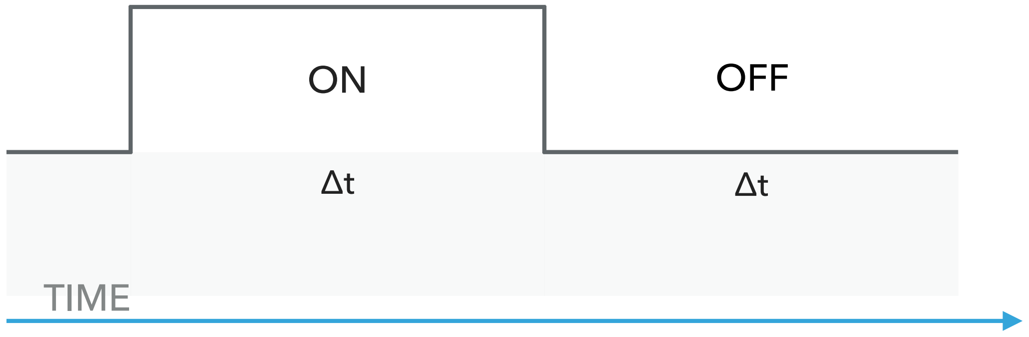
BLINKING LED IN FORTH

- To make the LED blink the phrases `LED ON` and `LED OFF` can be executed one after the other interleaving the executions with a delay. A possible way to do it simply requires the definition of the new word `DELAY ( n -- )`:

```
LED ON  10000 DELAY  LED OFF
```

- A second delay can be added to make the LED transition visible, for example by holding the LED on and off for the same amount of time:

```
LED ON  10000 DELAY  LED OFF  10000 DELAY
```

