FUJITSU

shaping tomorrow with you

Proces startu platformy na podstawie implementacji UEFI na procesory ARM.

Autor: Tomasz Marciniak.

Agenda

FUJITSU

- Od BIOS-u do UEFI
- Architektura sprzętu.
- Od wektora startowego do systemu operacyjnego.
- Implementacja UEFI – tianocore
- UEFI – jak to jest zbudowane.
- EFI Shell
- BeagleBone z UEFI

Od BIOS do UEFI

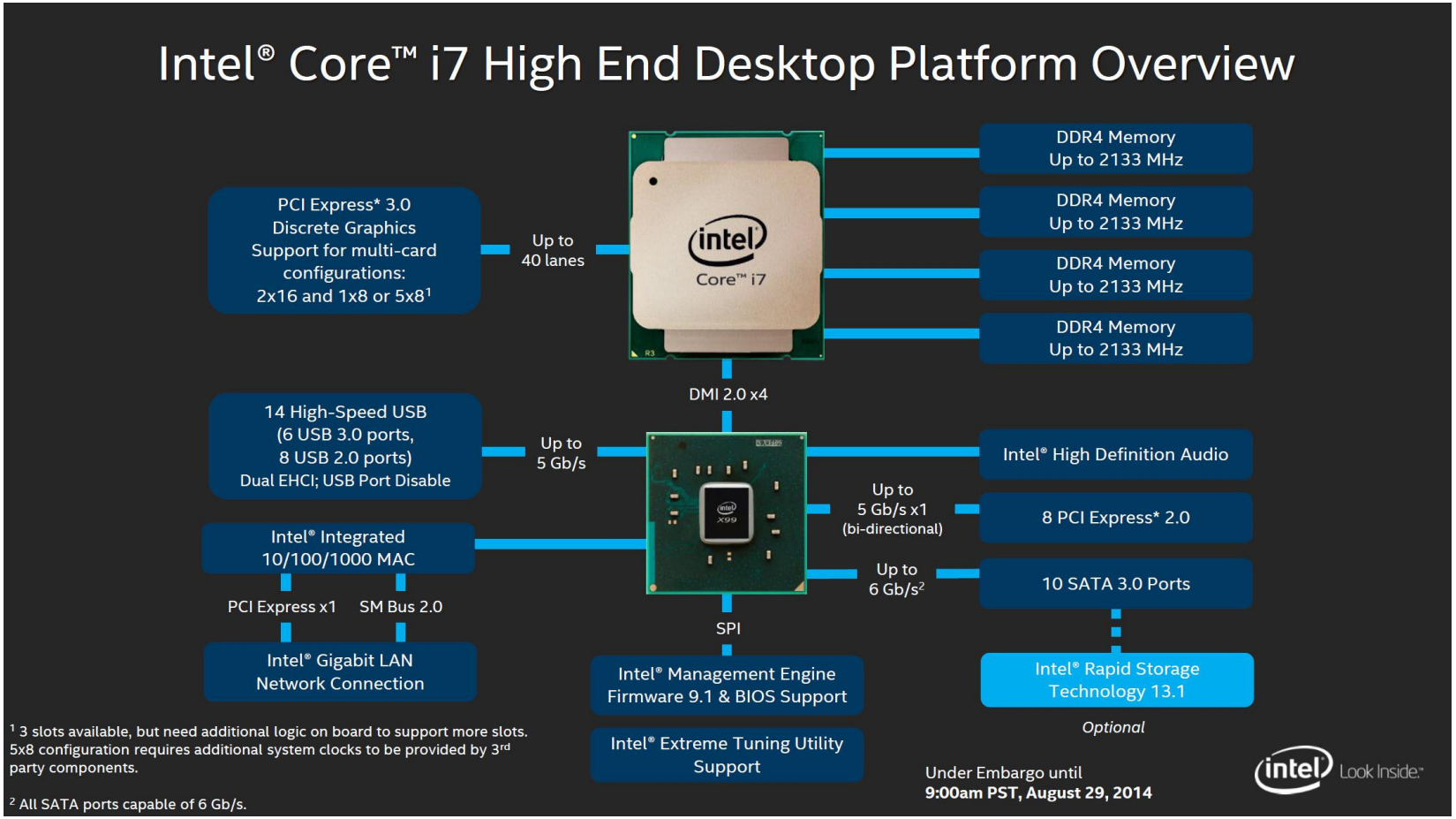
- Przed rokiem 2000 każdy BIOS był własnością producenta płyty.
- W 2000 roku Intel wydał pierwszą wersję EFI wraz z przykładową implementacją na licencji BSD
- Tianocore.org zostało założone w 2004 roku, zrzesza ono społeczność Open Source wokół EFI
- W 2005 roku, celu standaryzacji powstało UEFI (United Extensible Firmware Interface). Na początku składało się z 11 członków.
- Obecnie zrzesza cały przemysł wokół platform x86 oraz część producentów platform ARM.

