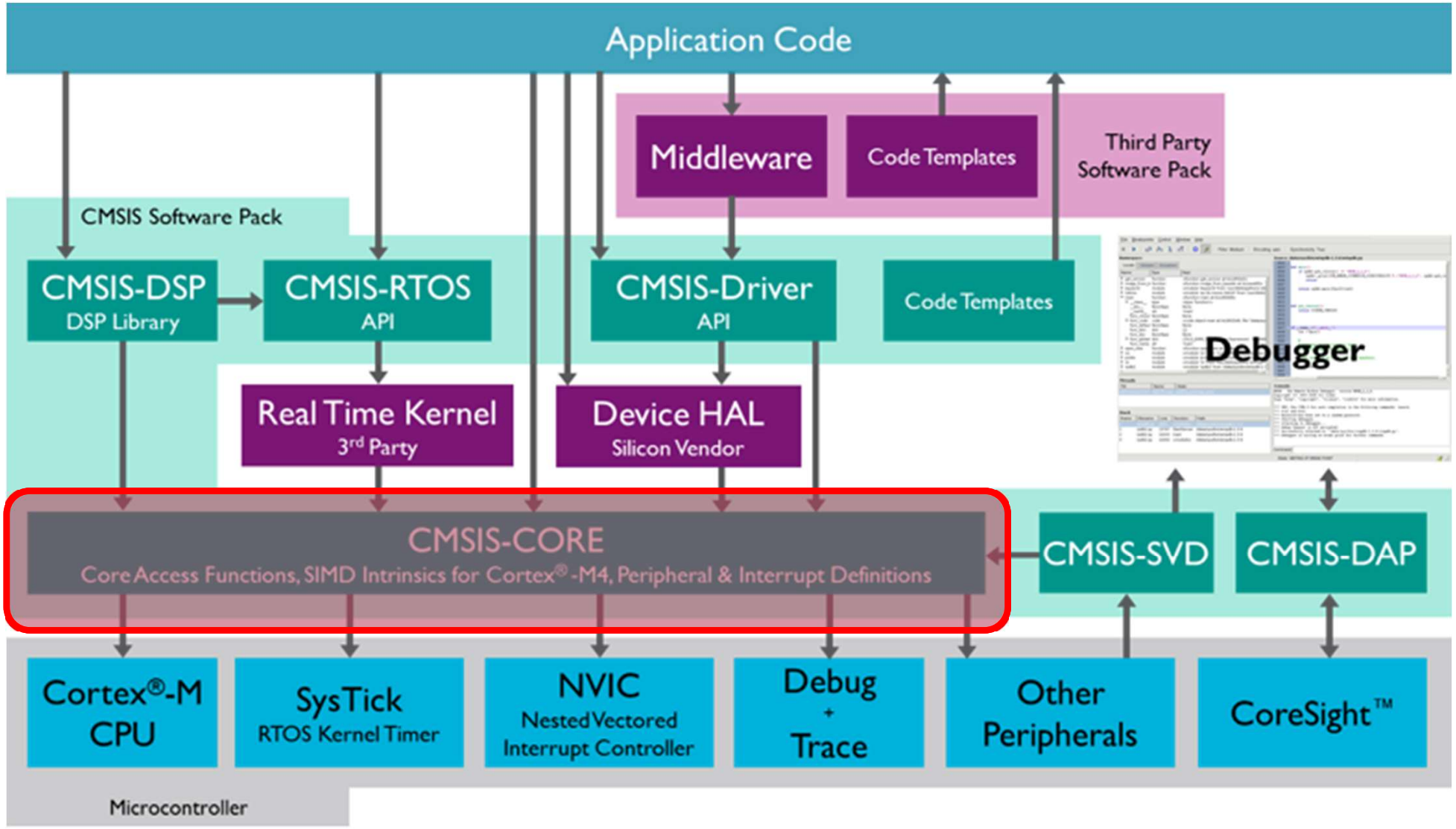
ARM Cortex magú mikrovezérlők

1. Labor: CMSIS

Scherer Balázs

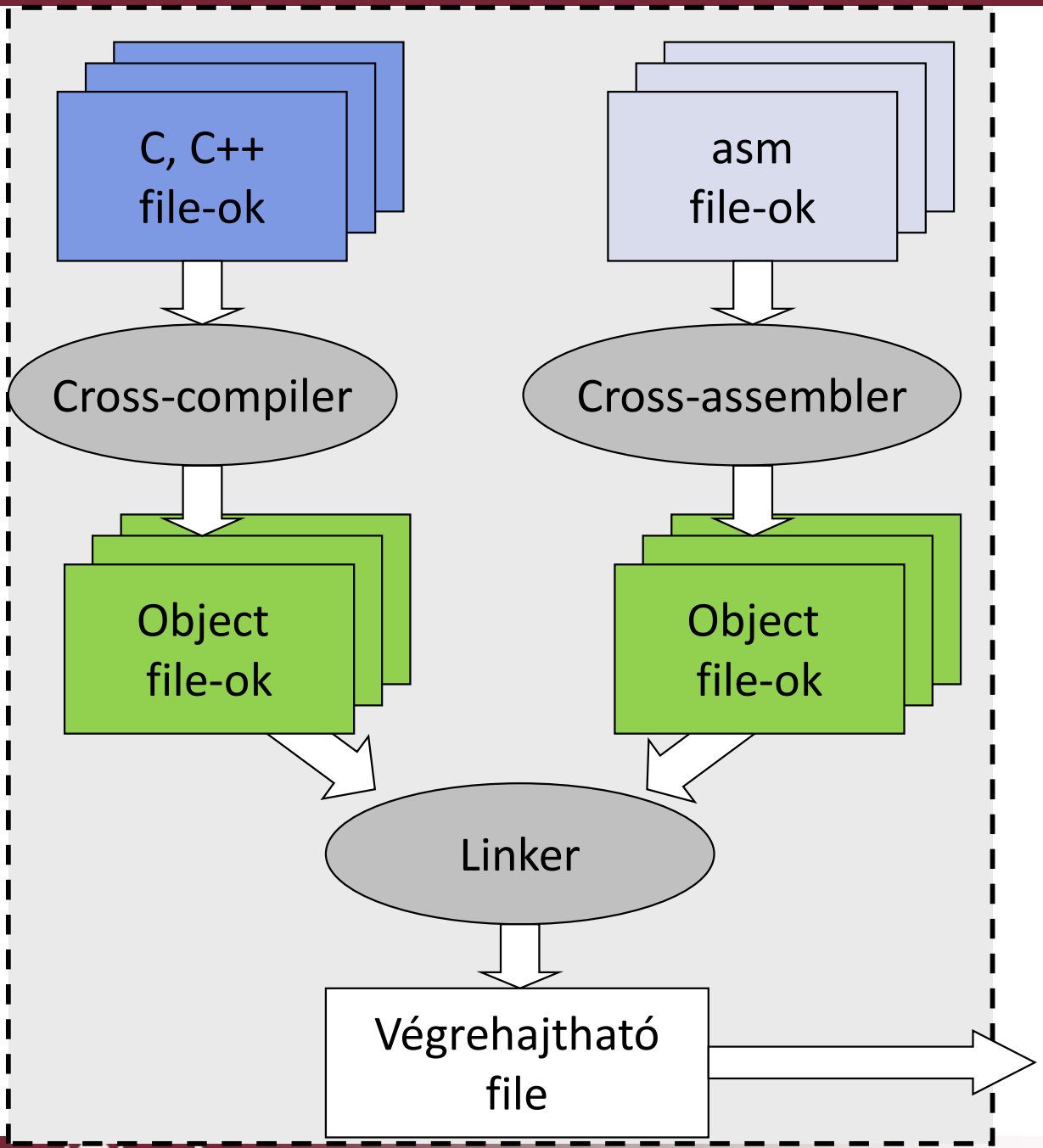
Méréstechnika és
Információs Rendszerek
Tanszék

CMSIS Core


Application Code
Middleware
Code Templates
Third Party Software Pack
CMSIS Software Pack
CMSIS-DSP
DSP Library
CMSIS-RTOS
API
CMSIS-Driver
API
Code Templates
Debugger
Real Time Kernel
3rd Party
Device HAL
Silicon Vendor
CMSIS-CORE
Core Access Functions, SIMD Intrinsics for Cortex® -M4, Peripheral & Interrupt Definitions
CMSIS-SVD
CMSIS-DAP
Cortex®-M CPU
SysTick
RTOS Kernel Timer
NVIC
Nested Vectored Interrupt Controller
Debug + Trace
Other Peripherals
CoreSight™
Microcontroller

Mit tartalmaz a reszet vektor?

