Nagyteljesítményű mikrovezérlők

6. CMSIS és Fejlesztőkörnyezetek

Scherer Balázs

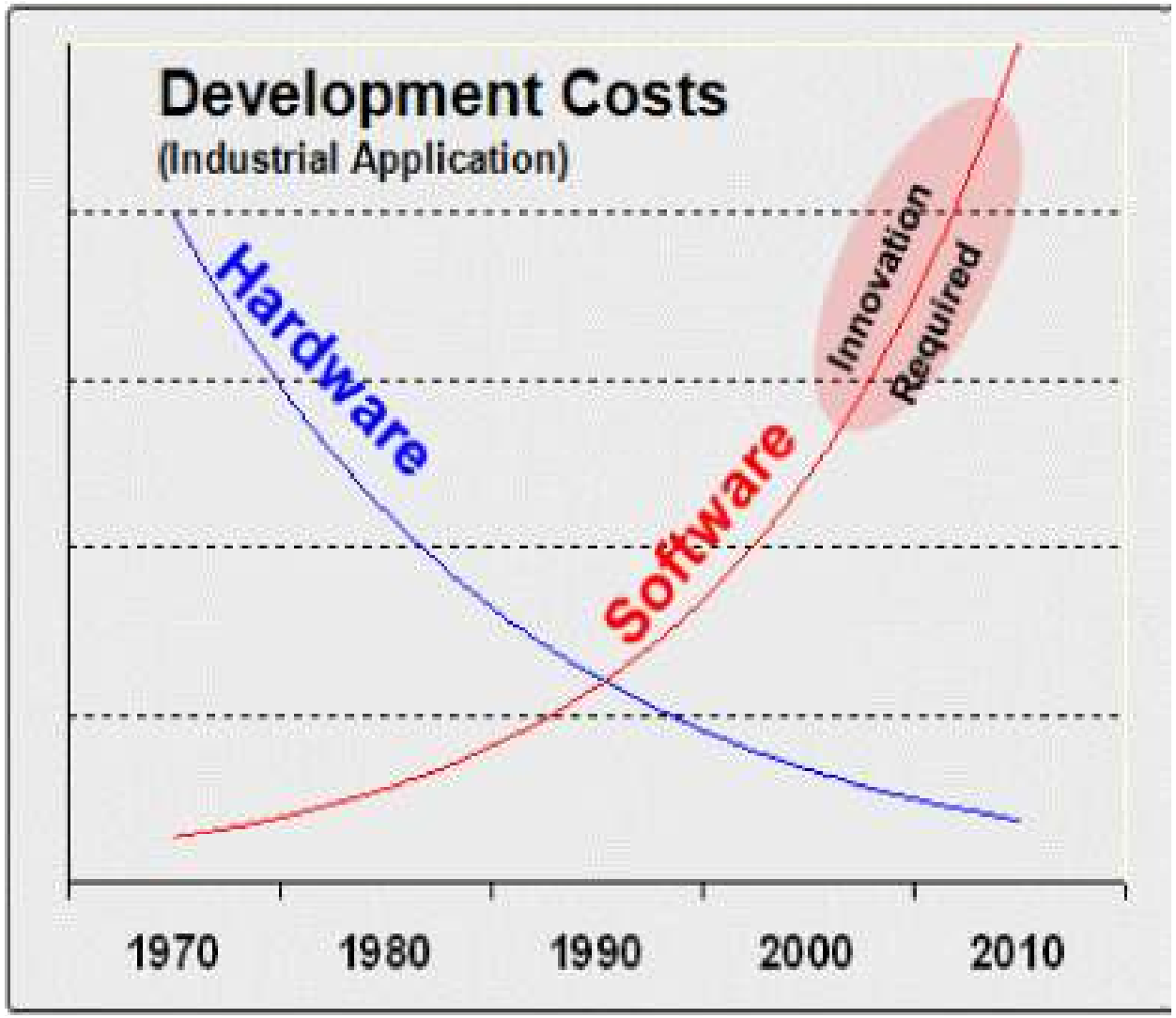
Méréstechnika és
Információs Rendszerek
Tanszék

CMSIS

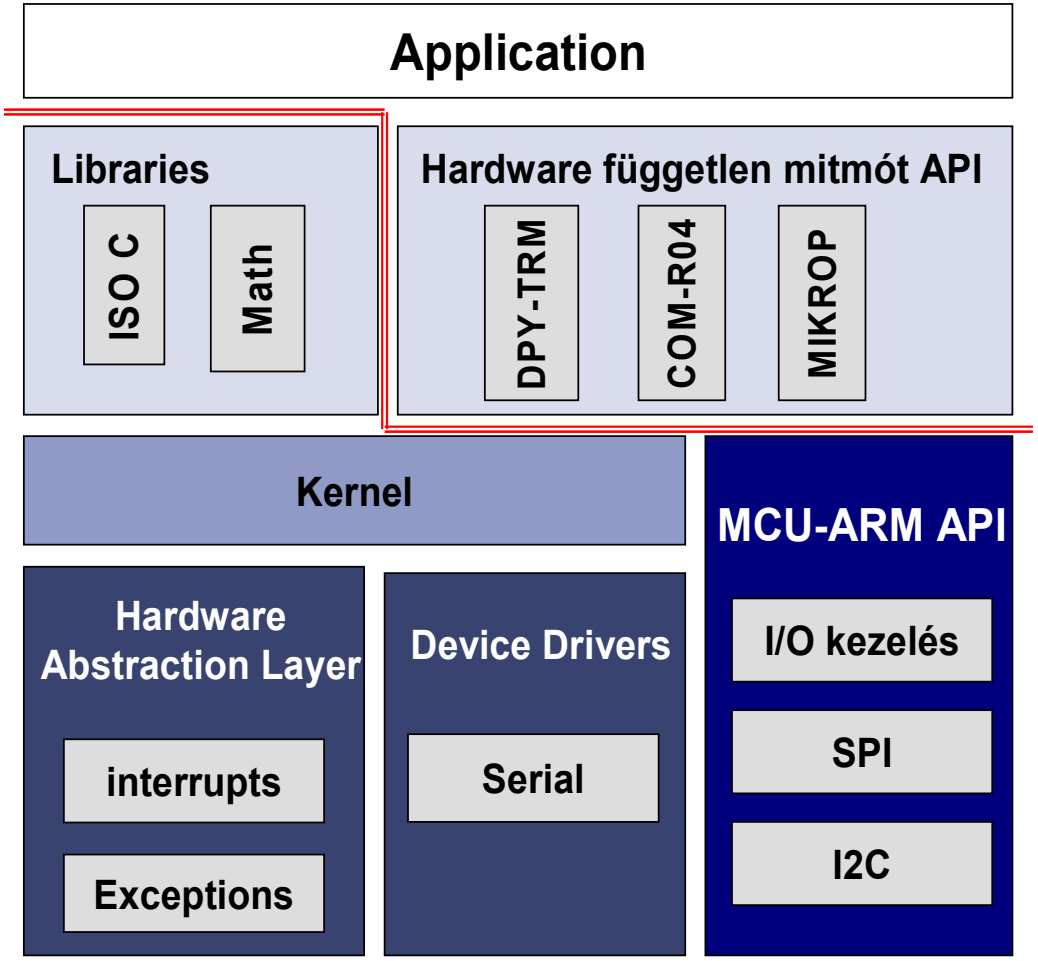
Cortex Microcontroller

Software Interface Standard

Fejlesztési költségek alakulása



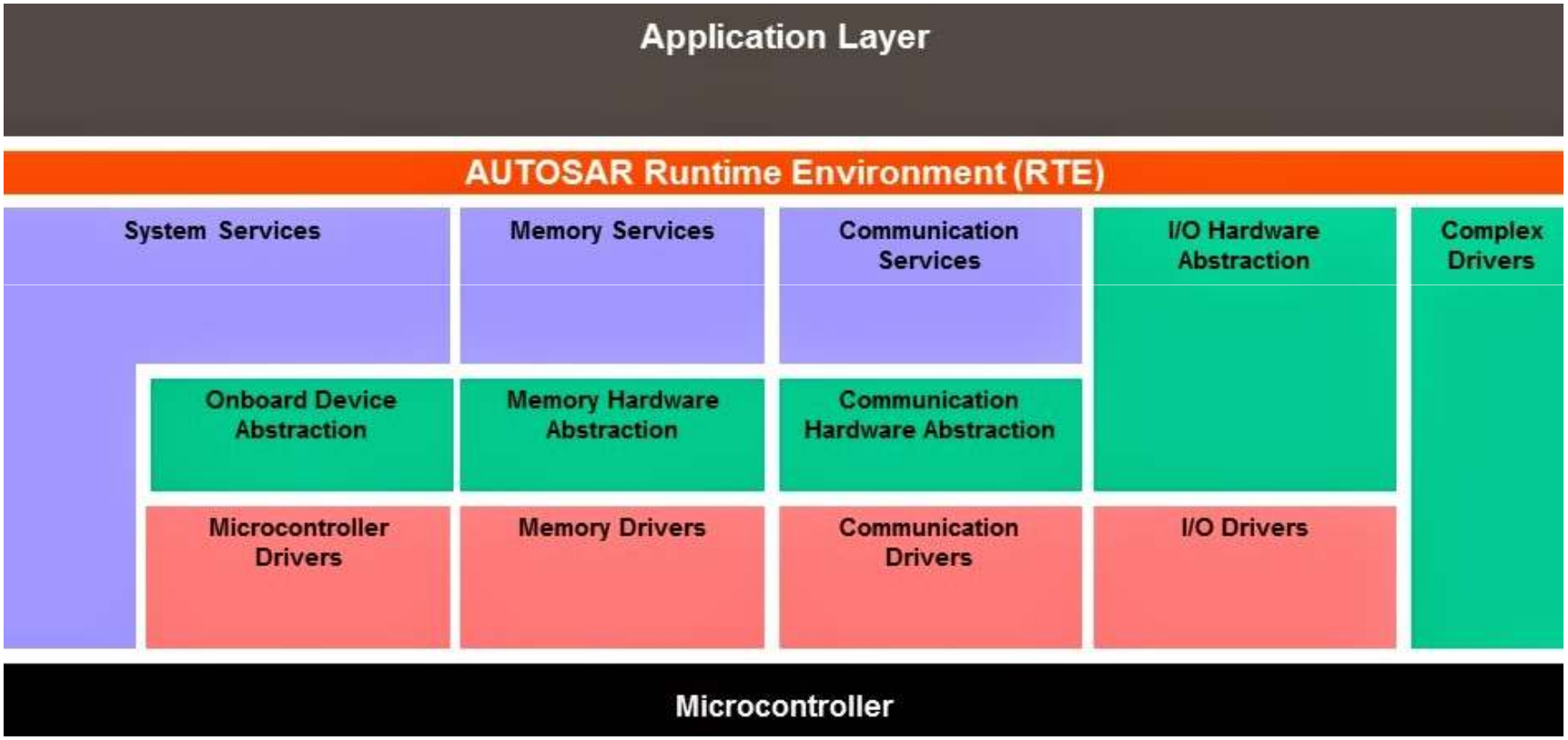
Egy általános beágyazott rendszer SW architektúrája a 90-es évek végén