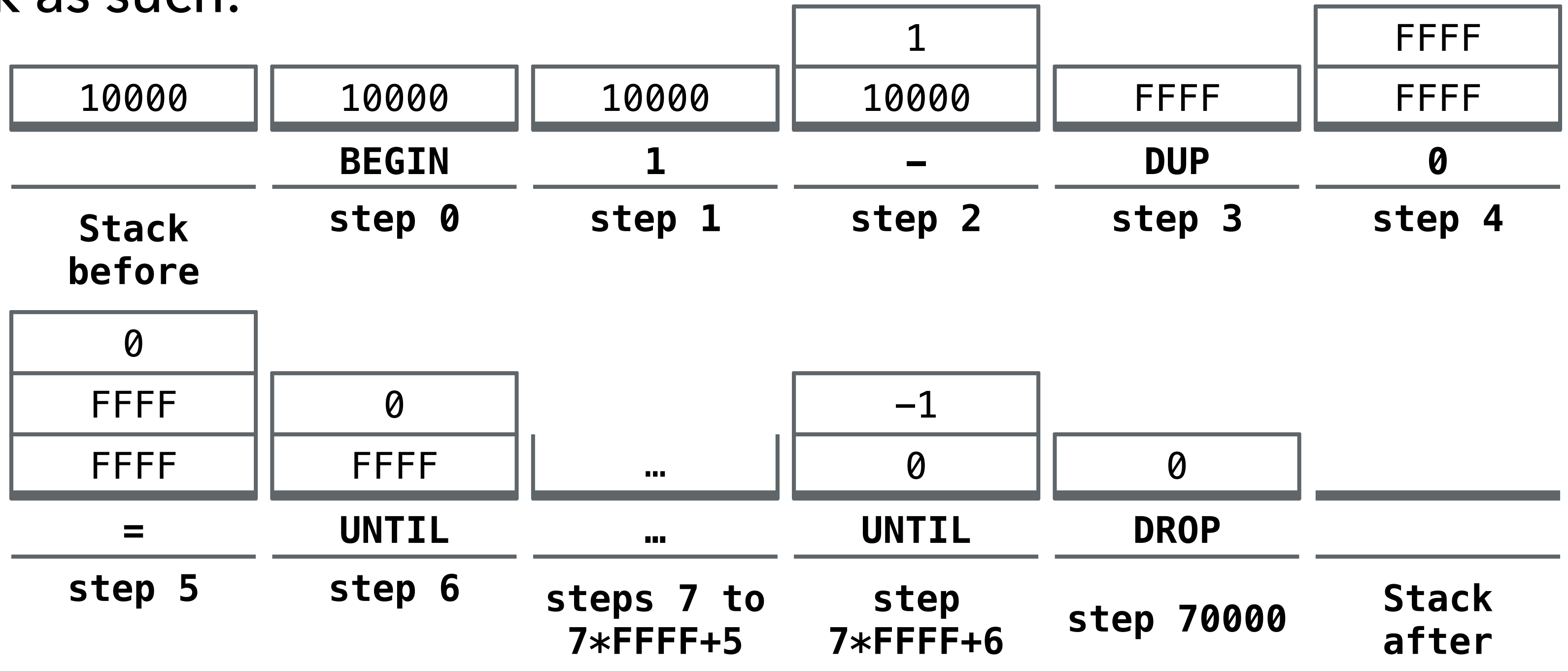


BLINKING LED IN FORTH

- As a first attempt, the DELAY word can be defined using a busy loop

```
: DELAY    BEGIN 1 -  DUP  0 =  UNTIL DROP ;
```

- The word consumes the TOS and leaves no values on the stack. The **steps** of the execution of DELAY with 0x10000 as TOS value modify the stack as such:

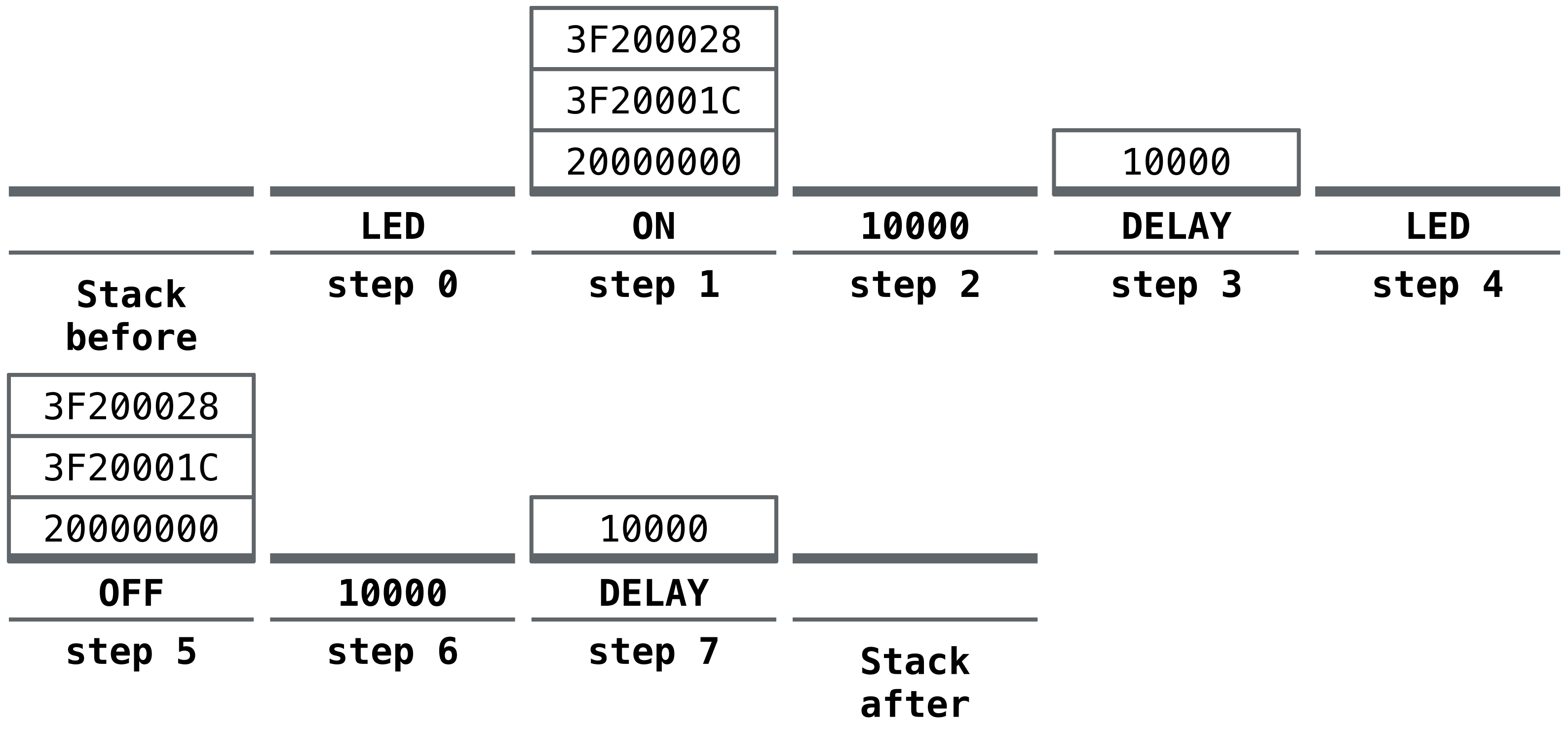


BLINKING LED IN FORTH

- The execution of the sequence:

```
LED ON  10000 DELAY  LED OFF  10000 DELAY
```

- has the following effects on the stack:

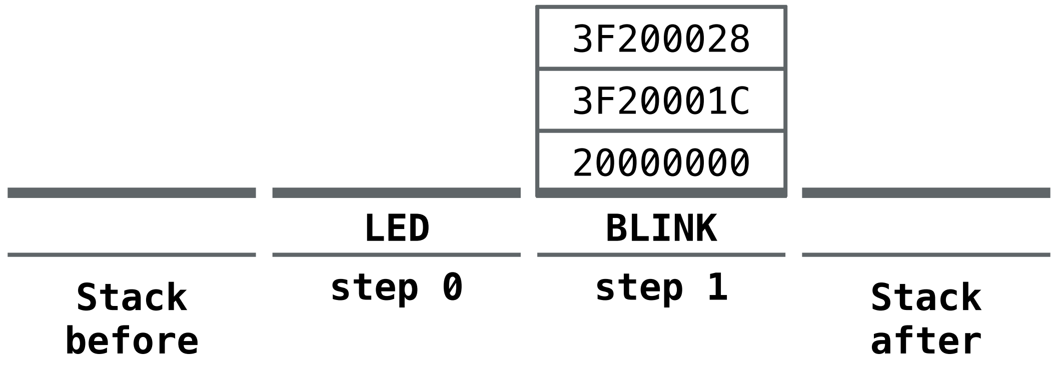


BLINKING LED IN FORTH

- The code to blink the LED could be placed into the word BLINK so that the phrase:

```
LED BLINK
```

- would blink the LED instead of the long sequence seen before



BLINKING LED IN FORTH

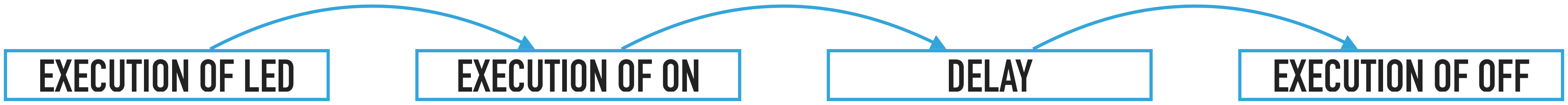
- LED is executed just once in the phrase:

```
LED BLINK
```