UEFI

FUJITSU

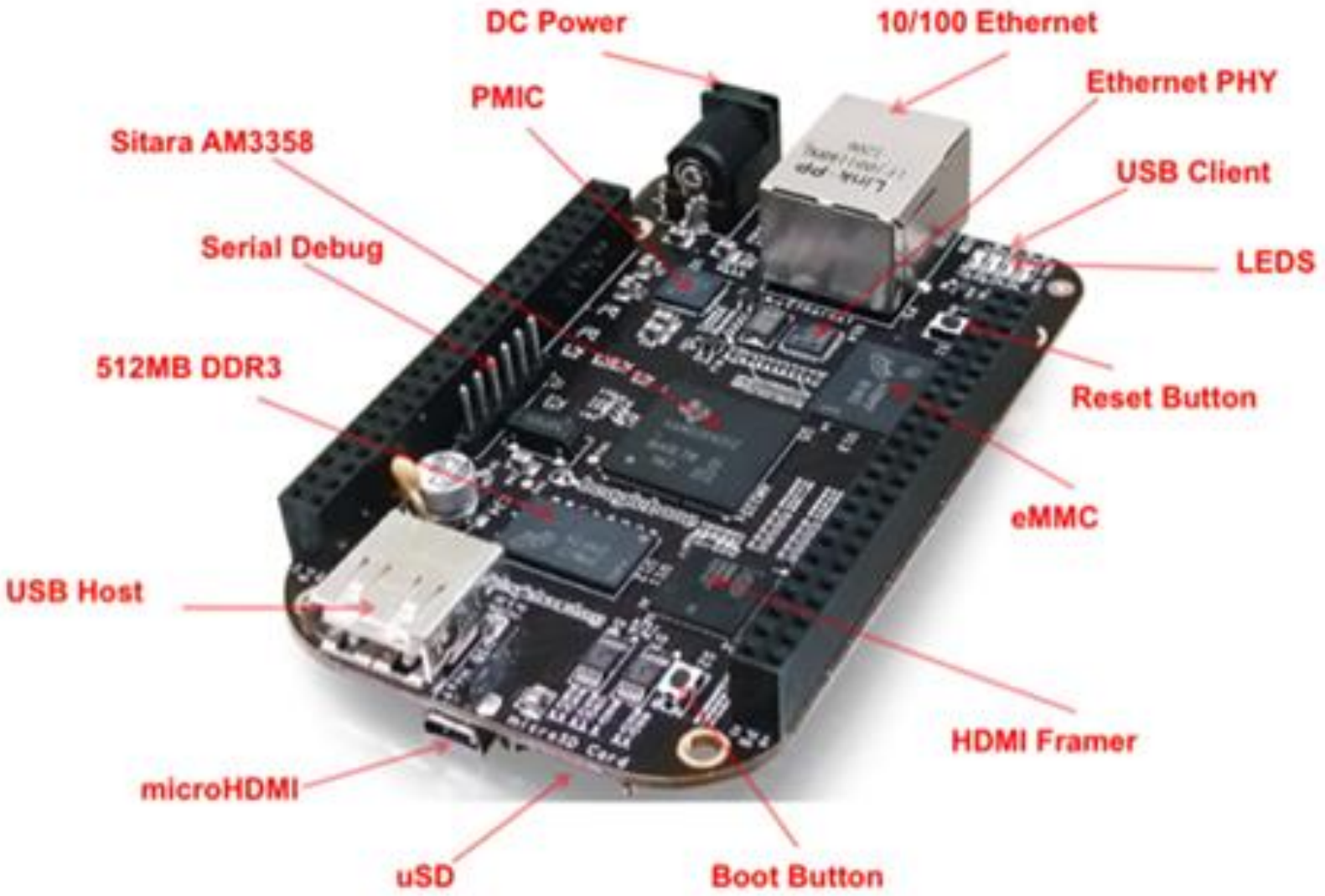
- Unified Extensible Firmware Interface.
- Firmware płyty głównej kooperujący z firmware innych urządzeń peryferyjnych.
- Jedyna wspierana przez producentów sprzętu metoda by wystartować system operacyjny na platformie x86.
- Wersję dla ARM wspiera obecnie Linaro.