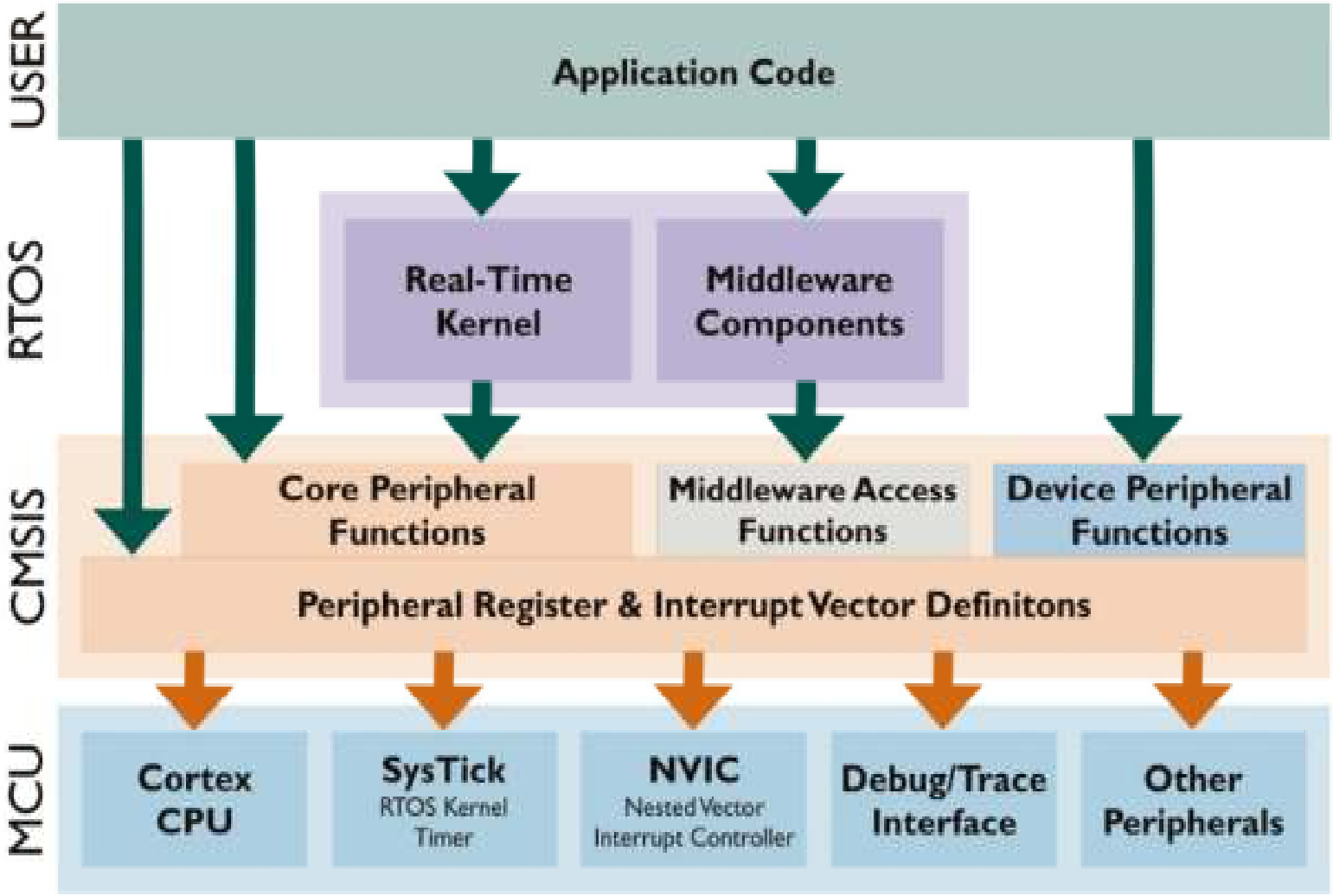
Felhasználói project

Előre fordított eCos distribúció

Hardware architektúra 2005-körül
AUTOSAR



CMSIS szerkezete (v1.3)



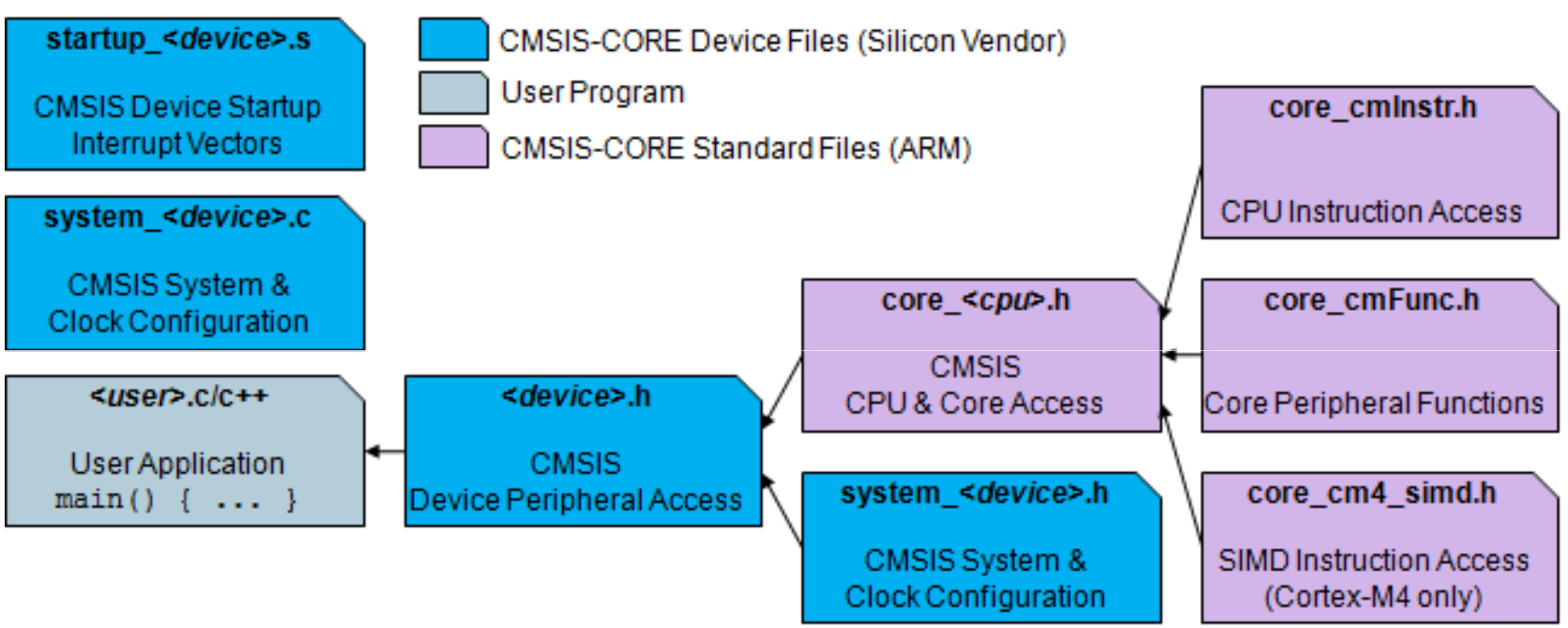
CMSIS Core

- **Hardware Abstraction Layer (HAL):** Az összes Cortex M variánshoz egy standardizált periféria és regiszter készlet kezelés a SysTick, NVIC, System Control Block, MPU, FPU registerszterekhez
- **Rendszer kivétel nevek:** A rendszer kivételekhez, interruptokhoz való hozzáférés deffiniálása
- **Header file szervezés definiciók:** Elnevezési konvenciók az mikrokontroller specifikus interruptok-hoz és header file-okhoz
- **Rendszer indítás:** Mikrovezérlő független inicializáló függvény interfész. Standardized SystemInit() függvény
- **Speciális utasítások támogatása**
- Globális változó a **system clock** frekvencia meghatározására

A CMSIS file-jai és könyvtárstruktúrája

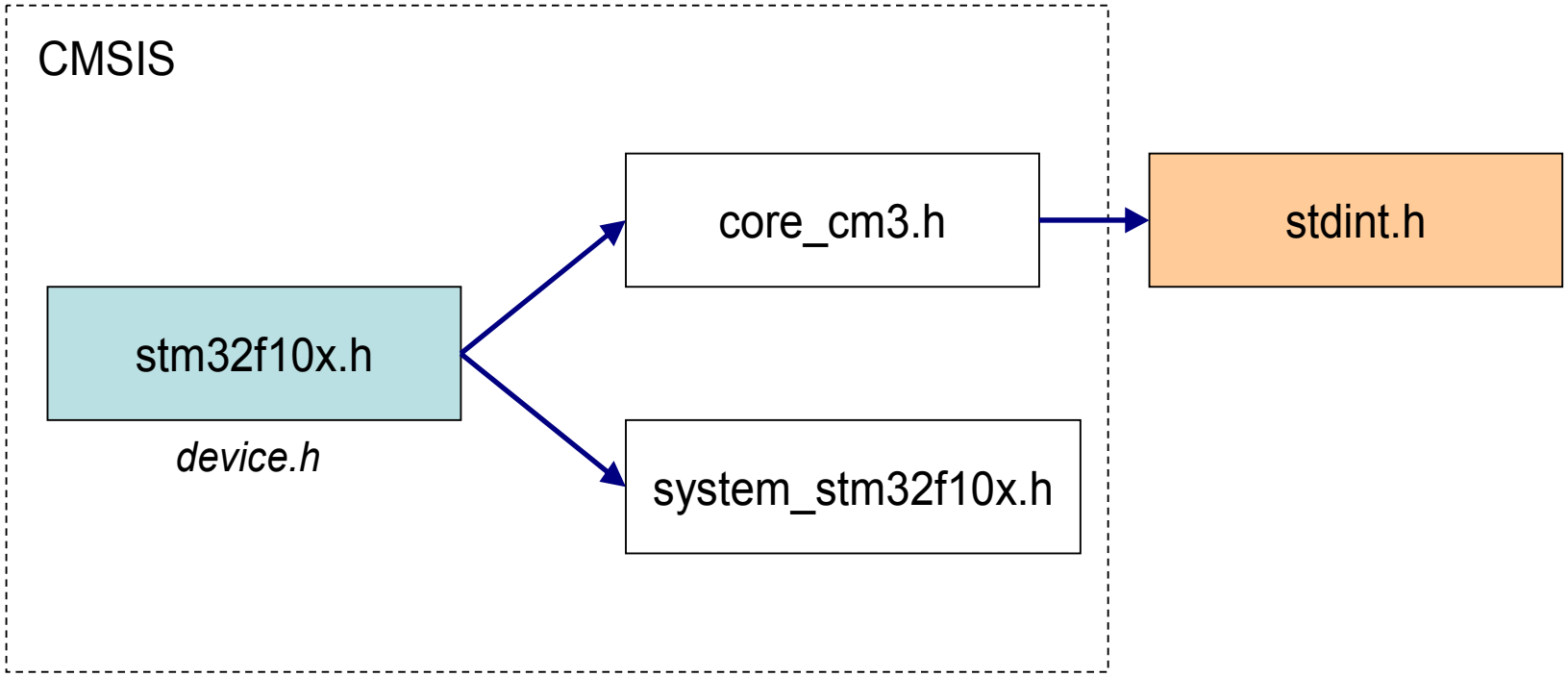
File	Szolgáltató	Leírás
device.h	Mikrovezérlő gyártó	A mikrovezérlő perifériáinak definítciója. Ez a file include-olhatja az összes többi mikrovezérlőhöz tartozó firmware headert.
core_cm0.h	ARM (fordítófüggő részekkel)	A Cortex-M0 alapperiférániának és regisztereinek definítciója
core_cm3.h	ARM (fordítófüggő részekkel)	A Cortex-M3 alapperiférániának és regisztereinek definítciója
core_cm0.c	ARM (fordítófüggő részekkel)	A Cortex-M0 alapperiférániának és regisztereinek kezelőfüggvényei
core_cm3.c	ARM (fordítófüggő részekkel)	A Cortex-M3 alapperiférániának és regisztereinek kezelőfüggvényei
startup_*device*	ARM, de a mikrovezérlő, vagy compiler gyártó adoptálhatja és testreszabhatja	A Cortex-M mikrovezérlő startupkódja és a teljes mikrovezérlő függő ugrótábla
system_*device*	ARM, de a mikrovezérlő gyártó, adoptálhatja és testreszabhatja	Mikrovezérlő függő inicializációs rész. Tipikusan a PLL és egyébb órajelforrások inicializálása történik itt.

