일 (*build- <distro>-<machine> /conf/bblayers.conf*)은 빌드 중에 고려해야 할 레이어를 BitBake에 알려줍니다.

```
openstlinux-[...]          OpenSTLinux 배포
├─ build- <distro>-<machine> 빌드 디렉토리
│ └─ conf
│   ├── bblayers.conf      로컬 구성 파일
│   ├── local.conf         레이어 구성 파일
│   └─ [...]
└─ 작업 공간
   ├── 레이어
   ├── 메타 오픈
   └─ [...]
```

6 OpenSTLinux 배포판 구축



환경 설정 스크립트가 실행되어 있어야 합니다

bitbake <이미지> 명령은 이미지를 구축하는 데 사용됩니다. *<image>* 는 대상 이미지 *st-image-weston* 을 지정 합니다 (기본 Wayland를 지원하는 OpenSTLinux의 경우 웨스턴 이미지).

```
PC $> 비트 베이킹 st-image-weston
```

BitBake는 Yocto 프로젝트의 핵심 구성 요소이며 OpenEmbedded 빌드 시스템에서 이미지를 빌드하는 데 사용됩니다. 이 빌드 엔진은 레시피 (.bb), 구성 (.conf) 및 클래스 (.bbclass) 파일에 저장된 메타 데이터에 따라 셸 및 Python 태스크를 실행합니다. [BitBake 치트 시트](#) 문서를 소개합니다 일부 BitBake 명령 줄 옵션은, 곧 레시피 (.bb)의 구문을 설명하고 표준 문서 대한 링크를 제공합니다.

참고 :

- 이 *st-image-weston* 이미지 용 스타터 패키지 (이미지) 및 개발자 패키지 (SDK ...) 용 소프트웨어 패키지가 빌드되었습니다.
- *<image>* 에 대한 다른 값 은 [OpenSTLinux 배포](#) 에서 예로 제안됩니다 (*DISTRO* 에 대해 선택된 값과 *<image>* 에 대한 값 사이의 호환성을주의하십시오)

기본적으로 모든 BitBake 작업은 빌드 디렉토리에서 수행됩니다 (Yocto Project Mega-Manual ^[1] 의 빌드 디렉토리 구조에 대한 자세한 정보 참조).

7 내장 이미지 점멸

으로 *Build- <배포판> - <컴퓨터> / tmp 를-의 glibc / 배포 / 이미지 / stm32mp1* 의 디렉토리는 전체 파일 시스템 이미지를받습니다.

디렉토리 구조 빌드 : 이미지

높히다

배포 패키지를 빌드하면 OP-TEE에 대한 이미지와 플래시 레이아웃 파일이 생성됩니다 (위의 "빌드 디렉토리 구조 : 이미지"를 확장 하여이 파일 **optee* *를보십시오/오):이 구성에 관심이있는 경우 다음을 참조하십시오. 어떻게 채우고 OP-TEE와 보드를 부팅합니다.

STM32CubeProgrammer 도구는 이미지가 내장 된 STM32MP15 보드를 플래시하는 데 사용됩니다. 이 도구는 보드와 관련된 스타터 패키지를 통해 설치되었다고 가정합니다.

보드에 따라 여러 플래시 장치 (microSD, eMMC ...)를 사용할 수 있습니다. 이 기사의 나머지 부분에서 microSD 카드는 플래시 장치로 간주됩니다.

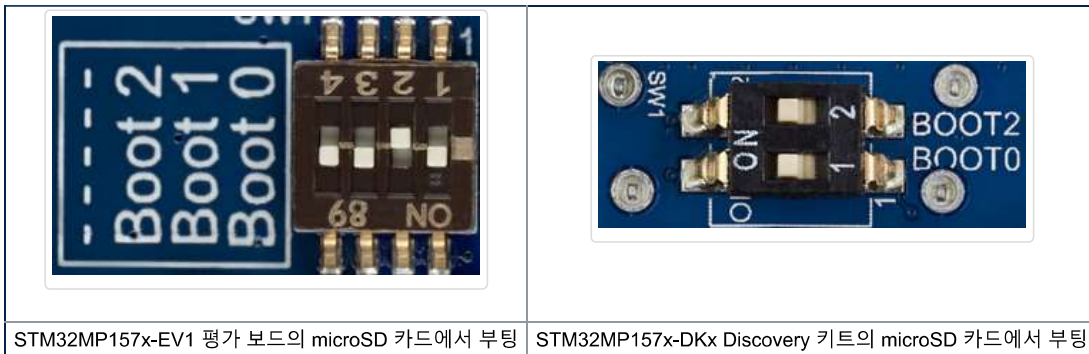
보드를 플래싱하는 절차는 보드에 해당하는 스타터 패키지 기사에 설명되어 있습니다 (스타터 패키지와 관련된 모든 기사는 범주 : 스타터 패키지에 있음).