Nowoczesna Architektura x86

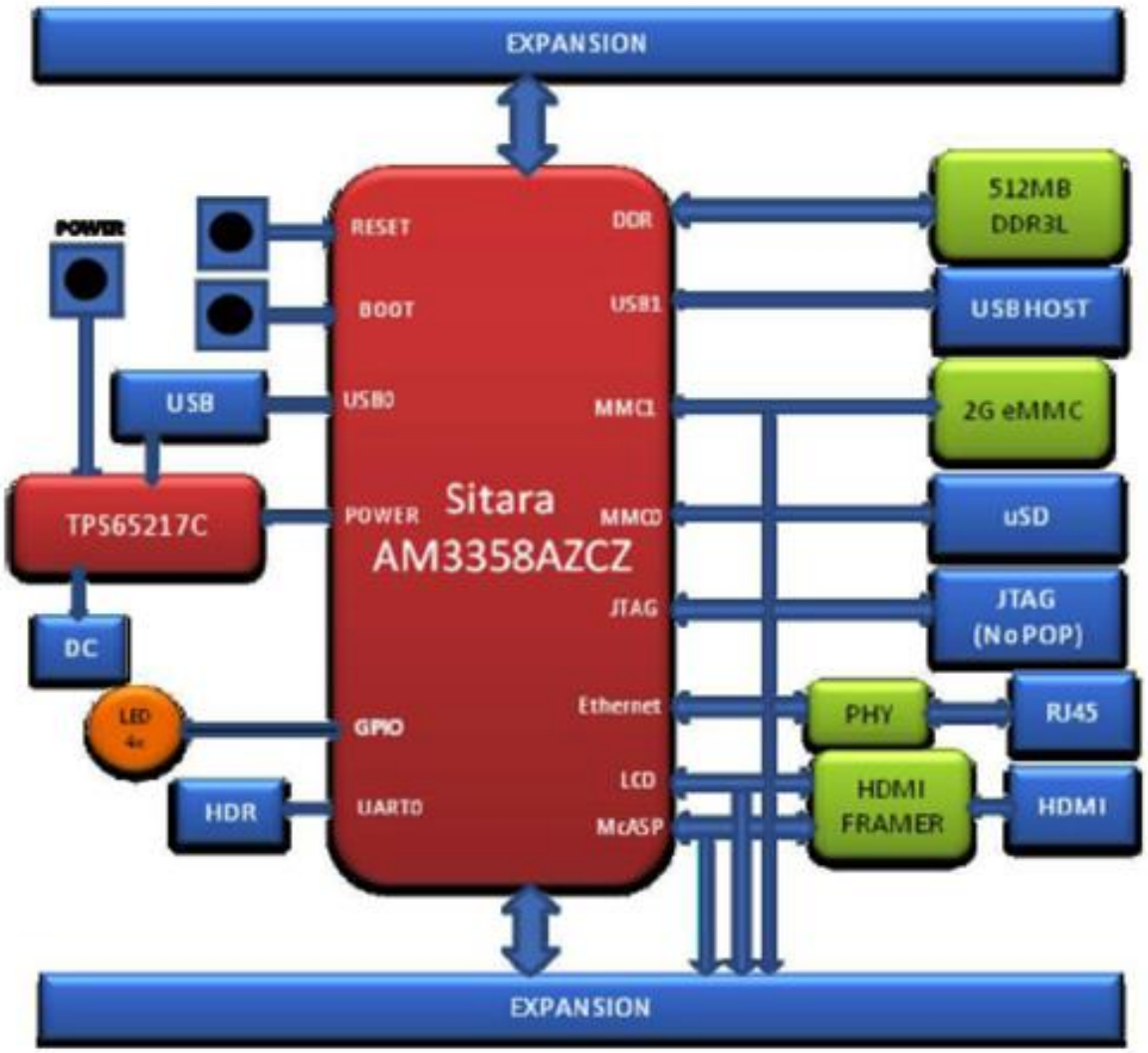


Architektura BeagleBone

FUJITSU



Architektura BeagleBone



Etapy uruchamiania platformy

FUJITSU