- However,
  - both ON and OFF need the same three values provided by LED

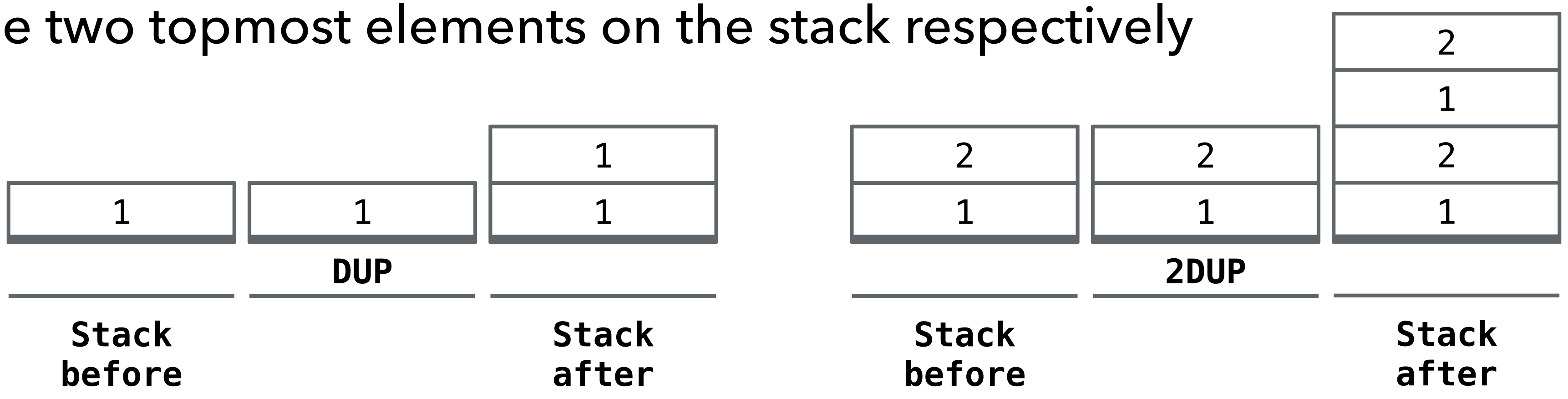
```
LED .S
20000000 3F20001C 3F200028
```

  - both consume the three values
- The three values put on the stack by LED must be preserved before the execution of ON so that OFF could find them

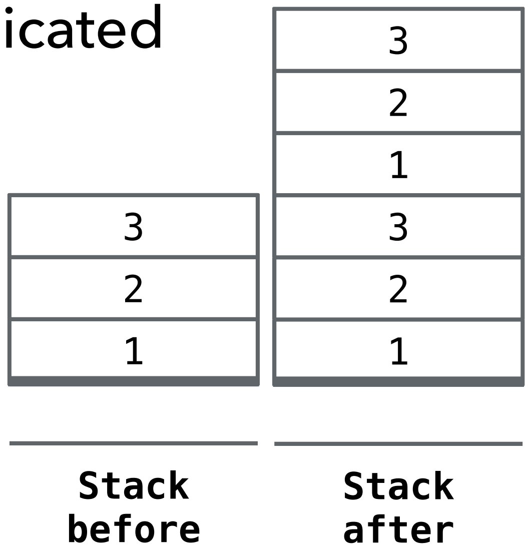


BLINKING LED IN FORTH

- The standard words DUP and 2DUP duplicate the topmost element and the two topmost elements on the stack respectively