CMSIS core struktúra



A device.h szerepe

- Az egyetlen szükséges include file a felhasználó számára (az induláshoz)



A *startup_device* file

- Fordító függő
- Startup Code, ugrótábla
- a GCC megvalósításnál a ugrótáblákhoz úgynevezett ***weak*** pragmákat használnak

```
DCD       USART1_IRQHandler,              /* USART1 */
```

Majd az így definiált nevet egy ***weak*** pragmával ráhuzalozzuk egy default handlerre:

```
#pragma weak USART1_IRQHandler = Default_Handler
```

A system_device.c

- Minden Cortex M3,M0,M7 vezérlő azonos módon elindítható legyen
- Minimum szolgáltatások

Függvény definició	Leírás
void SystemInit (void)	A mirovezérlő elindulásához szükséges rutinok végrehajtása. Tipikusan ez a függvény konfigurálja a PLL-t. Ez a függvény tipikusan a startup_device-ból híódik meg, de lehet, hogy a felhasználónak kell meghívnia.
void SystemCoreClockUpdate (void)	Beállítja a PLL-t a SystemCoreClock változó értékének megfelelően.

Változó definítció	Leírás
uint32_t SystemCoreClock	A rendszer órajel frekvenciáját tartalmazza.

A CMSIS coding guildlines-ai

- A CMSIS szabvány a MISRA 2004-es guildline-jainak betartásával készül.
- A CMSIS az adattípusokként az **<stdint.h>** -ba definiált típusokat használja
- A kifejezéseket tartalmazó #define-okat zárójelezni kell.
- A *Core Peripheral Access Layer (**CPAL**)* összes függvénye re-entráns kell, hogy legyen.
- A *Core Peripheral Access Layer (**CPAL**)* nem tartalmaz blokkoló kódot.
- Az összes interupt kezelő függvények a **_IRQHandler** utótaggal kell záródnia.

A CMSIS coding guildlines-ai 2.

- Nagybetűvel jelöl minden core, vagy periféria regisztert, illetve assemly utasítást.
- Az ún. ***CamelCase*** jelölést használja a perifériakezelő függvények és interupt függvények számára.
- PERIFÉRIA_ perfixet kell használni a perifériához tartozó függvényekhez (tehát a függvény neve elött nagybetűvel fel kell tüntetni a periféria nevét).
- Doxygen kommentekkel kell minden függvényt ellátni
  - A minimális komment a következő
    - Egy soros **brief** leírás
    - Kifejtett paraméter komment a **param**-al
    - Kifejtett visszatérési érték komment a **ret** paraméterrel.
    - Kifejtett leírás a függvény működéséről.

Gyártói megkötések

- Erős megkötések a gyártók számára, hogy hogyan kell megadniuk a periféria leírásokat
  - Irható, olvasható perifériák jelölése
  - Perifériák Struktúraként való kezelése
    - Union használata sok esetben
  - A tényleges kezelés úgy történik, hogy rámutatunk egy mutatóval a periféria struktúrára
  - Interrupt tábla elnevezéseinek és formájának megkötése

Regiszterek elérése

- A CMSIS és az LPC17xx.h használata

```
/* GPIOs
#define LPC_GPIO0_BASE        (LPC_GPIO_BASE + 0x00000)
#define LPC_GPIO1_BASE        (LPC_GPIO_BASE + 0x00020)
#define LPC_GPIO2_BASE        (LPC_GPIO_BASE + 0x00040)
#define LPC_GPIO3_BASE        (LPC_GPIO_BASE + 0x00060)
#define LPC_GPIO4_BASE        (LPC_GPIO_BASE + 0x00080)
```

```
#define LPC_GPIO0             ((LPC_GPIO_TypeDef      *) LPC_GPIO0_BASE    )
#define LPC_GPIO1             ((LPC_GPIO_TypeDef      *) LPC_GPIO1_BASE    )
#define LPC_GPIO2             ((LPC_GPIO_TypeDef      *) LPC_GPIO2_BASE    )
#define LPC_GPIO3             ((LPC_GPIO_TypeDef      *) LPC_GPIO3_BASE    )
#define LPC_GPIO4             ((LPC_GPIO_TypeDef      *) LPC_GPIO4_BASE    )
```

```
  union {
    __IO uint32_t FIOSET;
    struct {
      __IO uint16_t FIOSETL;
      __IO uint16_t FIOSETH;
    };
    struct {
      __IO uint8_t  FIOSET0;
      __IO uint8_t  FIOSET1;
      __IO uint8_t  FIOSET2;
      __IO uint8_t  FIOSET3;
    };
  };
  union {
    __O  uint32_t FIOCLR;
    struct {
      __O  uint16_t FIOCLRL;
      __O  uint16_t FIOCLRH;
    };
    struct {
      __O  uint8_t  FIOCLR0;
      __O  uint8_t  FIOCLR1;
      __O  uint8_t  FIOCLR2;
      __O  uint8_t  FIOCLR3;
    };
  };
} LPC_GPIO_TypeDef;
```

```
LPC_GPIO1->FIOPIN = 3;
```

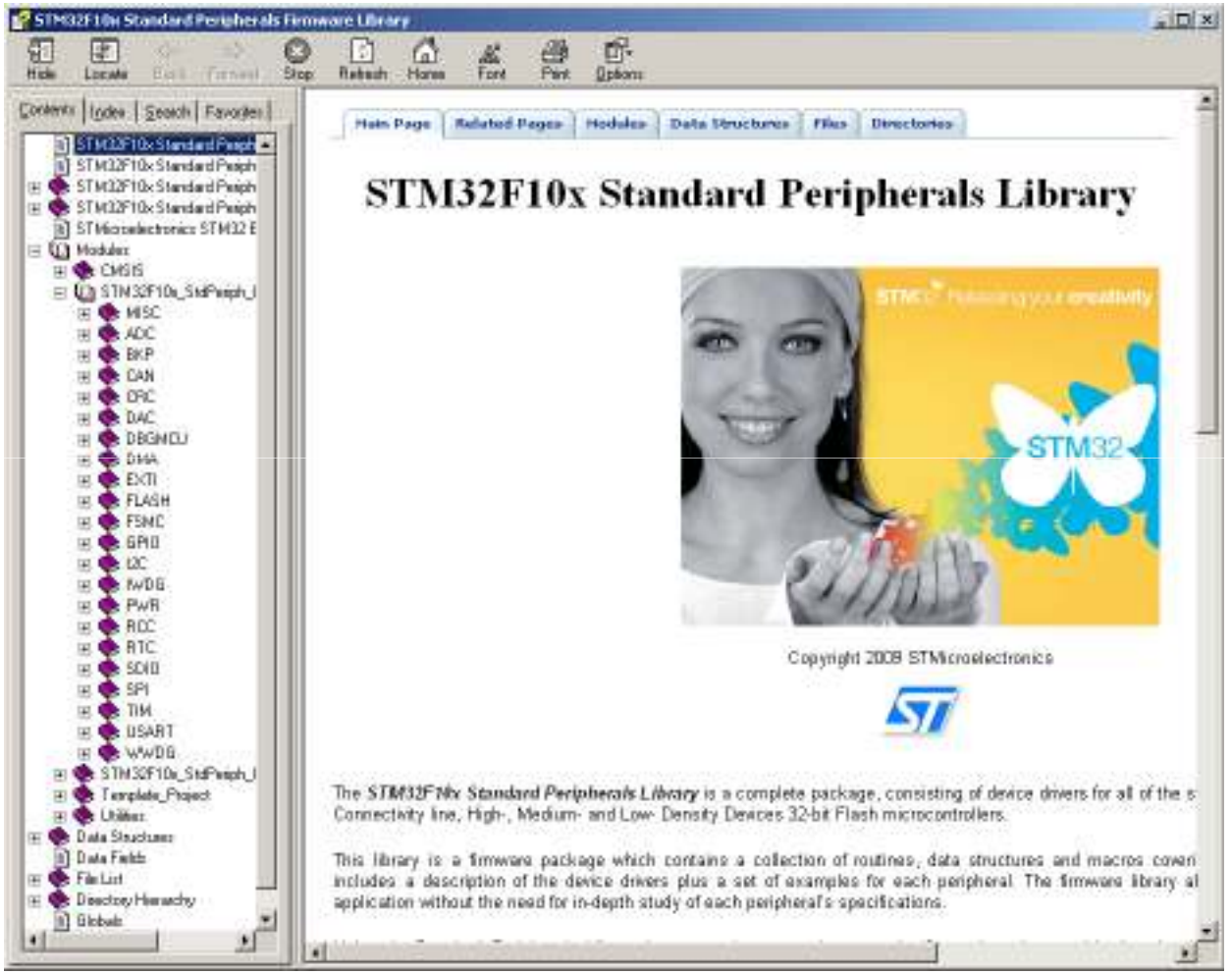
Software környezetek

- Túl komplex hardware alacsony szintről nagyon nehéz értelmes fejlesztést végezni
- Szükséges az erősebb szoftvertámogatás
- Gyártók elég nagy erőforrásokat fektetnek ebbe, de egy jó része az erőfeszítéseknek nem konvergens

Firmware Library-k
~ 2010

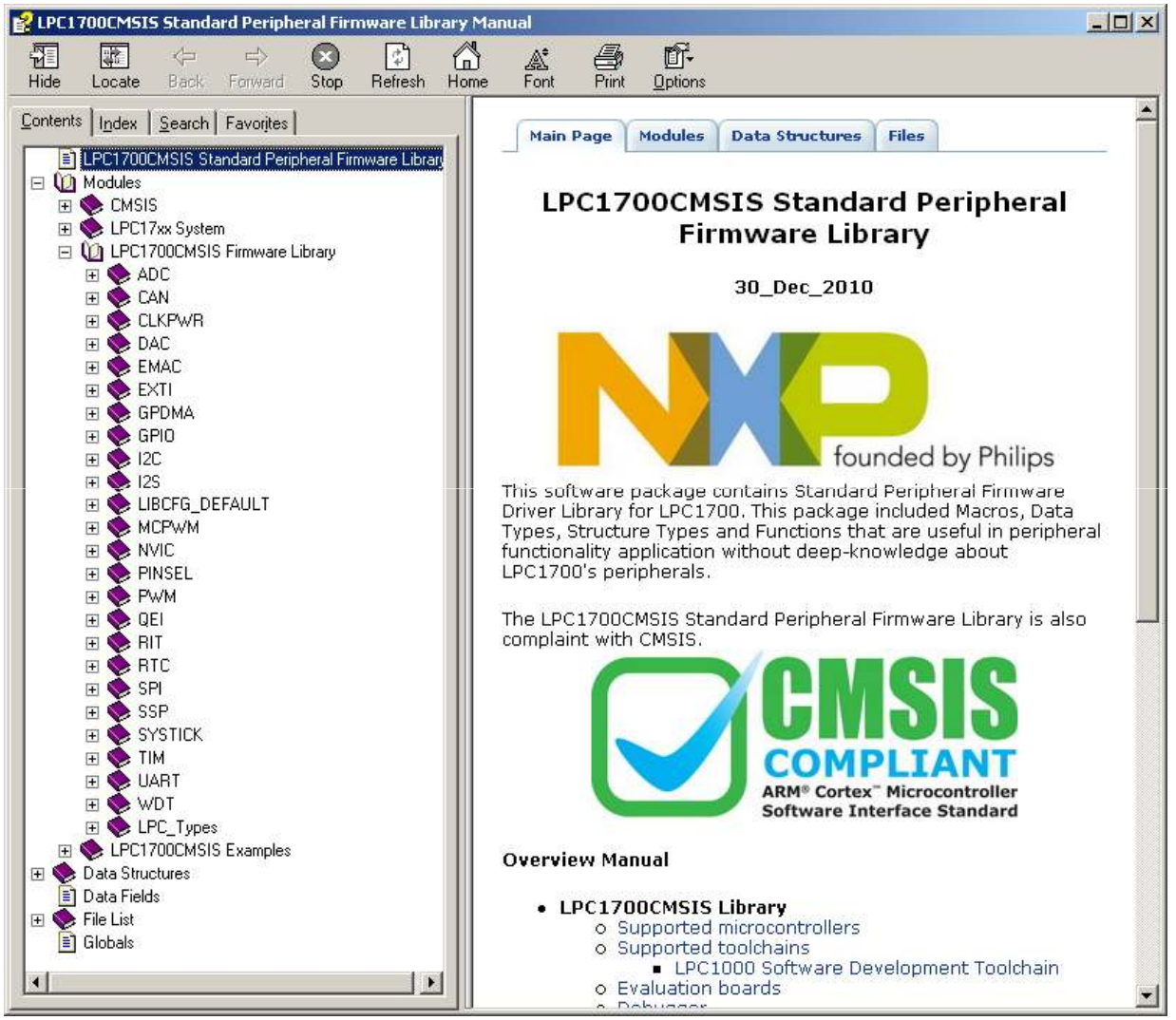
Az ST firmware library-e

- Doxygen-ben kommentezett
- Erősen struktúrákon alapuló rendszer
  - Minden perifériát lefed
  - Számtalan példával ellátva