1)SEC – Security.

2)PEI – Pre EFI Initialization.

3)DXE – Driver Execution Environment.

4)BDS – Boot Device Select.

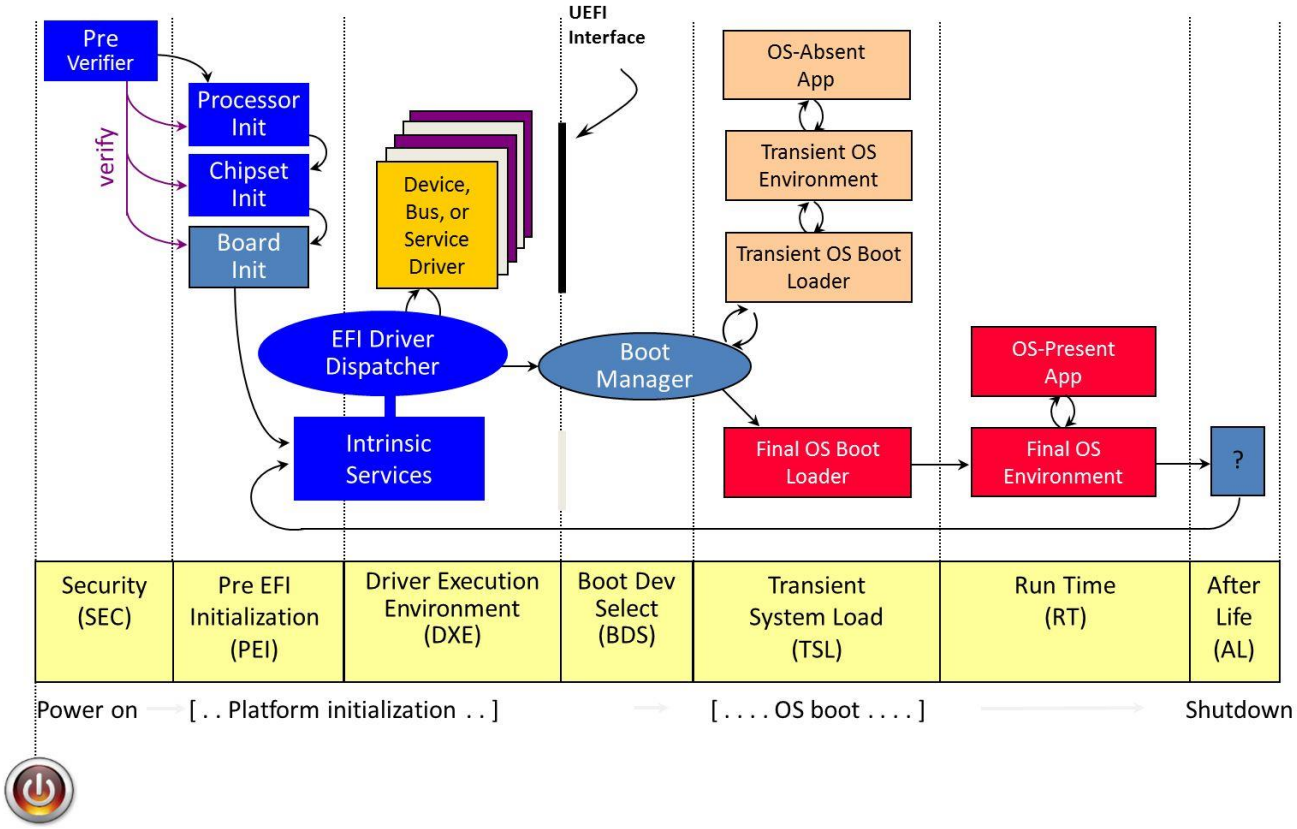
5)Handshake.