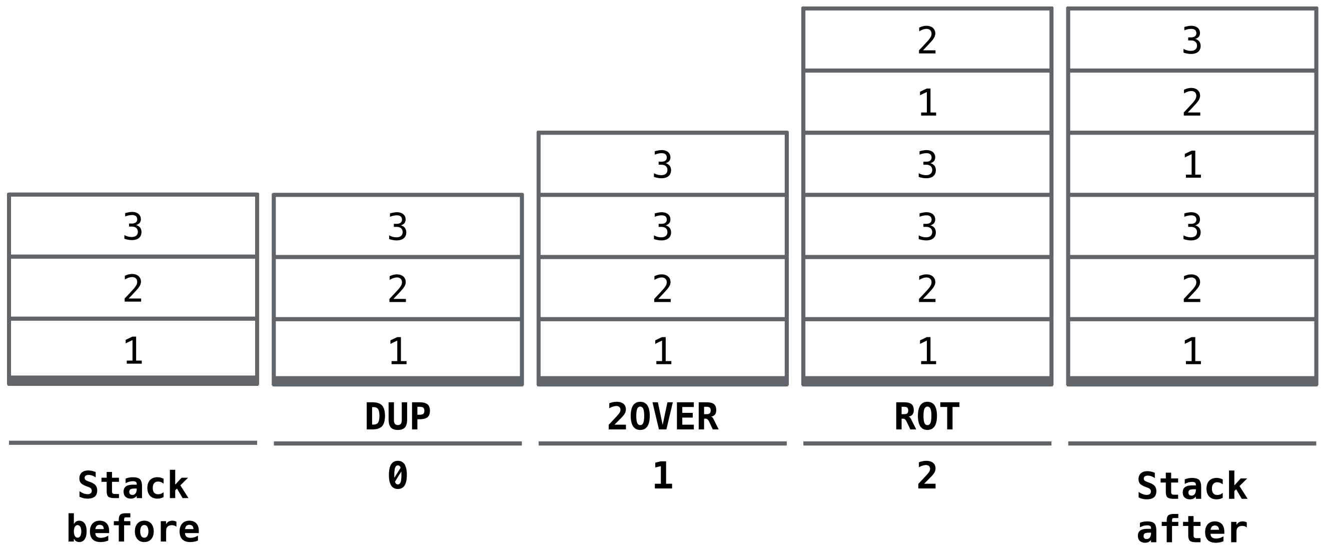
- In this case, a set of three values has to be duplicated
- A new word can be defined for this purpose