Az NXP firmware library-e

- Doxygen-ben kommentezett
- Erősen struktúrákon alapuló rendszer
  - Minden perifériát lefed
  - Számtalan példával ellátva



Fejlesztő környezetek ~ 2010

Fejlesztőkörnyezet lehetőségek

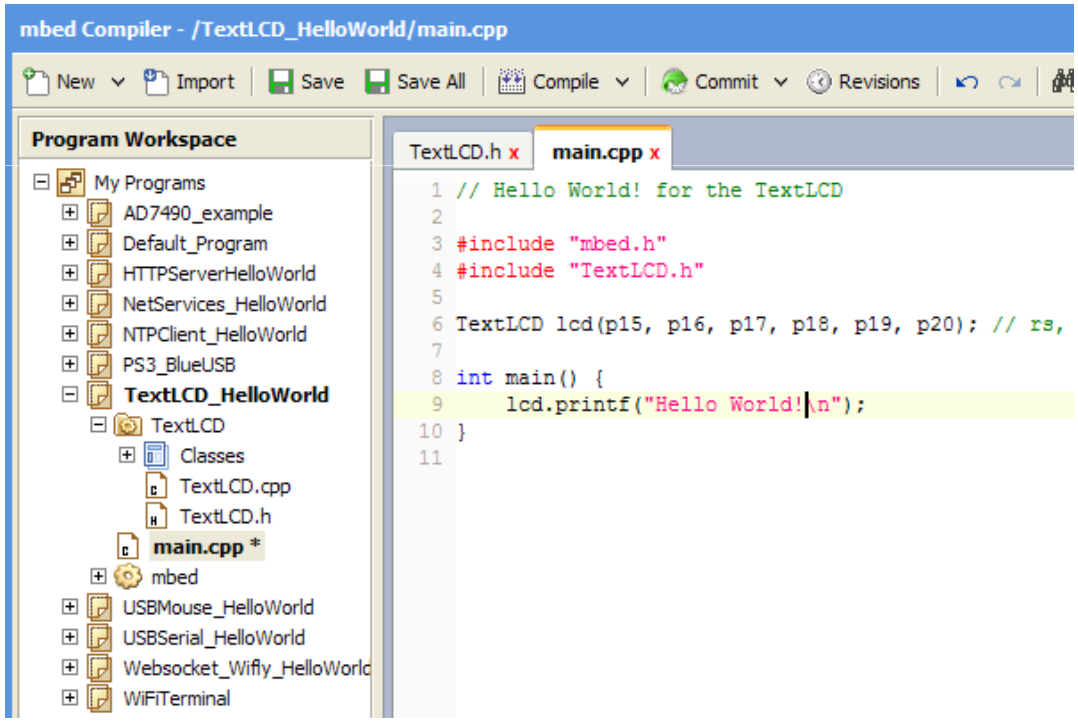
- Proffeszionális drága verziók
  - IAR Embedded Workbench
  - Keil MDK-ARM Microcontroller Development Kit
  - Code Composer Studio
- GNU alapú összeintegrált kevéssé drága verziók
  - Rowlely CrossWorks
- Cégek által támogatott
  - Red Suite by Code Red
- GNU alapú ingyenes
  - Coocox

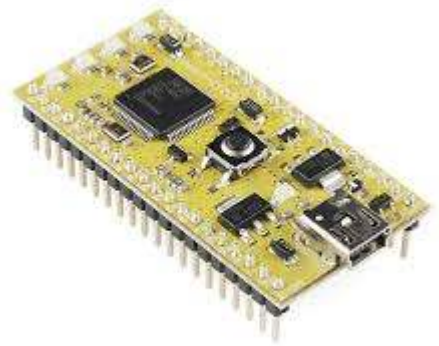
Eclipse CDT alapú környezetek

- Eclipse IDE, CDT pluginnal
- GNU GCC fordító
- GNU GDB debugger
- OpenOCD debug hardware támogatás

MBED webes fejlesztőkörnyezet

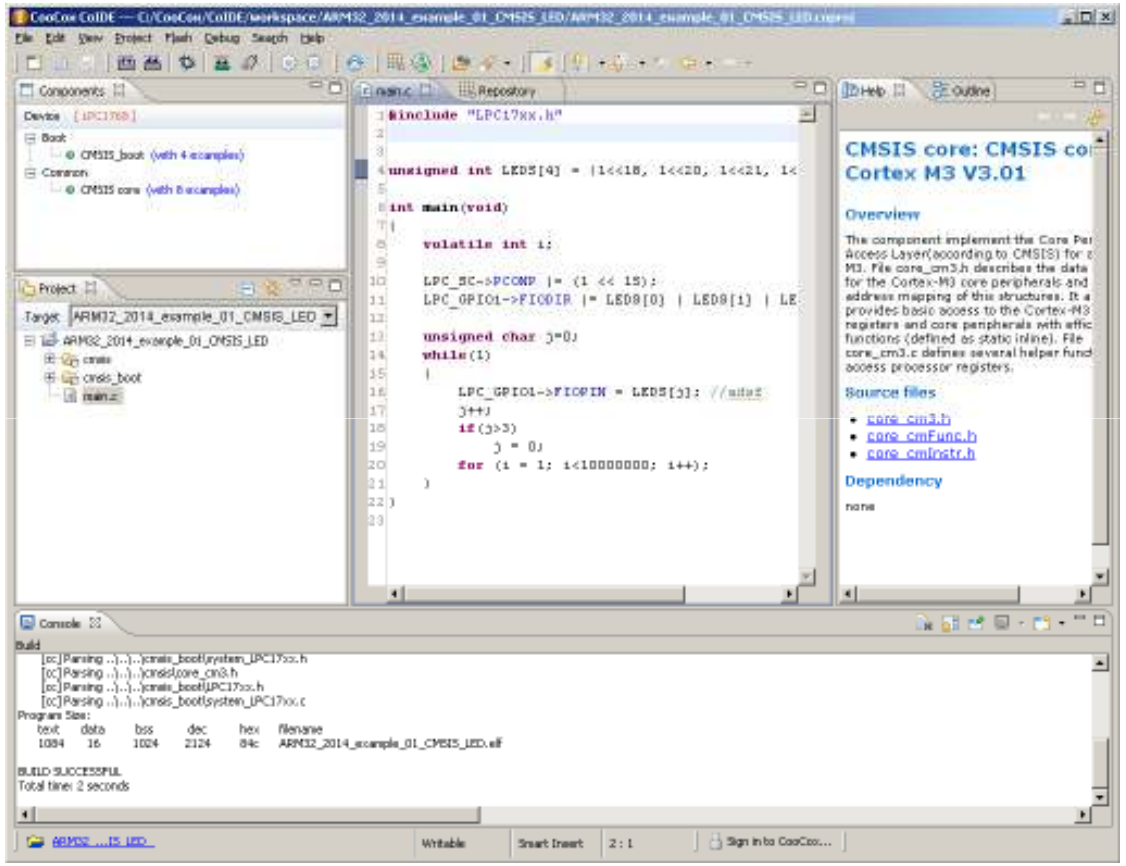
- Gyorsan magas szintű alkalmazás összerakása
- Web-es fejlesztőkörnyezet
- NXP LPC sorozatú mbed-ek





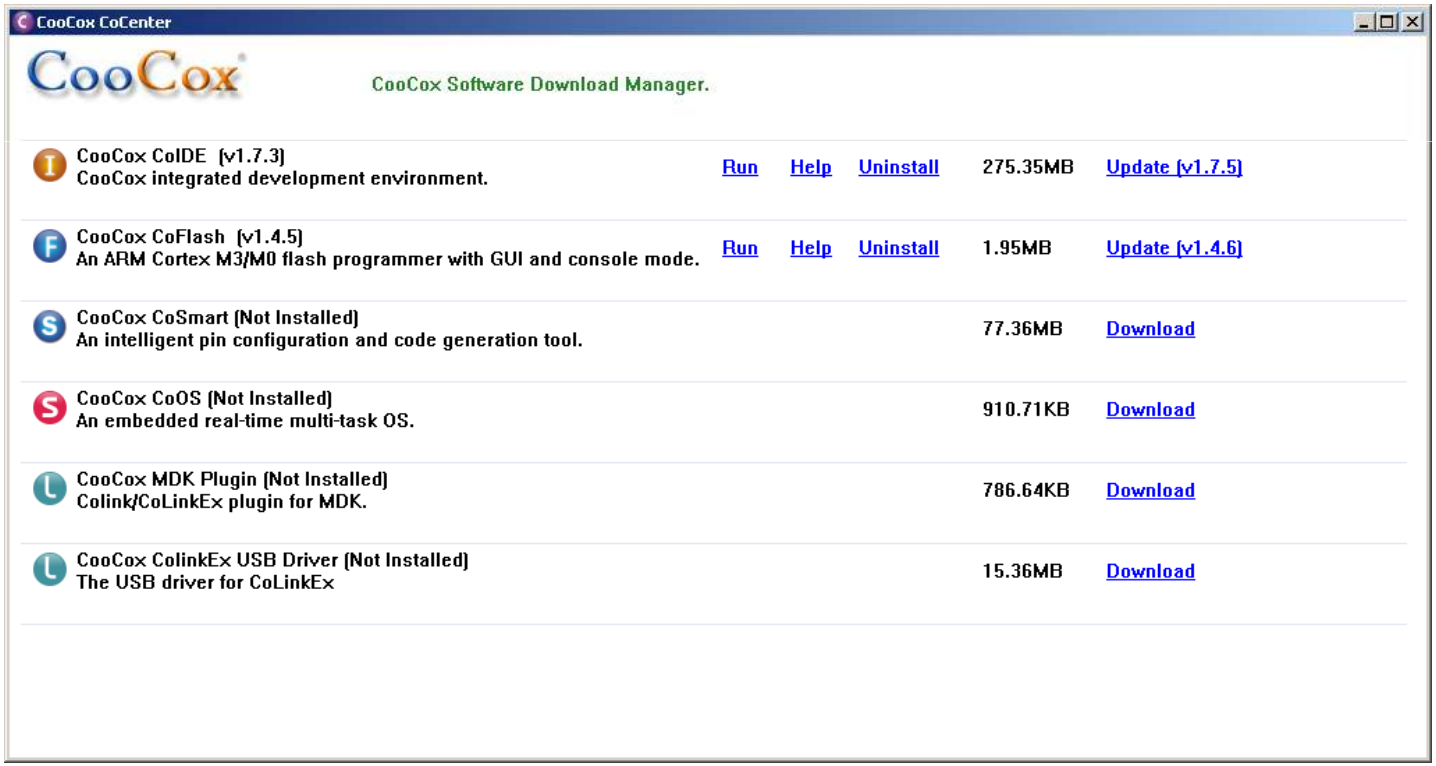
Coocox Eclipse alapú fejlesztőkörnyezet

- Kínaiak által összerakott Eclipse alapú fejlesztőkörnyezet
- Nagyon jó komponens alapúság
  - Chip és Board support
- Nem követi a firmware library-k változásait