6)RT – Runtime.

Platform Initialization

- SEC oraz PEI
- Inicjalizacja CPU.
- Inicjalizacja Chipsetu.
- Załadowanie mikrokodu.
- Inicjalizacja całej płyty.
- Wczesna inicjalizacja pamięci.
- Driver eXecution Environment – DXE.
- Inicjalizacja środowiska do uruchomienia sterowników UEFI.

Platform Initialization (PI) Boot Phases



PEI

- Wczesna inicjalizacja platformy od CPU do pamięci operacyjnej.
- W większości wypadków dzieje się przed pojawieniem się obrazu na monitorze.
- Możliwy jest dostęp do PCI.

DXE

- Przygotowanie środowiska sterowników UEFI.
- Wszystkie interfejsy PEI zostają usunięte.
- Stan systemu jest przekazywany z PEI w postaci HOB-ów (Hand Off Block).
- To w tej fazie następuje właściwa inicjalizacja platformy.

DXE

- DXE Core – standardowy zestaw serwisów niezbędnych do wystartowania jak i działania systemu: Boot Services, Runtime Services.
- DXE Dispatcher – Ładuje sterowniki DXE w odpowiedniej kolejności.
- DXE Driver – Odpowiedzialne za inicjalizację procesora, chipsetu oraz komponentów platformy.