Keresztfordíás



Cross compiler különbségek, C forráskód portolhatóság

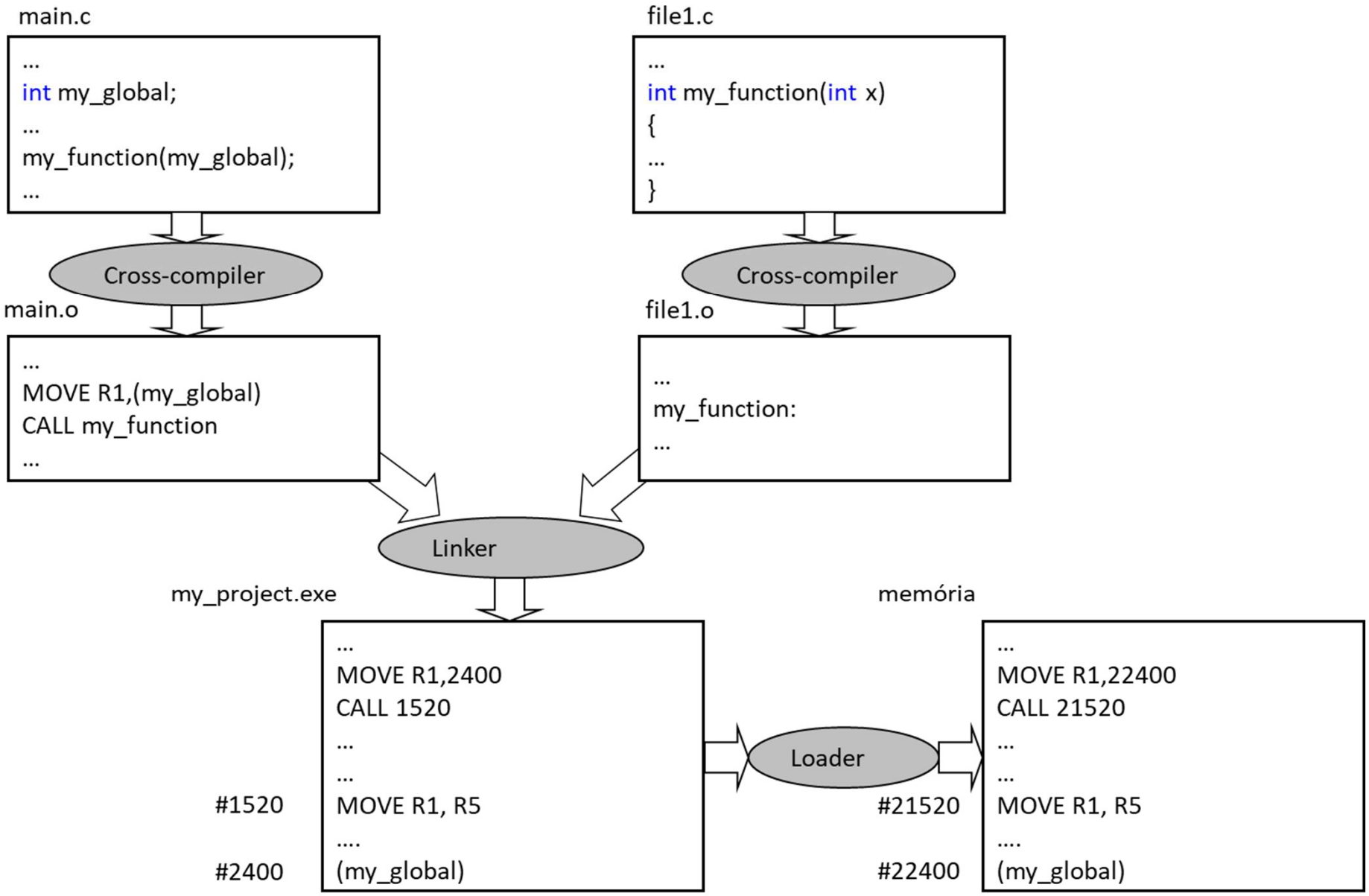
- Általános szerkezetekben nincs különbség
  - If, switch, for
- A változóknál probléma lehet
  - Egy Integer lehet 16/32 bites, a float mérete is változhat
  - Little-Endián Big endian
  - Struktúrák máshogy csomagolódhatnak
- Standard input/output
  - Normál esetben consol, beágyazott esetben lehet sorosport
- Interrupt kezelés
  - Minden target esetében gyakorlatilag máshogy

Cross Linker/locator

- Normál PC-s esetben
  - A helyi gépen tárolódik az elkészült file, nincs szükség letöltésre
  - Indításkor az operációs rendszer load-erje betölti a programot a megfelelő memória és elindítja.
- Beágyazott esetben
  - A programot le kell juttatni a target-re
  - Általában nincs loader, a kód „tisztán” fut

- Cimfeloldás
  - Ugróutasításoknál, változóknál, az object file nem tartalmaz konkrét címet, csak jelzi a linkernek, hogy töltse ezeket a részeket ki.
- Locator
  - A kód elhelyezése a ROM/RAM területek megadott helyére (Ez csak a beágyazott rendszerekre jellemző)