BLINKING LED IN FORTH

- The word 3DUP is defined as such:

```
: 3DUP   DUP 2OVER ROT ;
```

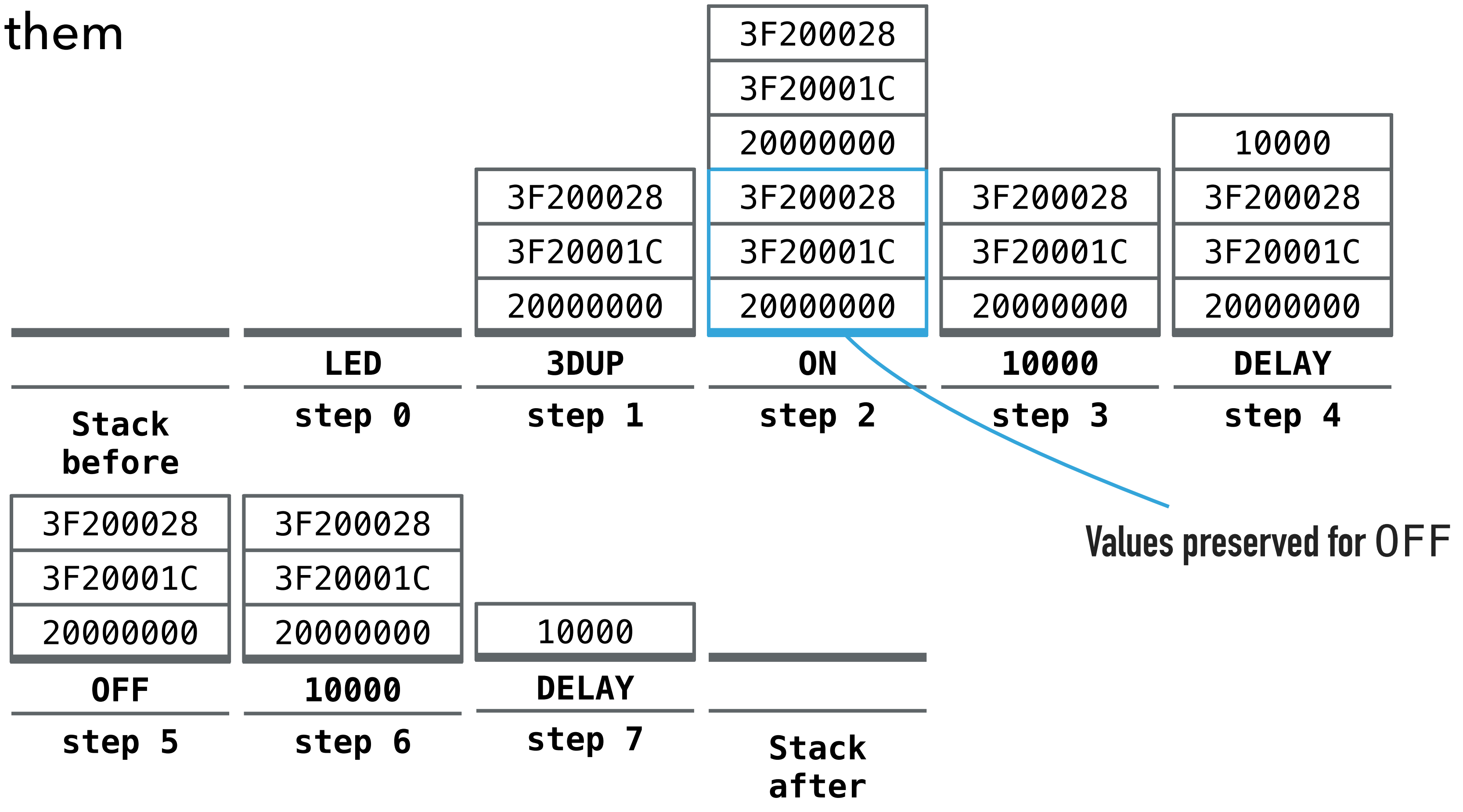


- The word BLINK can be now defined:

```
: BLINK   3DUP  ON  10000 DELAY  OFF  10000 DELAY ;
```

BLINKING LED IN FORTH

- The three values put on the stack by LED are preserved on the stack by the execution of 3DUP before the execution of ON so that OFF could find them

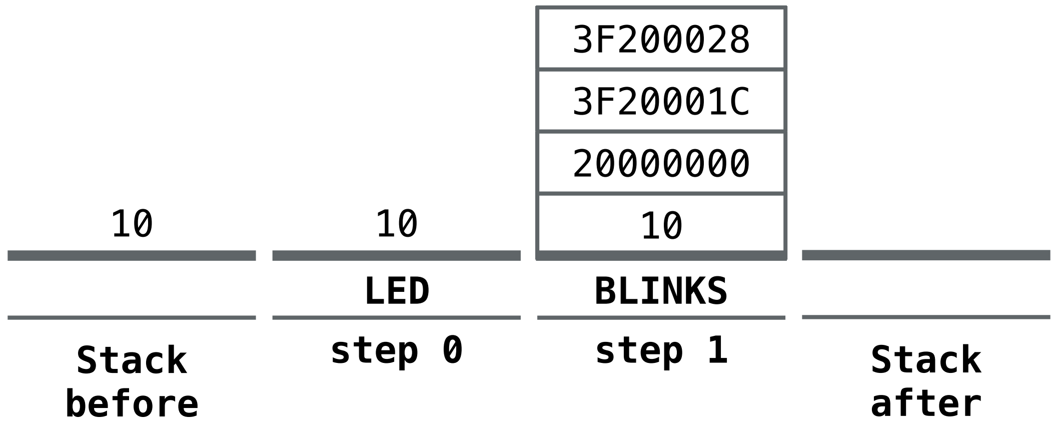


BLINKS

- The LED driving language could be extended with the word BLINKS so that the phrase:

```
10 LED BLINKS
```

- would blink the LED 10 times



BLINKS

- A quick and long definition for BLINKS could be this:

```
VARIABLE COUNTER
: BLINKS 2OVER DROP COUNTER ! BEGIN 3DUP BLINK COUNTER @ 1 - DUP COUNTER ! 0 = UNTIL
2DROP 2DROP ;
```