- [STM32MP157x-EV1 평가 보드에서 내장 이미지 플래싱](#)
- [STM32MP157x-DKx Discovery 키트에서 내장 이미지 플래싱](#)

플래시가 종료되면 이미지가 플래시 된 플래시 장치 (예 : microSD 카드)가 부팅 소스로 선택되도록 부팅 스위치를 구성해야 합니다.

- [STM32MP157x-EV1 평가 보드의 부팅 스위치](#)
- [STM32MP157x-DKx Discovery 키트의 부팅 스위치](#)

아래 그림 은 다른 보드의 **microSD 카드** 에서 **부팅** 하기위한 부팅 스위치 구성을 보여줍니다 .



8 부팅 순서 확인

보드의 스타터 패키지 기사에 설명 된 것과 동일한 기본 명령을 실행하여 빌드를 확인할 수 있습니다 : [Category : Starter Package](#) .

9 Arm Cortex-A에서 실행되는 소프트웨어 수정

OpenSTLinux 배포가 설치 및 구축되었으므로 표준 OpenEmbedded 메소드 (예 : 계층 작성) 및 유틸리티 (예 : *devtool*) 덕분에이를 사용자 정의하십시오 .

9.1 레이어 생성

OpenSTLinux 배포판 위에 개발 된 소프트웨어 패키지는 하나 이상의 사용자 정의 계층에 추가되어야 합니다. 같은 방식으로 OpenSTLinux 배포의 계층에 이미 존재하는 패키지의 수정 (애드온, 개선, 사용자 정의)은 특히 유지 관리를 쉽게하기 위해 사용자 정의 계층에 추가해야 합니다.

새로운 개방형 임베디드 레이어 생성하는 방법 문서 작성하고 이러한 사용자 층 (들)을 관리 할 수있는 여러 가지 방법을 제안한다.

9.2 devtool 유틸리티

OpenEmbedded는 임베디드 장치 용 Linux 배포판을 만드는 데 사용되는 빌드 프레임 워크이며 원래 개발 환경이 아닙니다. *devtool* 이라는 OpenEmbedded 컴패니언 유틸리티 는 OpenEmbedded 빌드 시스템을 사용하여 빌드 된 이미지에 통합 될 코드를 개발, 빌드 및 테스트하는 데 도움이됩니다.

OpenEmbedded - devtool의 기사를 소개에게 배포의 새로운 응용 프로그램이나 라이브러리를 추가, 수정 및 통합 할 수있는 방법들이 도구에 대한 개요를 제공합니다.

다음 기사는 "개발자 패키지로 크로스 컴파일하는 방법"의 배포 패키지 컨텍스트와 동일한 예제를 설명합니다.

- [배포 패키지를 사용하여 크로스 컴파일하는 방법](#)

9.3 개발자 패키지를 이용한 개발 통합

Yocto 빌드 프로세스의 이전 장 9.2에 설명 된 일부 개발을 통합하기 위해 다음과 같은 방법을 설명합니다.

- [고객 응용 프로그램을 추가하는 방법](#)
- [Linux 커널을 사용자 정의하는 방법](#)

10 Arm Cortex-M에서 실행되는 소프트웨어 수정

STM32Cube MPU 패키지가 Yocto 프로젝트와 함께 STM32MPU 임베디드 소프트웨어 배포에 통합되는 방법을 이해하려면 STM32CubeMP1 용 배포 패키지를 참조하십시오.

다음과 같은 경우에 유용 할 수 있습니다.

- 자신의 배포판에서 STM32Cube MPU 패키지를 제거하십시오.
- 자체 배포를 위해 자체 큐브 Cortex-M 프로젝트 설치
- ST가 직접 배포 한 STM32Cube MPU 패키지 수정

11 나만의 리눅스 배포판 만들기

먼저 다음 기사를 통해 하드웨어 수정에 맞게 소프트웨어를 사용자 정의 할 수 있습니다.

자신의 배포판을 만드는 방법 은 웨이 랜드 또는 x11과 같은 일부 프레임 워크와 alsa, nfs, wifi, bluetooth, ipv6 등과 같은 일부 기능을 활성화하려는 경우 다음 단계를 제공합니다.

그런 다음 누락 된 사용자 도구 (예 : libdrm, evtests, iptables, i2c-tools 등)를 추가하기 위해 기존 이미지를 사용자 정의 할 수 있습니다. 이 기사 : 자신만의 이미지를 만드는 방법 으로 이동해야 합니다.

12 자신만의 스타터 및 개발자 패키지 생성

이제 OpenSTLinux 배포가 수정되었거나 새 배포가 생성되었으므로 개발자에게 새 배포의 추가 기능 또는 향상된 기능이 풍부한 새로운 스타터 및 개발자 패키지를 제공해야 합니다.