Fordítás folyamata



Szekciók

- El kell helyezni a programot különböző RAM/ROM területekre
  - Programkód ROM
  - Változók RAM
- Szegmensek
  - Program startup
  - Inicializált változók
  - Konstans változók
  - Inicializálatlan adat
  - Stack
- Linker Script
  - Hova kerülnek fizikailag a szegmensek

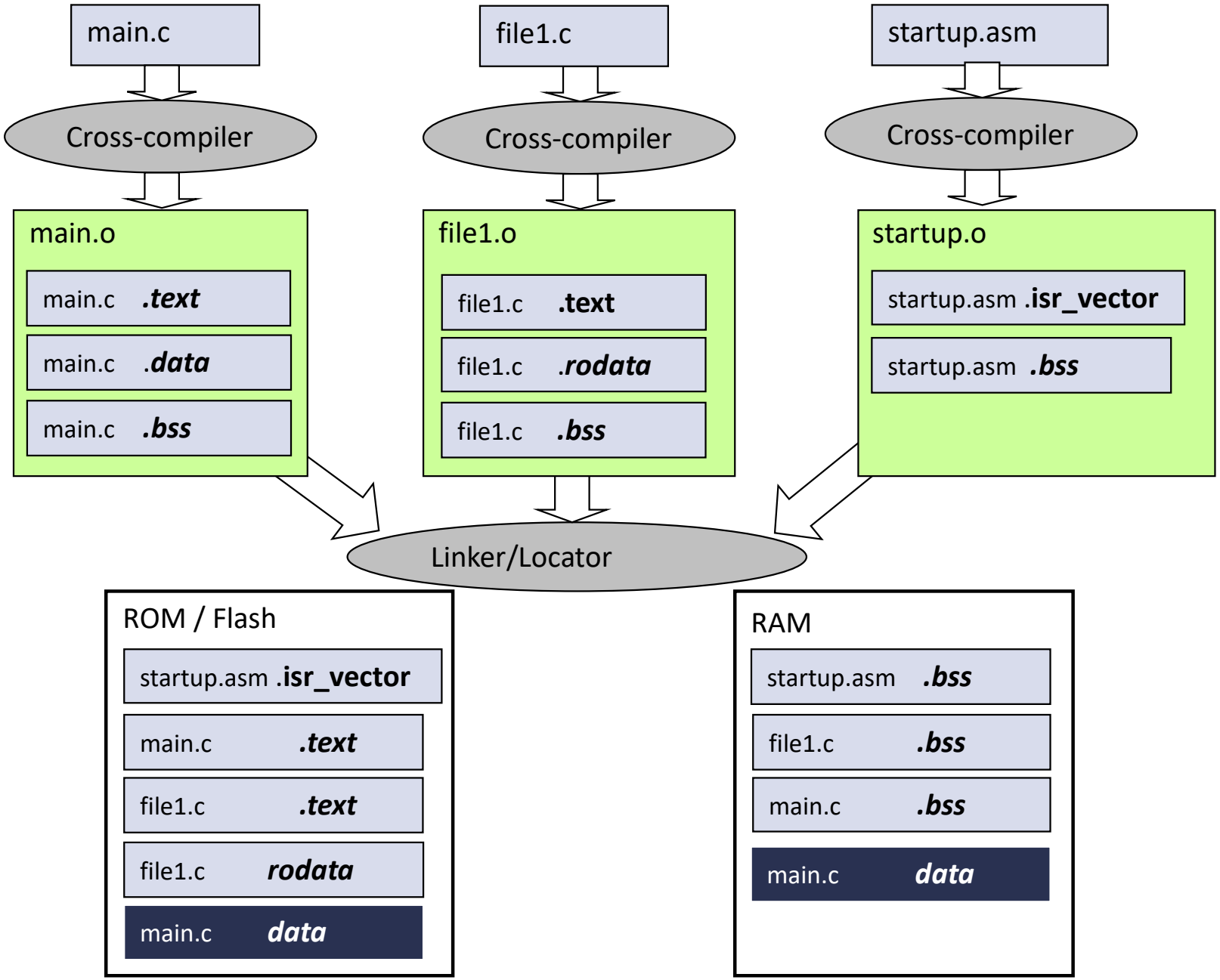
Szekciók

- El kell helyezni a programot különböző RAM/ROM területekre
  - Programkód ROM
  - Változók RAM

- Szegmensek (***GCC nevek***)
  - Program startup: ***.isr_vector***
  - Program: ***.text***
  - Inicializált változók: ***data***
  - Konstans változók: ***.rodata***
  - Inicializálatlan globális adat: ***bss***
  - Stack
  - Heap