- To be continued…

HARDWARE SPECS

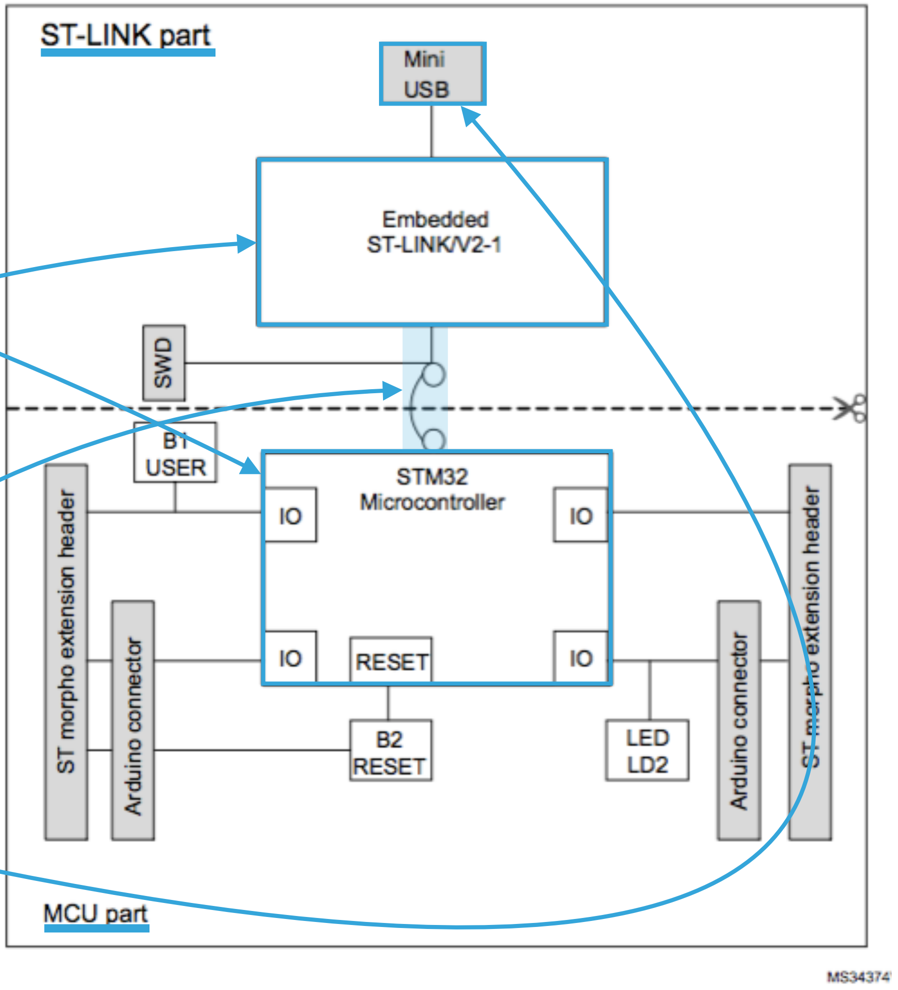
- The Nucleo-64 STM32F446 belongs to the Nucleo-64 family from ST microelectronics
  - Each board is provided with an STM32 MCU
  - The boards provide combinations of performance, power consumption and features
  - ArduinoTM Uno V3 connectivity support
  - ST morpho headers
  - Integrated ST-LINK/V2-1 debugger and programmer

UM1724 User manual - (en.DM00105823.pdf)