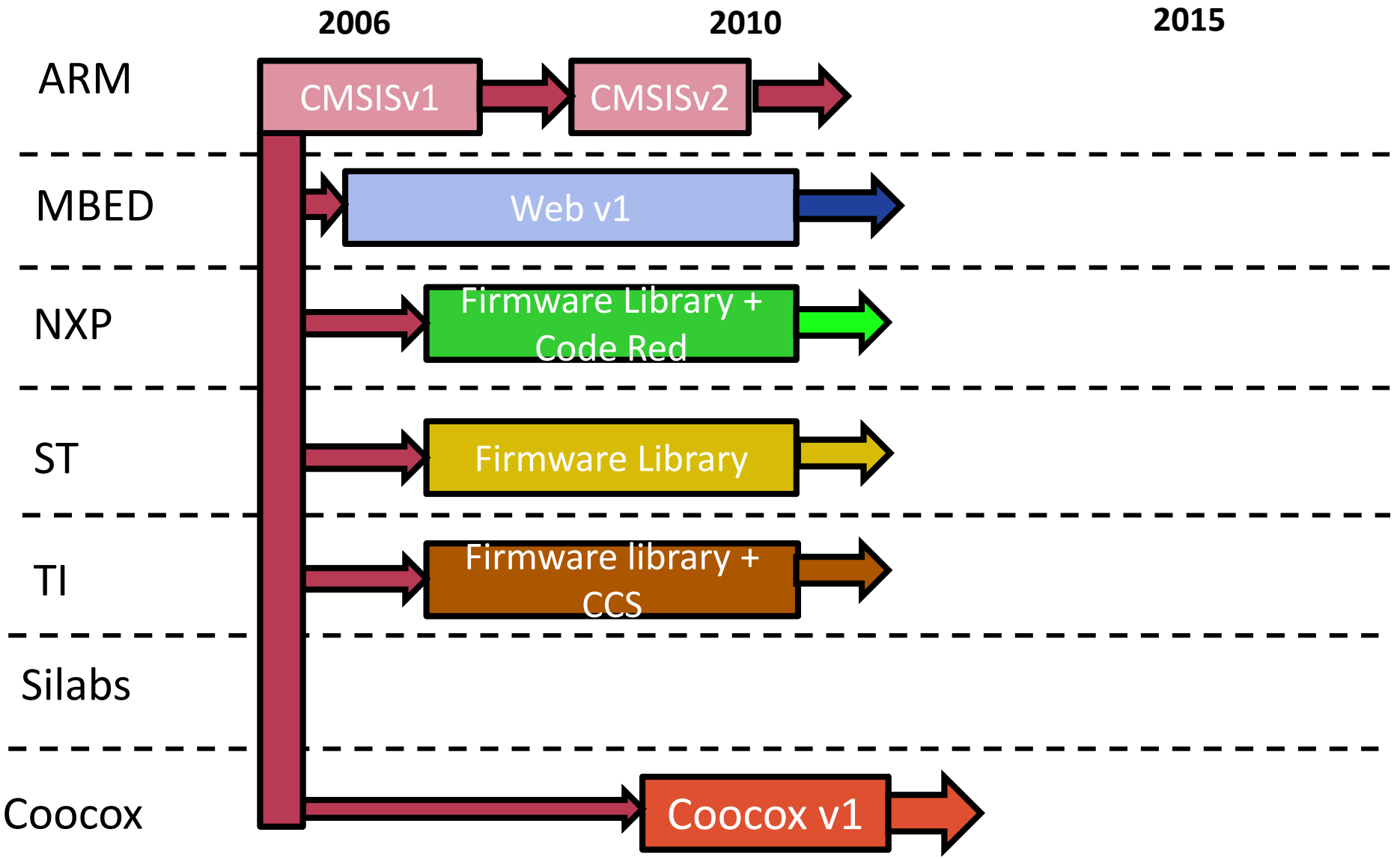


Fejlesztőkörnyezet Coocox

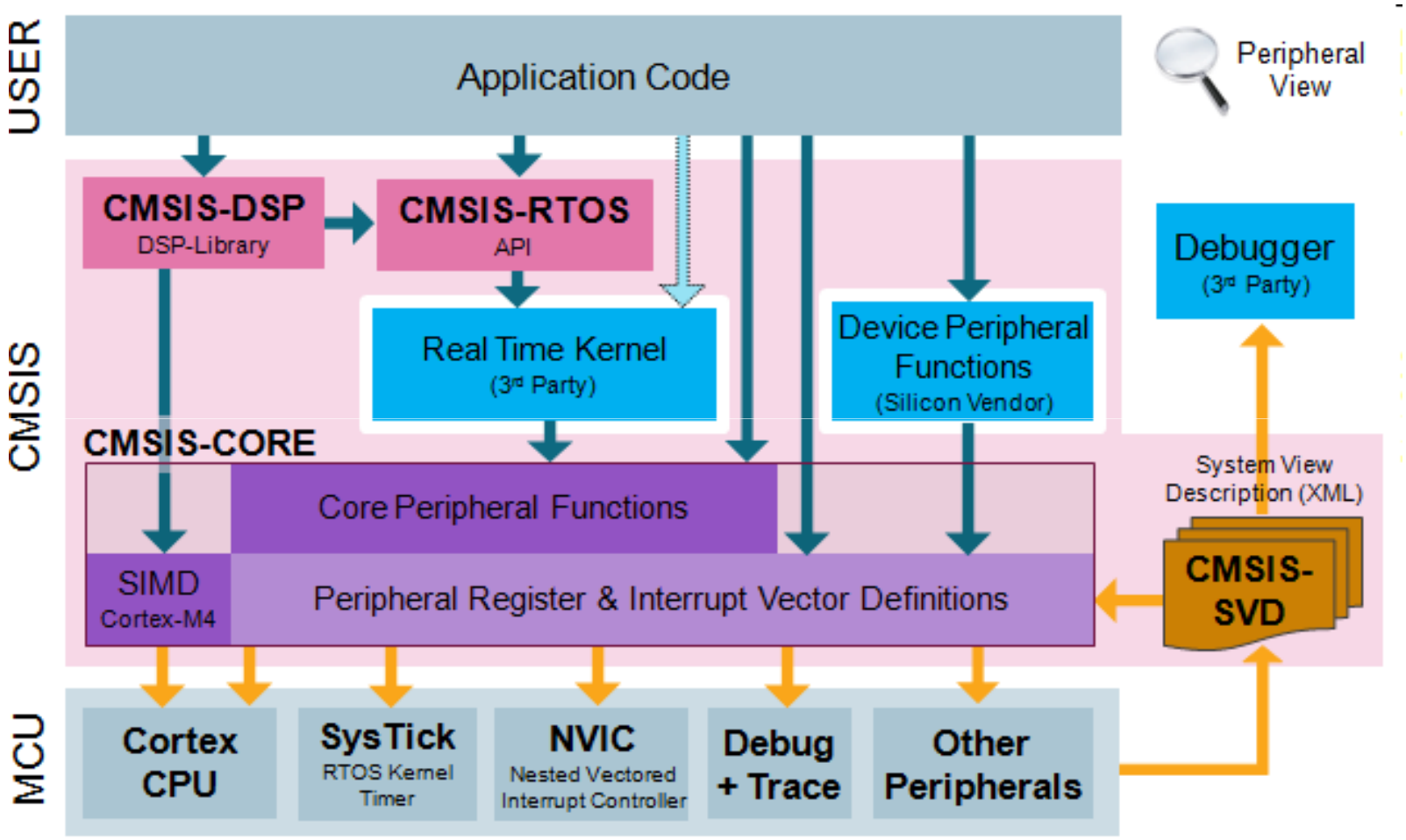
- Jól támogatott: a lényegesebb Cortex M architektúrák benne vannak
- Jó a debugger támogatása
- Egyszerűen telepíthető: CoCenter



Szoftver trendek 2010



~2011 CMSIS v3 szerkezeti változása



CMSIS DSP library

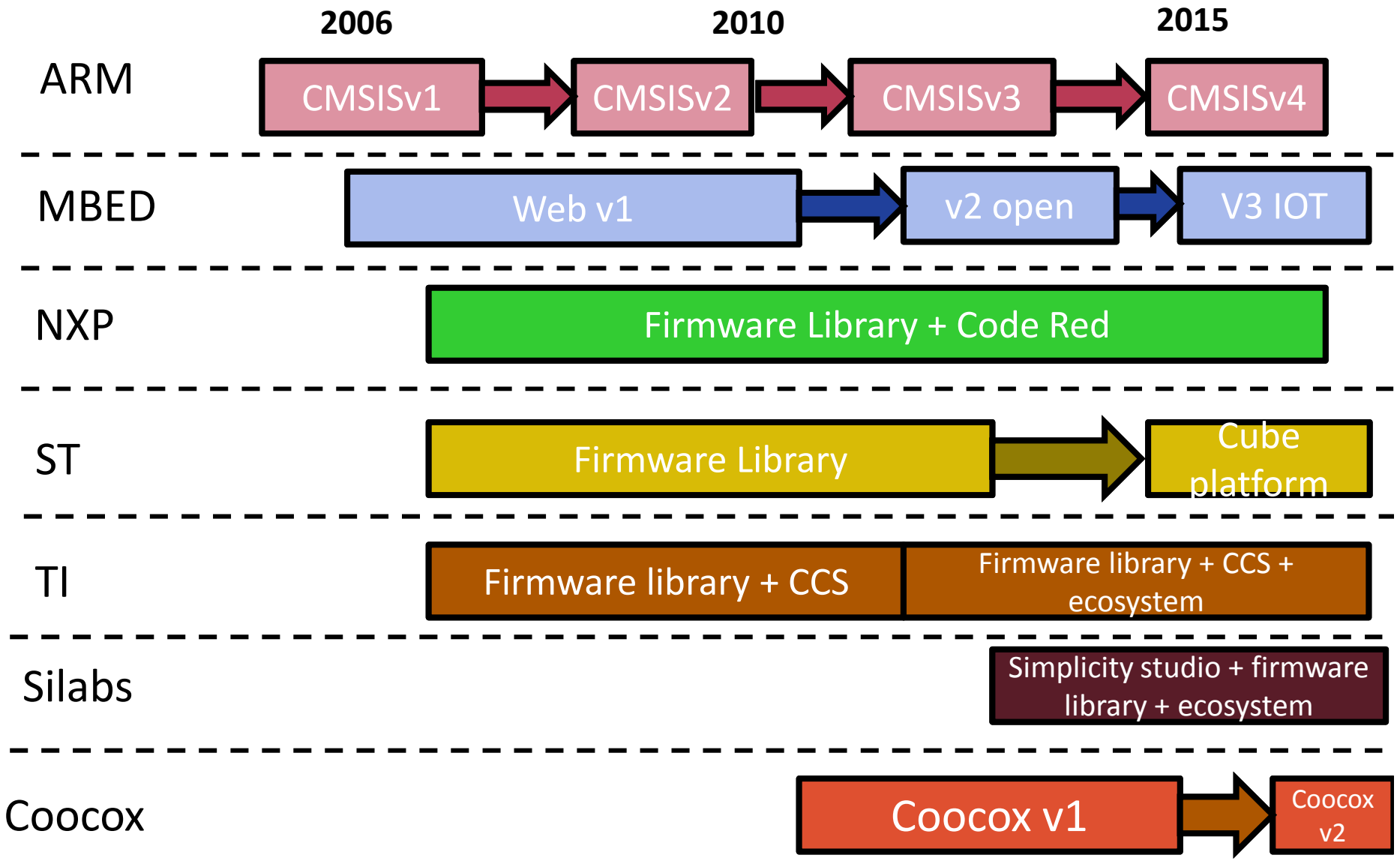
- Cortex M4-re, M3-ra optimalizált assembly szintü rutinok
- Basic math functions
  - Vector abszolút éték, összeadás, szorzás, kivonás
- Gyors komplex matematikai függvények
  - Cosinus, Sinus, Gyökvonás
  - Komplex számok matematikája
- Szűrök
  - FIR, IIR