- Linker Script
  - Hova kerülnek fizikailag a szegmensek

Szekciók működés közben



Egy Linker script (egyszerűsített)

```
/* Entry Point */
ENTRY(Reset_Handler)

/* Highest address of the user mode stack */
_estack = 0x20030000;    /* end of RAM */
/* Generate a link error if heap and stack don't fit into RAM */
_Min_Heap_Size = 0x200;      /* required amount of heap  */
_Min_Stack_Size = 0x800; /* required amount of stack */

/* Specify the memory areas */
MEMORY
{
RAM (xrw)      : ORIGIN = 0x20000000, LENGTH = 192K
FLASH (rx)      : ORIGIN = 0x8000000, LENGTH = 2048K
}

/* Define output sections */
SECTIONS
{
  /* The startup code goes first into FLASH */
  .isr_vector :
  {
    .........
  } >FLASH
```

Egy Linker script (egyszerűsített)

```
/* Uninitialized data section */
  . = ALIGN(4);
  .bss :
  {
  ……..
  } >RAM

  /* User_heap_stack section, used to check that there is enough RAM left */
  ._user_heap_stack :
  {
    . = . + _Min_Heap_Size;
    . = . + _Min_Stack_Size;
    . = ALIGN(4);
  } >RAM

}
```

Egy Linker script (egyszerűsített)

```
  /* The program code and other data goes into FLASH */
  .text :
  {
    ……
  } >FLASH

  /* Constant data goes into FLASH */
  .rodata :
  {
    …….
  } >FLASH

  /* used by the startup to initialize data */
  _sidata = LOADADDR(.data);

  /* Initialized data sections goes into RAM, load LMA copy after code */
  .data :
  {
  } >RAM AT> FLASH

/* Uninitialized data section */
  . = ALIGN(4);
  .bss :
  {
  ……..
  } >RAM
/* User heap stack section, used to check that there is enough RAM left */
```

The *startup_device* file

- The interrupt table

```
* The minimal vector table for a Cortex M3. Note that the proper constructs
* must be placed on this to ensure that it ends up at physical address
* 0x0000.0000.
*
*******************************************************************************/
   .section  .isr_vector,"a",%progbits
  .type  g_pfnVectors, %object
  .size  g_pfnVectors, .-g_pfnVectors


g_pfnVectors:
  .word  _estack
  .word  Reset_Handler
  .word  NMI_Handler
  .word  HardFault_Handler
  .word  MemManage_Handler
  .word  BusFault_Handler
  .word  UsageFault_Handler
  .word  0
  .word  0
  .word  0
  .word  0
  .word  SVC_Handler
  .word  DebugMon_Handler
  .word  0
  .word  PendSV_Handler
  .word  SysTick_Handler

  /* External Interrupts */
  .word     WWDG_IRQHandler                   /* Window WatchDog              */
  .word     PVD_IRQHandler                    /* PVD through EXTI Line detection */
  .word     TAMP_STAMP_IRQHandler             /* Tamper and TimeStamps through the EXTI line */
```

Annak a megadása, hogy ez a rész a Flash legelejére az IT vektor részre kerül

Az IT vektor tábla, ami az üres helyeket is tartalmazza, hogy minden vektor a megfelelő pozitcióba kerüljön

A Reset_Handler

```
70   .weak  Reset_Handler
71   .type  Reset_Handler, %function
72 Reset_Handler:
73   ldr   sp, =_estack    /* Atollic update: set stack pointer */
74
75 /* Copy the data segment initializers from flash to SRAM */
76   movs  r1, #0
77   b  LoopCopyDataInit
78
79 CopyDataInit:
80   ldr  r3, =_sidata
81   ldr  r3, [r3, r1]
82   str  r3, [r0, r1]
83   adds  r1, r1, #4
84
85 LoopCopyDataInit:
86   ldr  r0, =_sdata
87   ldr  r3, =_edata
88   adds  r2, r0, r1
89   cmp  r2, r3
90   bcc  CopyDataInit
91   ldr  r2, =_sbss
92   b  LoopFillZerobss
93 /* Zero fill the bss segment. */
94 FillZerobss:
95   movs  r3, #0
96   str  r3, [r2], #4
97
98 LoopFillZerobss:
99   ldr  r3, = _ebss
100  cmp  r2, r3
101  bcc  FillZerobss
102
103 /* Call the clock system intitialization function.*/
104  bl  SystemInit
105 /* Call static constructors */
106    bl __libc_init_array
107 /* Call the application's entry point.*/
108  bl  main
109  bx  lr
110 .size  Reset_Handler, .-Reset_Handler
```