Boot Device Select

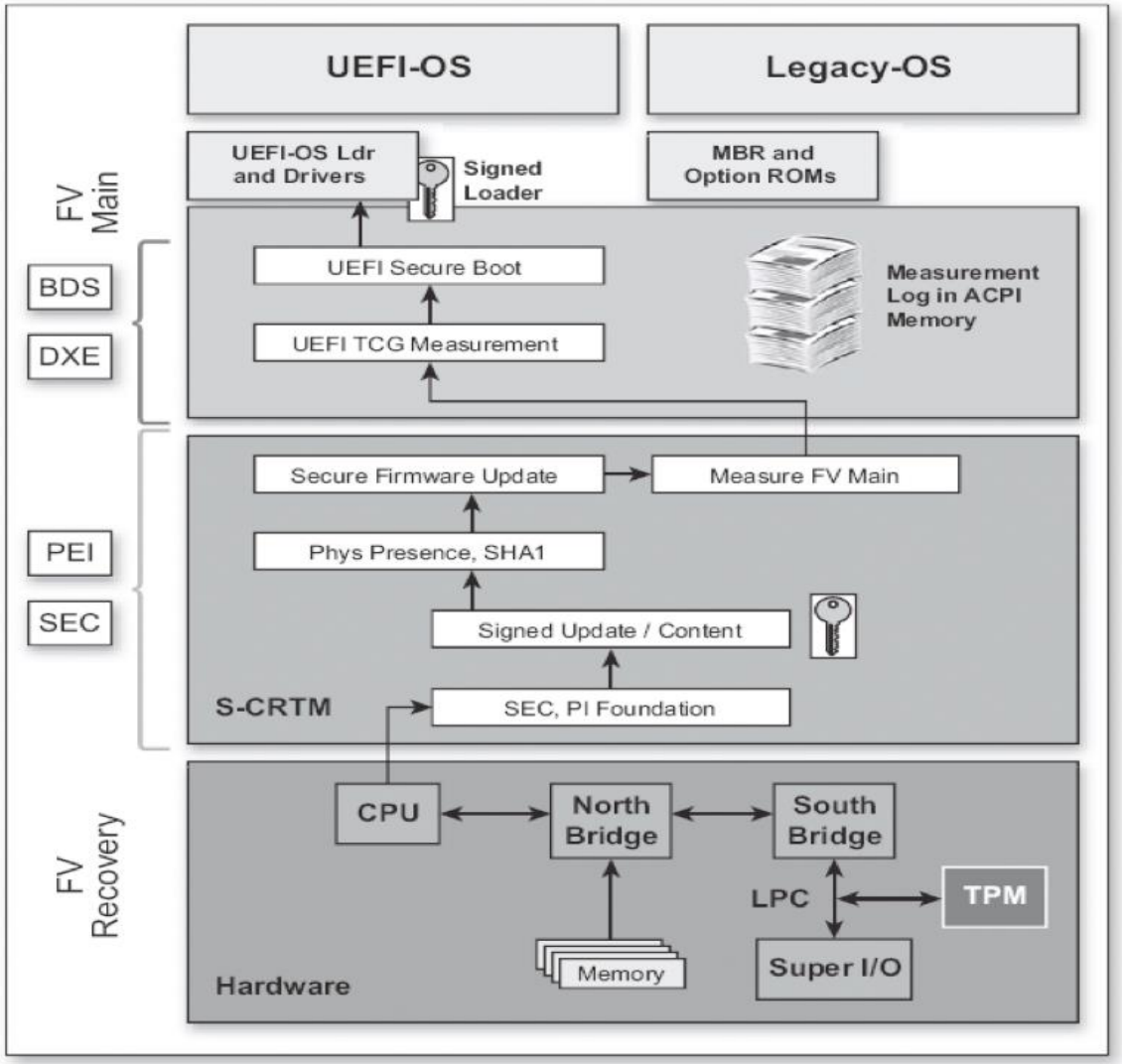
- Wybierane są urządzenia które mogą startować.

- Następuje decyzja o kolejności startowania urządzeń.

Run Time

- Udostępnia serwisy oraz sterowniki potrzebne do poprawnego działania systemu.
- Pozwala odkonywać autoryzowanych zmian w ustawieniach platofrmy poprzez dedykowany interfejs.
- Pozwala dokonać aktualizacji oprogramowania UEFI na płycie głównej.

Schemat Systemu UEFI



EFI Development Kit

FUJITSU

- EDK2 to obecny standard udostępniony w wersji otwartoźródłowej.
- https://github.com/tianocore/edk2
- http://www.tianocore.org/edk2/
- Najnowsza wersja to EDK 2.7

Programowanie w UEFI

FUJITSU

- EDK2 jest otwartoźródłową implementacją UEFI, udostępnioną przez Intel by wspomagać wdrażanie UEFI jako standardu przemysłowego.

- Najczęściej używanym elementem jest Module Development Environment (MDE). Zawiera on podstawowe biblioteki oraz Protokoły UEFI.