CMSIS DSP library

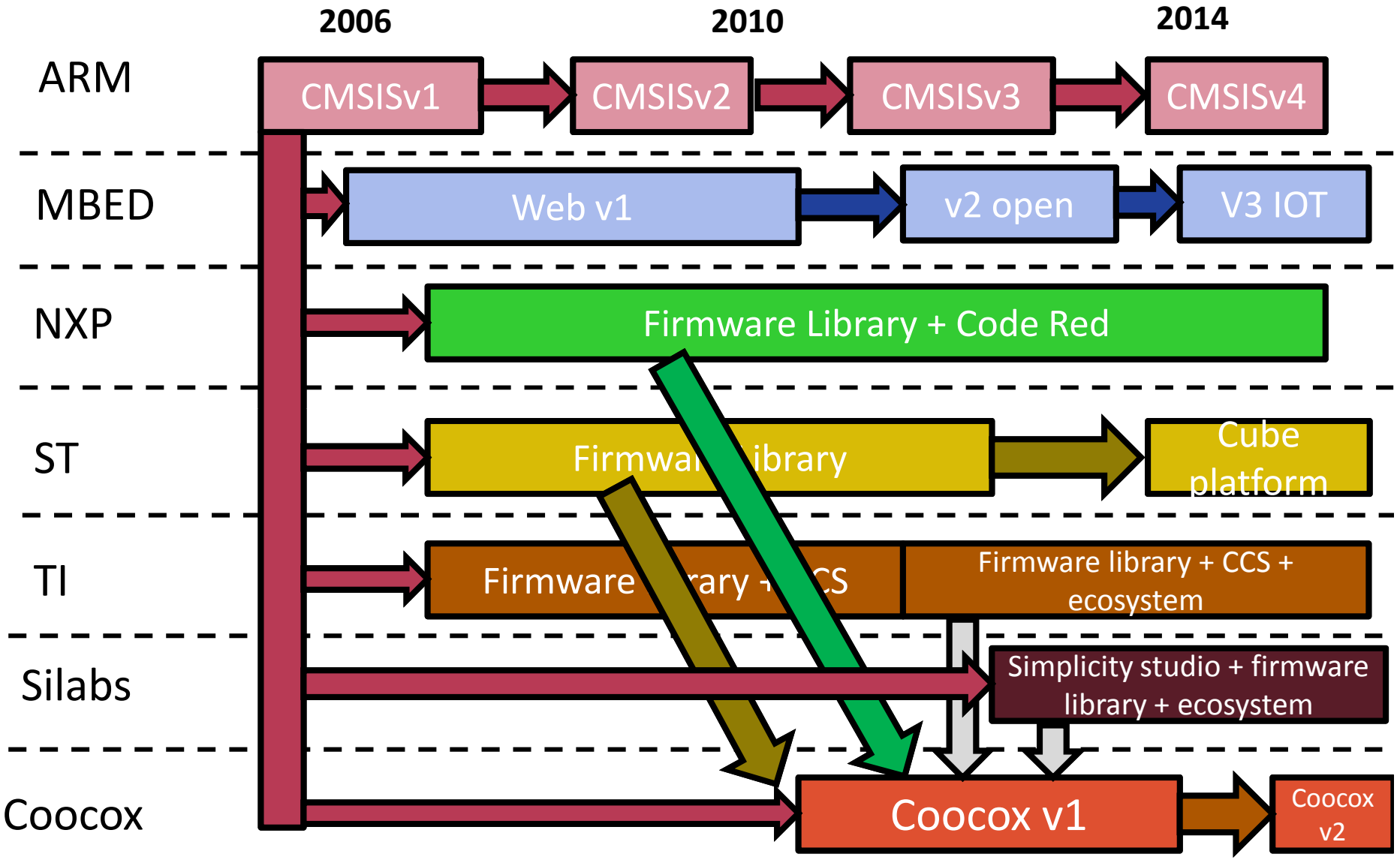
- Mátrix műveletek
  - Kivonás, összeadás, szorzás, transzponálás, invertálás
- Alkalmazás specifikus függvények
  - PID szabályozás
  - Interpolásció
  - FFT
  - Statisztikai függvények

Mostani állapot

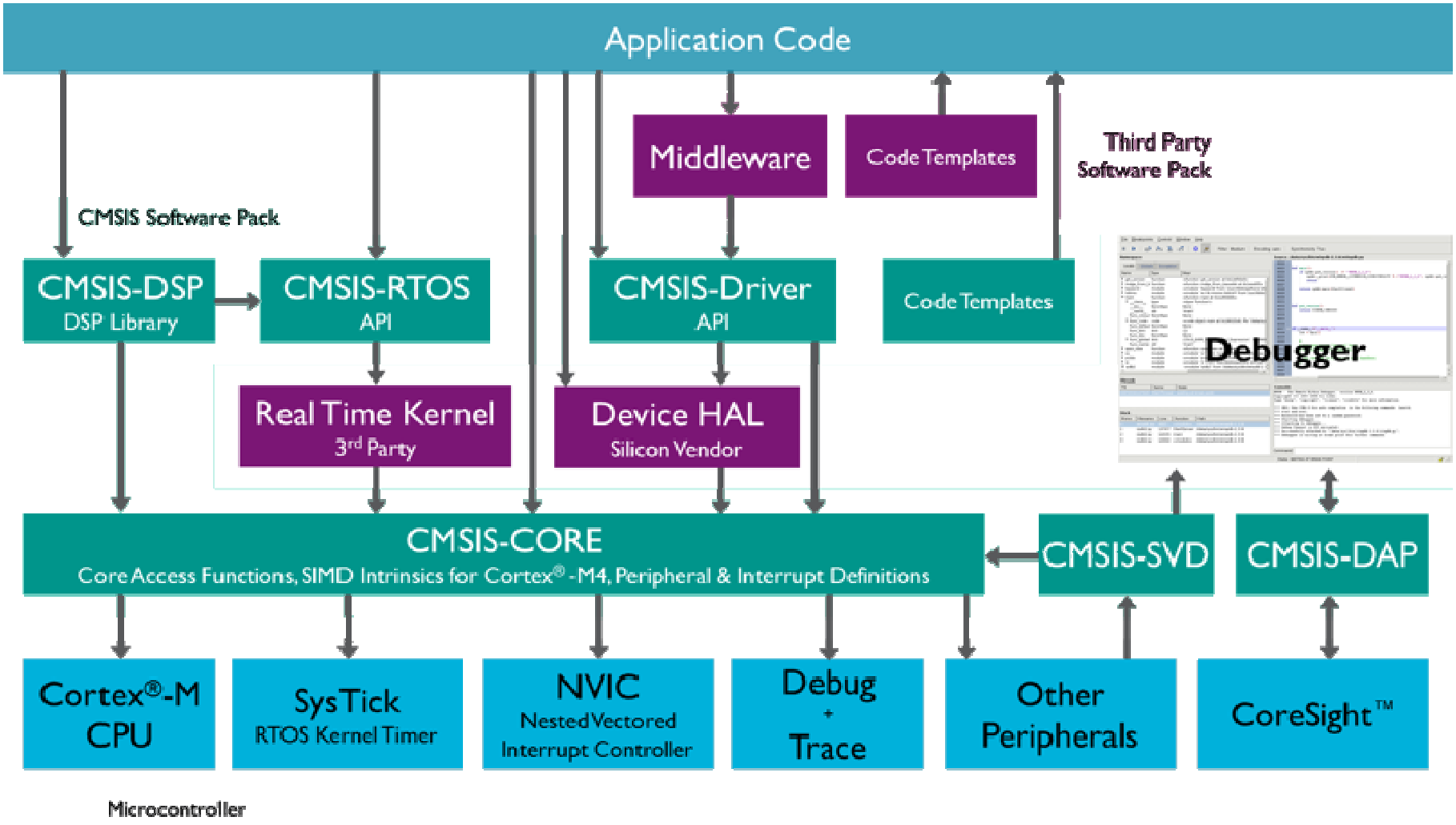
Szoftver trendek 2015



Egymásra hatások Firmware library-k

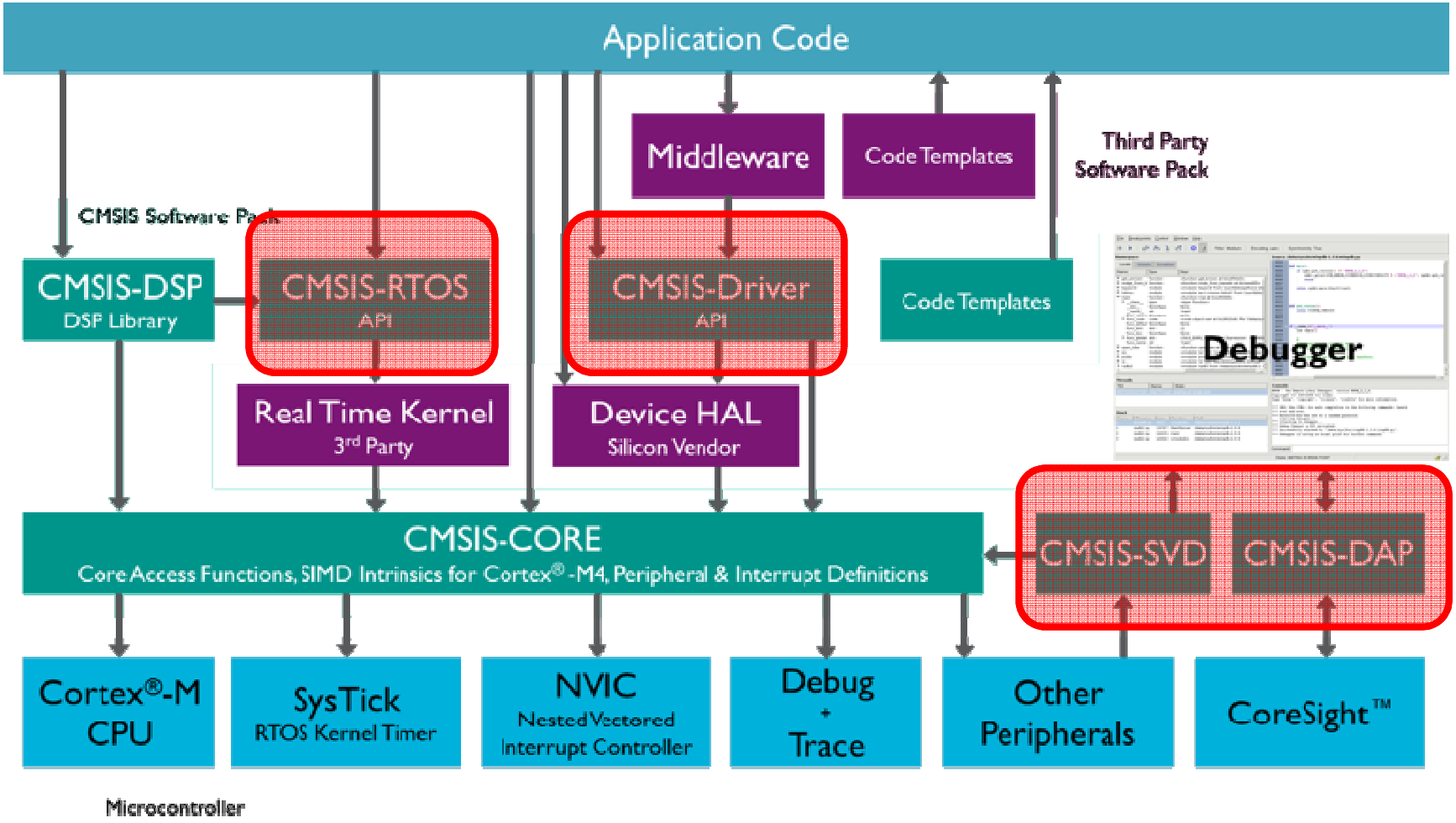


CMSIS szerkezete (v4.2)