A Stack pointer beállítása

Az inicializált változók kezdőértékének átmásolása a Flash-ből a RAM-ba

Az inicializáltalan változók helyeinek nullázása

A CMSIS SystemInit függvényének meghívása, hogy stabil, ismert órajellel tudjunk tovább menni

A main függvény meghívása

LEDS villogtatás a STM32F429 Discovery-n

- User LD3: Zöld LED a PG13-ra kötve
- User LD4: A piros LED a PG14-ra kötve

GPIOG blokk órajel engedélyezés

6.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Res.	**DMA2DEN**	DMA2EN	DMA1EN	CCMDATARAMEN	Res.	BKPSRAMEN	Reserved	
	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 6 **GPIOGEN:** IO port G clock enable

This bit is set and cleared by software.

0: IO port G clock disabled

1: IO port G clock enabled

GPIO leírás egy modern vezérlőnél

Table 39. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOx_MODER** (where x = C..I/J/K)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**GPIOx_ OTYPER** (where x = A..I/J/K)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_ OSPEEDR** (where x = A..I/J/K except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOx_PUPDR** (where x = C..I/J/K)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = A..I/J/K)	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

GPIO lábak kimenetnek konfigurálása

Figure 25. Basic structure of a five-volt tolerant I/O port bit

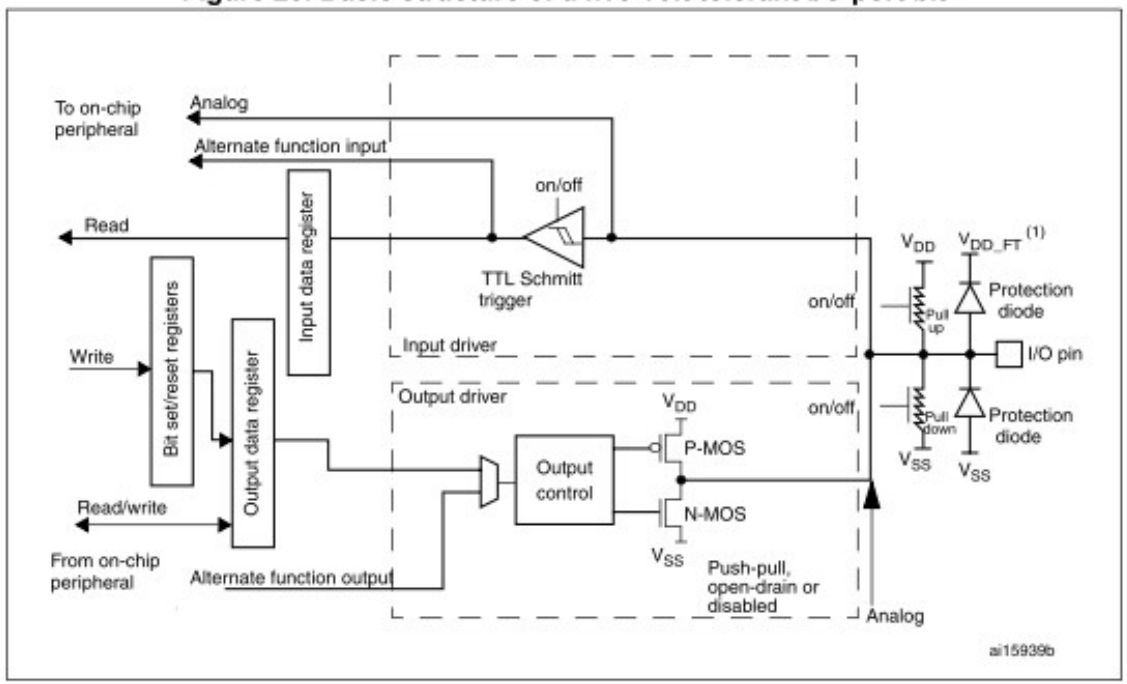


Table 35. Port bit configuration table(1)

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

GPIO lábak kimenetnek konfigurálása

- Setting port pins to output

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

GPIO állapot manipulálás

- Setting pins value to 1 or 0

8.4.6 GPIO port output data register (GPIOx_ODR) (x = A..I/J/K)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

GPIO állapot manipulálás

- Setting pins value to 1 or 0

8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I/J/K)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