EFI Protocol

FUJITSU

- Protokoły to specjalne interfejsy dołączane do obrazów urządzeń.
- Protokoły pozwalają na zbudowanie interfejsu dla skomplikowanego urządzenia podłączonego do platformy.
- Protokół to struktura wskaźników do funkcji oraz czasami danych z nimi powiązanych.
- Protokół identyfikowany jest poprzez GUID.

Obraz UEFI

- Podstawową jednostką w oprogramowaniu EFI jest obraz(Image). Zawiera on nagłówek PE/COFF. Może być sterownikiem lub aplikacją.

- Dla danego obrazu może być zainstalowancyh wiele protokołów. Składają się one na obraz urządzenia UEFI.

EFI Shell

FUJITSU

- Efi Shell to interaktywne środowisko uruchomieniowe pozwalające na uruchomienie aplikacji EFI. (Np. Boot loadera grub64.efi).
- Implementuje podstawowy interfejs Shell-owy.
- Ma dostęp do większości urządzeń, sterowników I protokołów UEFI dostępnych również dla systemu operacyjnego.

BeagleBone z UEFI

FUJITSU

- Potrzeba jedynie kilku kroków by zbudować gotowy obraz UEFI dla BeagleBone, wbudować go do obrazu karty sd i sprawić by działał.
- Na maszynie z Ubuntu instalujemy gcc-arm-linux-gnueabi.
- Podążamy za aktualną wersją jak budować na linuxie:
- https://github.com/tianocore/tianocore.github.io/wiki/Using-EDK-II-with-Native-GCC
- https://github.com/tianocore/tianocore.github.io/wiki/Common-instructions

BeagleBone z UEFI

FUJITSU

- Tutaj niestety zaczynają się schody. Potrzebne definicje są poza jedną, prefixu dla kompilatora arm.
- Dla tego trzeba dodać kilka linii do pliku definicji narzędzi autowygenerowanego komendą: source edksetup.sh

Conf/target.txt

```
ACTIVE_PLATFORM       = BeagleBoardPkg/BeagleBoardPkg.dsc
TOOL_CHAIN_TAG        = GCC5
TARGET_ARCH           = ARM
```

BeagleBone z UEFI

FUJITSU

- Ttrzeba dodać kilka linii do pliku definicji narzędzi

Conf/tools_def.txt

Pod linią

DEFINE GCC5_X64_PREFIX = ENV(GCC5_BIN)

Dodajemy następną:

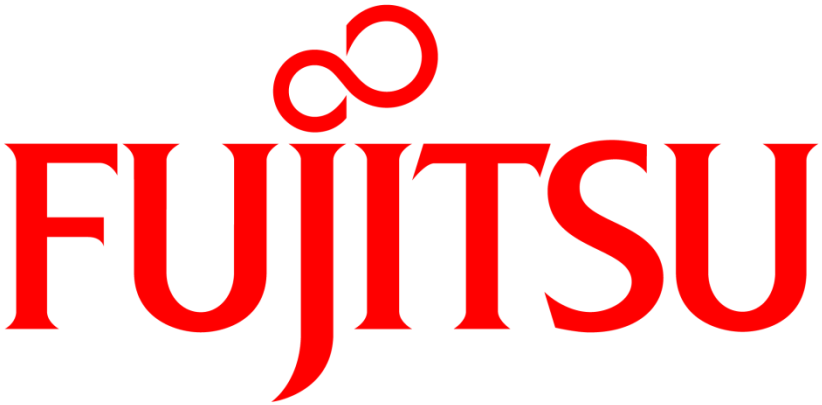
DEFINE GCC5_ARM_PREFIX = arm-linux-gnueabi-

Budujemy to przy pomocy komendy build wywołanej w katalogu głównym edk2

Jak uruchomić

FUJITSU

- Wystarczy podmienić w obrazie plik u-boot.bin w bootloaderze i powinno działać.
- Argumenty uruchomienia kernela nalezy skopiować z konfiguracji u-boota.



shaping tomorrow with you