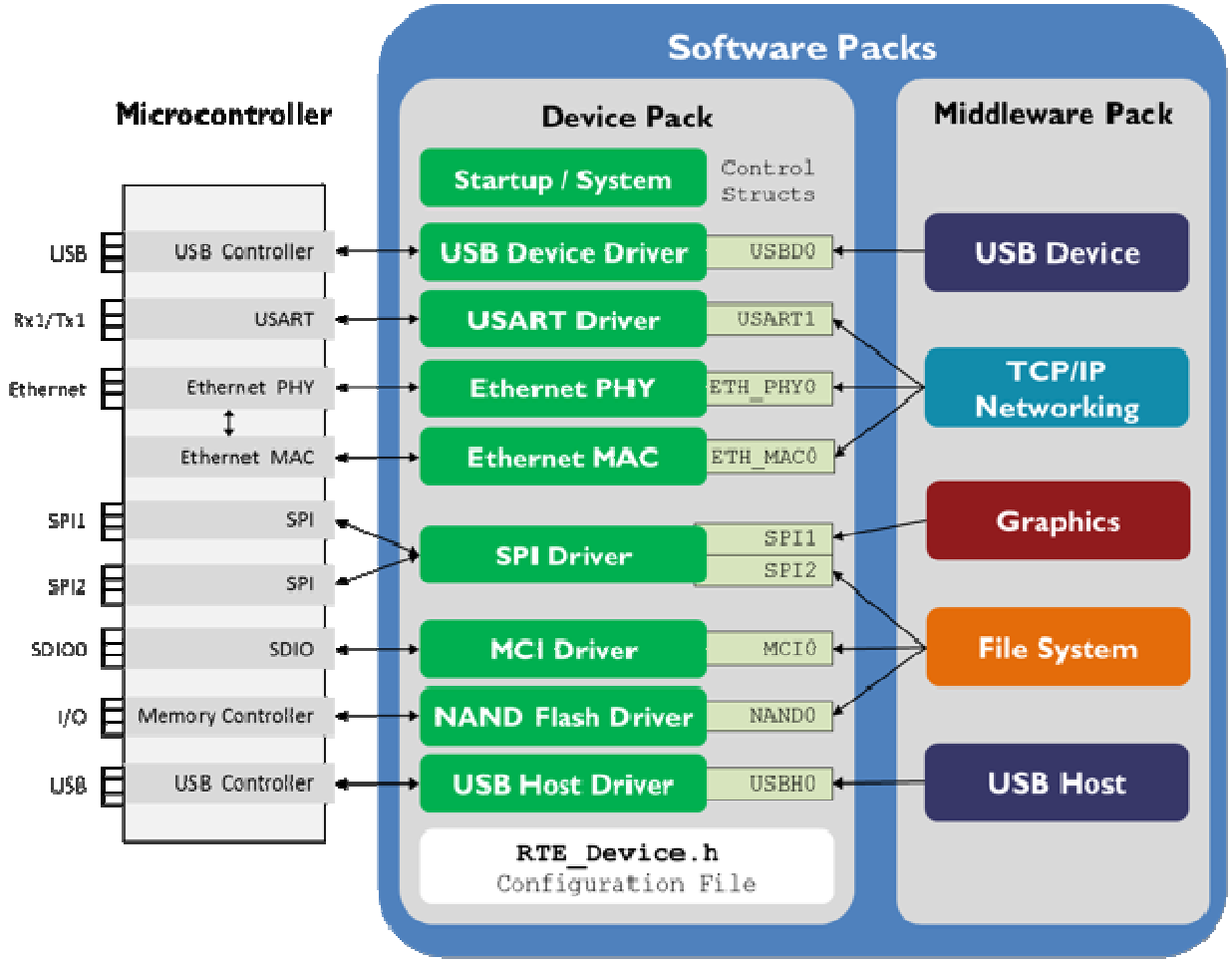
CMSIS v4

- Kérdés, hogy mikor és mennyire fogják átvenni v4 módosításait



CMSIS Driver API

- Gyártó független periféria driver library

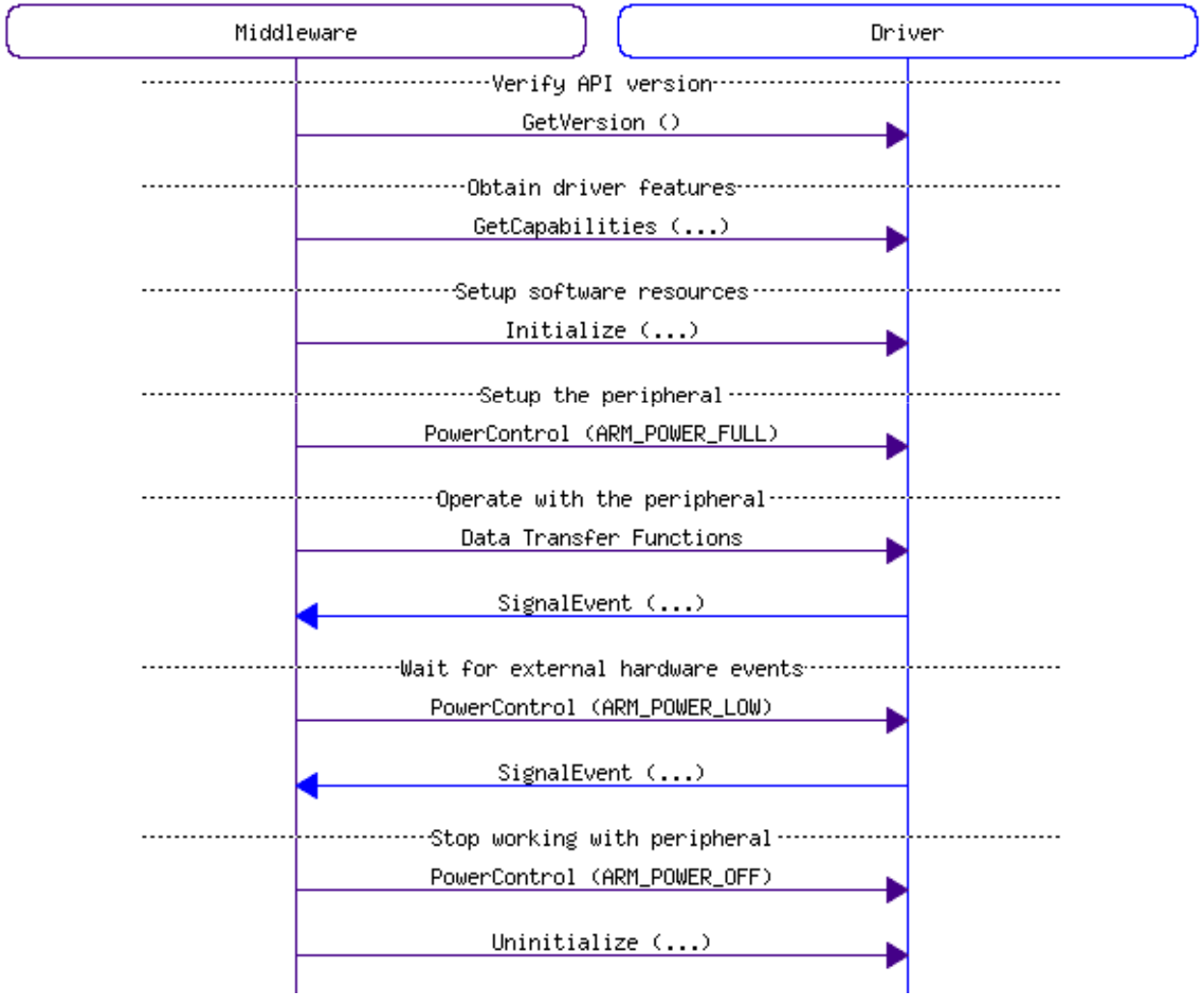


CMSIS Driver API

- Egyenlőre az LP18xx-re van próba implementáció
- A Keil kezdi még átvenni a dolgot

Header File	Description
Driver_Common.h	Common Driver Definitions
Driver_ETH.h	Ethernet Interface
Driver_ETH_MAC.h	Ethernet MAC Interface
Driver_ETH_PHY.h	Ethernet PHY Interface
Driver_Flash.h	Flash Interface
Driver_I2C.h	I2C Interface
Driver_MCI.h	MCI Interface
Driver_NAND.h	NAND Interface
Driver_SPI.h	SPI Interface
Driver_USART.h	USART Interface
Driver_USB.h	USB Interface
Driver_USBD.h	USB Device Interface
Driver_USBH.h	USB Host Interface

CMSIS Driver API

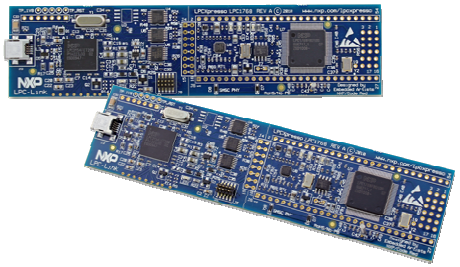


Gyártói megoldások

NXP megoldások

- Két harware vonulat:
  - Mbed

  - LPCXpresso

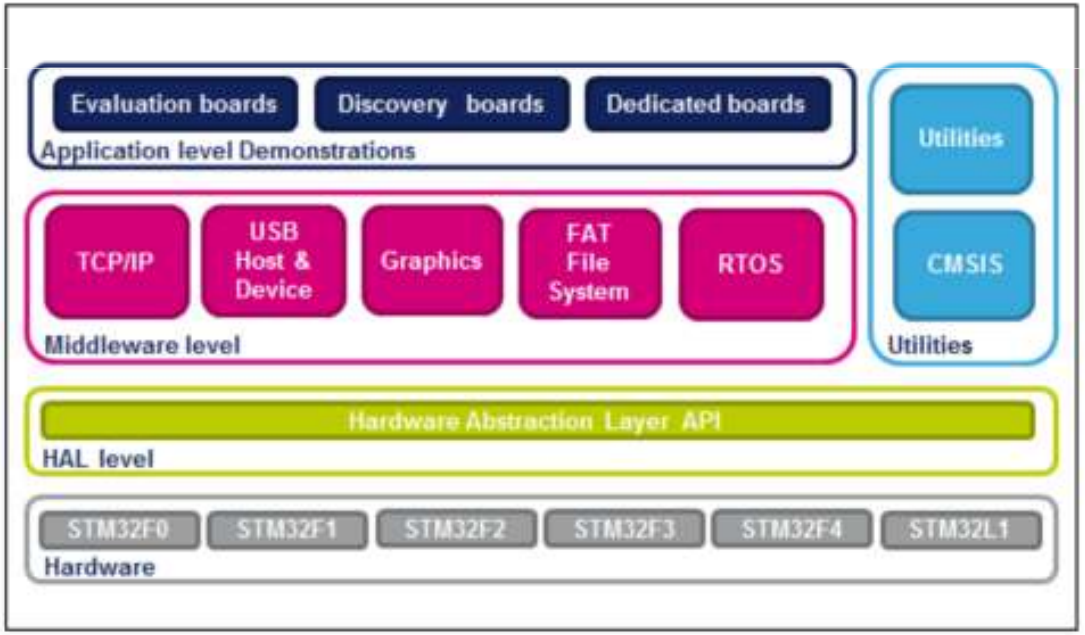


- Szoftver
  - Firmware Library
  - LPCXpresso / Code-Red ecosystem támogatás
    - Debuggolás bizonyos kódméret után fizetős