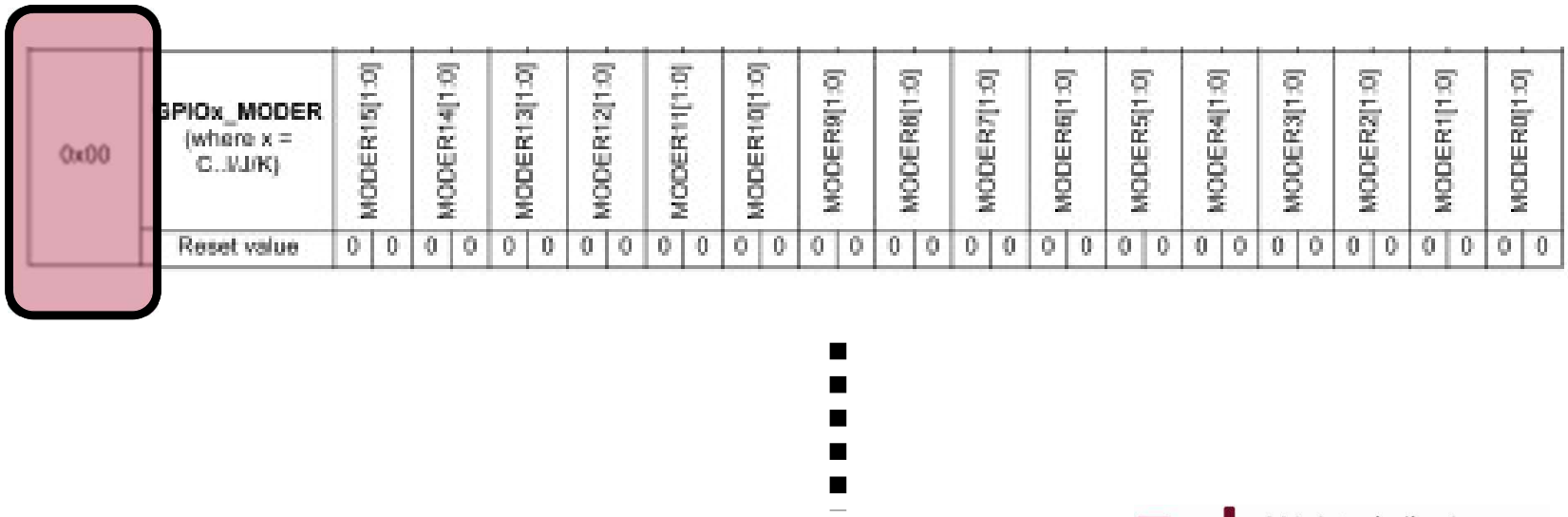
1: Sets the corresponding ODRx bit

Regiszterek szükséges tartalma összefglalva

- RCC_AHB1ENR: bit 6 -> 1
- GPIOG_MODER: bit 26,28 -> 1
- GPIOG_BSRR: Set reszet regiszter (az alsó 16 biten lehet 1-be állítani a lábakat) tehát 13,14-es bit a LED vezérlő lábak 1-be állítása, a felső 16 biten pedig lehet törölni ezeket a biteket (29, 30)