전제 조건 :

- 여기에서는 devtool update-recipe 덕분에 레시피가 소스 파일의 변경 사항을 반영하는 패치로 업데이트되었다고 가정합니다. *명령*
- 그래서, 원래의 조리법 (중 *.bb*가) 변경, 또는 자신 층의 조리법으로 수정 된 (새로운 개방형 포함 된 레이어를 만드는 방법) 변경과 수정, 그래서 사용자 정의 조리법 (한 *.bbappend*)과 패치는 원래 패치와 명확하게 격리되어 있습니다 (유지 관리를 용이하게 하기 위해 권장되는 방법).
- 환경 설정 스크립트를 실행하십시오. 에 대한 선택 배포판 및 기계, 값이있는 당신이 초보 및 개발자 패키지 (참조 생성 할 OpenSTLinux 분배 가능한 모든 값을)

```
PC $> DISTRO = <distro> MACHINE = <machine> 소스 레이어 /meta-st/scripts/envsetup.sh
예 :
PC $> DISTRO = openstlinux-weston MACHINE = stm32mp1 소스 레이어 /meta-st/scripts/envsetup.sh
```

- 수정 으로 Build- <배포판> - <컴퓨터> /conf/local.conf 를 사용하도록 구성되어 조리법 아카이버 수 있도록 파일; 개발자 패키지 (Linux 커널, U-Boot, TF-A)를 위한 "소스 코드"소프트웨어 패키지를 생성하는 것이 목적입니다.

```
ST_ARCHIVER_ENABLE = "1"
```



자체 스타터 및 개발자 패키지를 생성하지 않는 경우이 구성을 사용하지 않도록 주의하십시오.

- 배포에 통합 된 마지막 개발이 이미지에 반영되도록하고 개발자 패키지의 "소스 코드"소프트웨어 패키지를 생성하기 위해 이미지를 빌드하십시오. 에 대한 선택 <이미지>, 당신은 초보 및 개발자 패키지를 생성하려는 값 (참조 OpenSTLinux 분배 가능한 모든 값을)

```
PC $> 비트 베이크 <이미지>
예 :
PC $> 비트 베이크 st-image-weston
```

- 스타터 패키지의 이미지 (바이너리)는 build- <DISTRO>-<MACHINE> / tmp-glibc / deploy / images / stm32mp1 디렉토리에 있습니다.

이 images / stm32mp1 디렉토리의 구조는 Starter Package 의 디렉토리 구조와 분명히 유사합니다.


- Developer Package 소프트웨어 패키지 (Linux 커널, U-Boot 및 TF-A)의 "소스 코드"는 build- <distro>-<machine> / tmp-glibc / deploy / sources / arm-openstlinux_weston-linux에서 사용 가능합니다. -gnueabi 디렉토리

이 sources / arm-openstlinux_weston-linux-gnueabi 디렉토리의 구조는 TF-A, Linux 커널 및 U-Boot 용 개발자 패키지 의 디렉토리 구조와 분명히 유사합니다.

디렉토리 구조 빌드 : 소스

넓히다


- 선택적으로 다른 "소스 코드"디렉토리를 압축하여 개발자 패키지 소프트웨어 패키지를 가져 오십시오.
- [OpenSTLinux 배포 용 SDK를 만드는 방법](#) 기사 [를 사용](#) 하여 개발자 패키지 용 새 소프트웨어 개발 키트 (SDK)를 마지막 요소 [로 생성](#) 하십시오.



SDK 항목은 64 비트 아키텍처 호스트 PC 용으로 만 빌드 할 수 있습니다.

13 필수 명령에 대한 빠른 링크

STM32MPU 내장 소프트웨어 배포 용 배포 패키지에 이미 익숙한 경우 필수 명령에 대한 빠른 링크가 아래에 나열되어 있습니다.



아래 링크를 사용하면 다른 기사로 리디렉션됩니다. 이 빠른 링크로 돌아가려면 브라우저 의 *뒤로* 버튼을 사용하십시오

명령에 대한 링크
STM32MP157x-EV1 평가 보드 스타터 패키지의 필수 명령
STM32MP157x-DKx Discovery 키트 스타터 패키지의 필수 명령
STM32MP1 OpenSTLinux 배포판 설치
OpenEmbedded 환경 설정 스크립트를 실행하십시오.
빌드 일 - 이미지 - 웨스턴의 이미지를
STM32MP157x-EV1 평가 보드의 microSD 카드에서 이미지를 플래시
STM32MP157x-DKx Discovery 키트의 microSD 카드에서 이미지를 플래시
OpenEmbedded 배포판으로 개발 (devtool)
배포에 STM32Cube MPU 패키지 개발 통합

14 참조

1. [옥토 프로젝트는 메가 설명서 - 빌드 디렉토리](#)