code_red

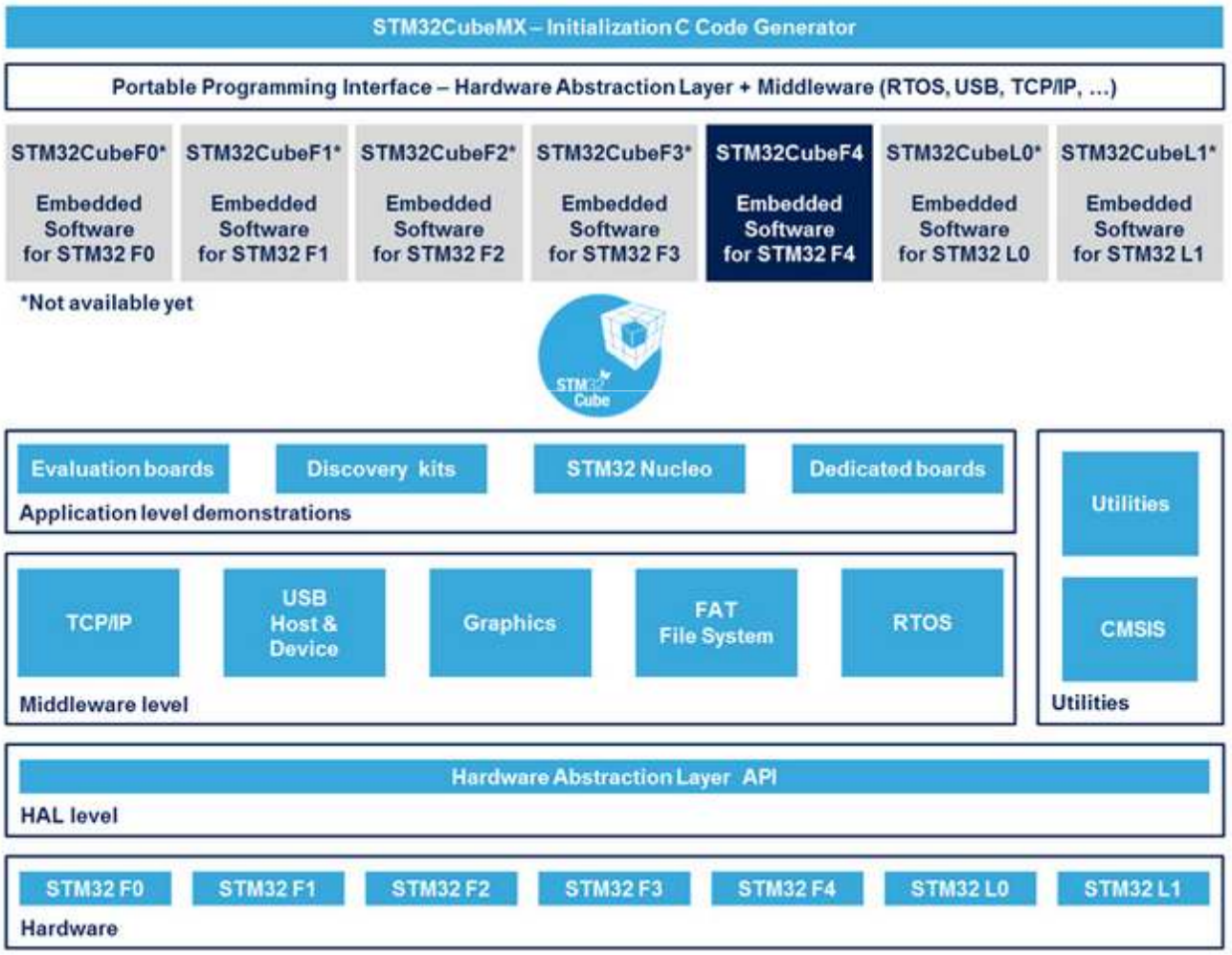
ST megoldások

- Két harware vonulat:
  - Nucleo
  - Discovery board
- Szoftver
  - Firmware Library (2014-ben leváltásra kerül)
  - Cube platform
  - Nincs saját IDE



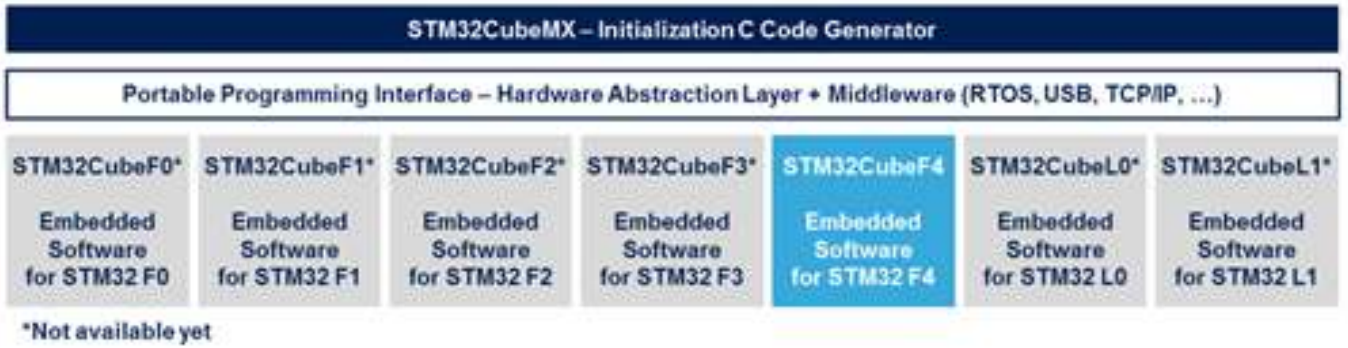
STMCube32

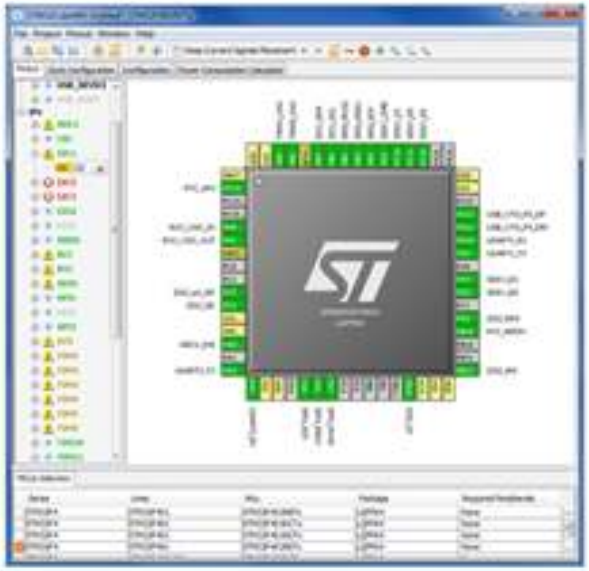
- A hagyományos Firmware library-t leváltó API készlet + eCosystem



STMCube32 Mx

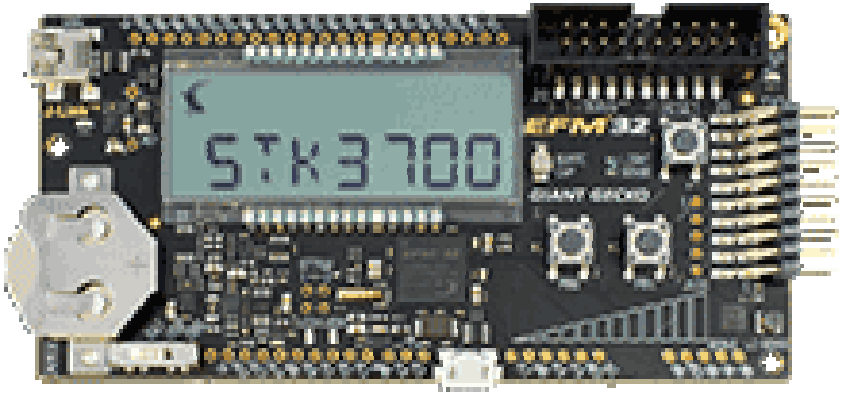
- Grafikus periféria, és órajel inicializáló rendszer





Silabs megoldások

- Komplett fejlesztőkártyák
  - 2015 rádióval integrált verziók

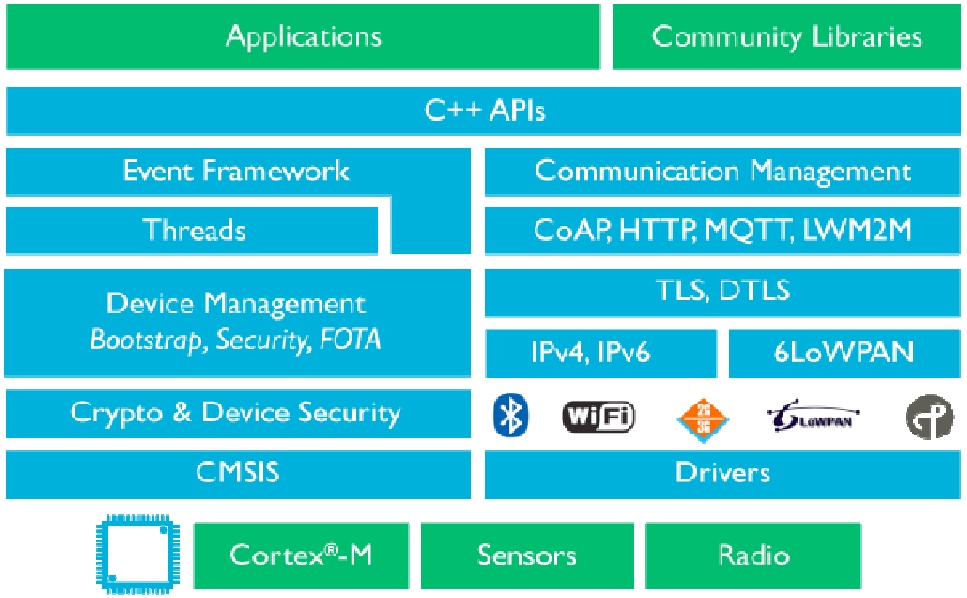


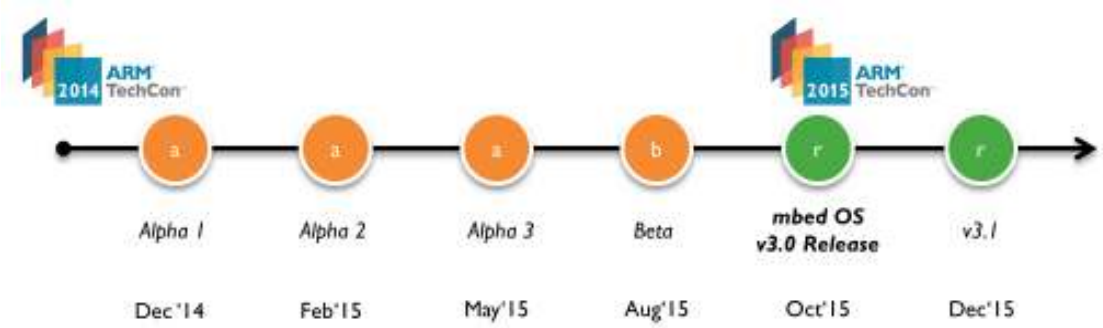
- Simplicity stúdió
  - Saját Firmware library
  - Saját ecosystem
  - Debug támogatás
  - Fogyasztás támogatás
  - Nagyon új még
  - A Coocox pár dologban jobb

Fejlesztői környezetek

MBED most

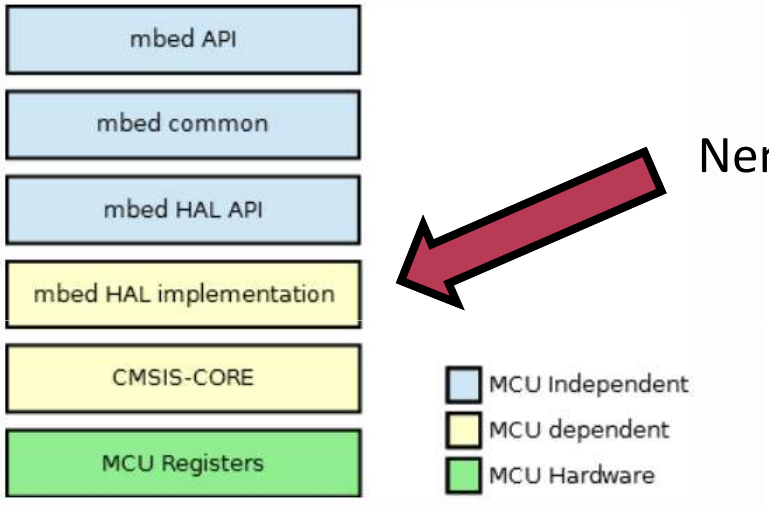
- 2013-ban kinyitják a platformot. Több mint 20 alapkártyát támogatnak
- Forráskód elérhető.
- Exportálható különböző környezetekbe
- 2015-re erősen megcélozzák az IoT világot





MBED problémák

- Alap API C++ szintaktikájú
- Jelenleg nem követi a CMSIS driver struktúrát



Nem használja a CMSIS driver réteget

- Web-ből nem debuggolható a normál „JTAG”-es módon
- Exportálásnál nem exportálja ki a forrásfile-okat
- Coocox export debuggolása még nem az igazi (1 hónapos feature)

Coocox 2.0

- Továbbfejlesztés
- Git alapú komponens adatok
  - Egészen máshogy kell majd a komponenseket kezelni
- Megváltozik a project struktúra

- De milyen API-kat fog támogatni?

Szoftver trendek 2015, mi a várható jövő?