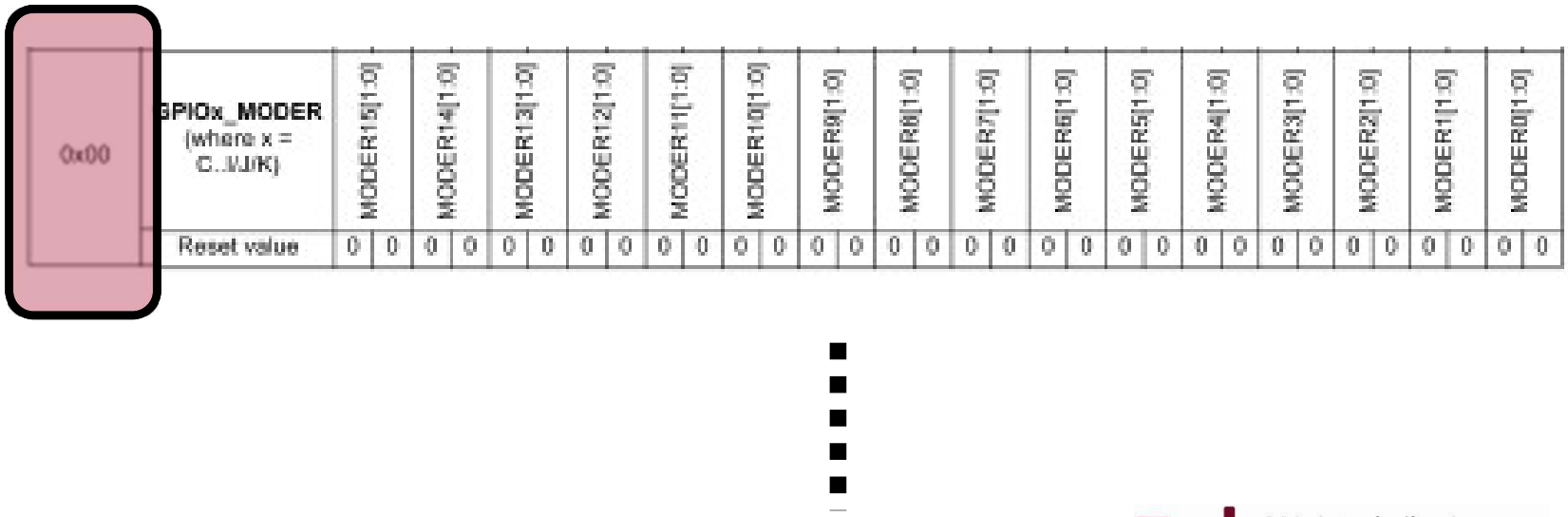
GPIO Register Map

- Minden GPIO blokk sok regiszterből áll, de minden GPIO blokk ugyanazokat a regisztereket tartalmazza
  - GPIOG starts at: 0x4002 1800



Regiszter hozzáférés

- Minden GPIO blokk sok regiszterből áll, de minden GPIO blokk ugyanazokat a regisztereket tartalmazza
  - GPIOG starts at: 0x4002 1800
  - Régimódi hozzáférés a GPIOG_MODER regiszterhez:
    - #define GPIOG_MODER *((uint32_t*)(0x40021800))



Az összes regiszter ofszettel megadva

Table 39. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOx_MODER (where x = C..I/J/K)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPIOx_ OTYPER (where x = A..I/J/K)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_ OSPEEDR (where x = A..I/J/K except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (where x = C..I/J/K)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (where x = A..I/J/K)	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Azonos tulajdonságú perifériák több példányban

0x4002 2800 - 0x4002 2BFF	GPIOK
0x4002 2400 - 0x4002 27FF	GPIOJ
0x4002 2000 - 0x4002 23FF	GPIOI
0x4002 1C00 - 0x4002 1FFF	GPIOH
0x4002 1800 - 0x4002 1BFF	GPIOG
0x4002 1400 - 0x4002 17FF	GPIOF
0x4002 1000 - 0x4002 13FF	GPIOE
0X4002 0C00 - 0x4002 0FFF	GPIOD
0x4002 0800 - 0x4002 0BFF	GPIOC
0x4002 0400 - 0x4002 07FF	GPIOB
0x4002 0000 - 0x4002 03FF	GPIOA

CMSIS definiciók és szabályok

- Struktúrák használata regiszter blokkok leírására

```
typedef struct
{
  __IO uint32_t MODER;    /*!< GPIO port mode register,                Address offset: 0x00      */
  __IO uint32_t OTYPER;   /*!< GPIO port output type register,         Address offset: 0x04      */
  __IO uint32_t OSPEEDR;  /*!< GPIO port output speed register,        Address offset: 0x08      */
  __IO uint32_t PUPDR;    /*!< GPIO port pull-up/pull-down register,   Address offset: 0x0C      */
  __IO uint32_t IDR;      /*!< GPIO port input data register,          Address offset: 0x10      */
  __IO uint32_t ODR;      /*!< GPIO port output data register,         Address offset: 0x14      */
  __IO uint16_t BSRRL;    /*!< GPIO port bit set/reset low register,   Address offset: 0x18      */
  __IO uint16_t BSRRH;    /*!< GPIO port bit set/reset high register,  Address offset: 0x1A      */
  __IO uint32_t LCKR;     /*!< GPIO port configuration lock register,  Address offset: 0x1C      */
  __IO uint32_t AFR[2];   /*!< GPIO alternate function registers,      Address offset: 0x20-0x24 */
} GPIO_TypeDef;
```

CMSIS definiciók és szabályok

- Előre definiált memória hozzárendelések

```
#define GPIOA                ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE                ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOF                ((GPIO_TypeDef *) GPIOF_BASE)
#define GPIOG                ((GPIO_TypeDef *) GPIOG_BASE)
#define GPIOH                ((GPIO_TypeDef *) GPIOH_BASE)
#define GPIOI                ((GPIO_TypeDef *) GPIOI_BASE)
#define GPIOJ                ((GPIO_TypeDef *) GPIOJ_BASE)
#define GPIOK                ((GPIO_TypeDef *) GPIOK_BASE)
```

- Ezek alapján a GPIOG_MODER regiszteréhez való hozzáférés:
  - GPIOG->MODER