HARDWARE SPECS

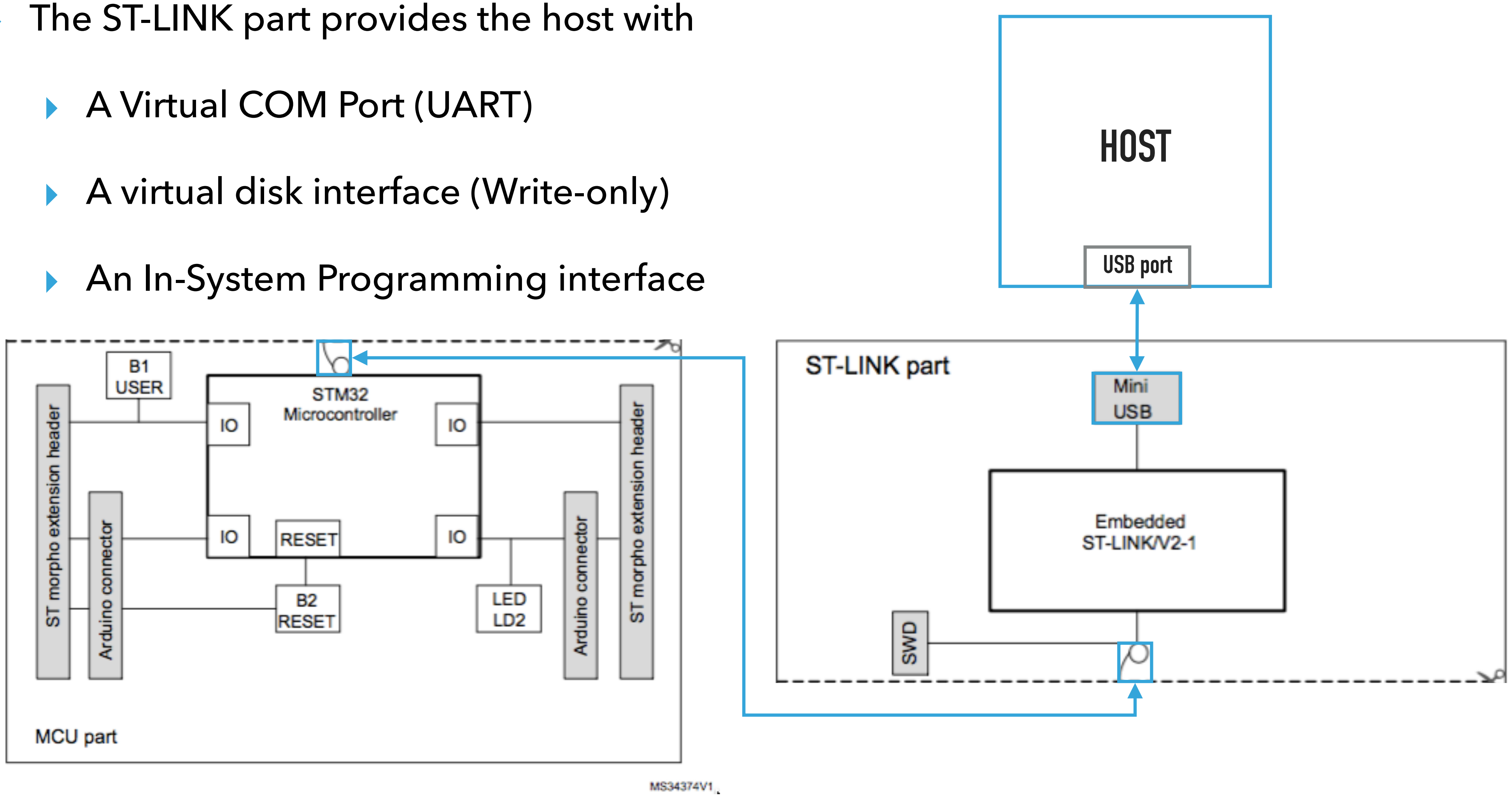
The Nucleo-64 STM32F446 board is provided with:

- An STM32F446 MCU
- Arduino connectors
- ST morpho connectors
- LEDs (2)
- Buttons (2)
- An embedded ST-Link debugger and programmer using another STM32 MCU, which is factory-programmed
    - Connected to
        - the MCU through a Serial Wire Debug (SWD) interface (JTAG)
        - a development machine through a mini USB port
    - Provides power to the MCU part
- The MCU part is detachable



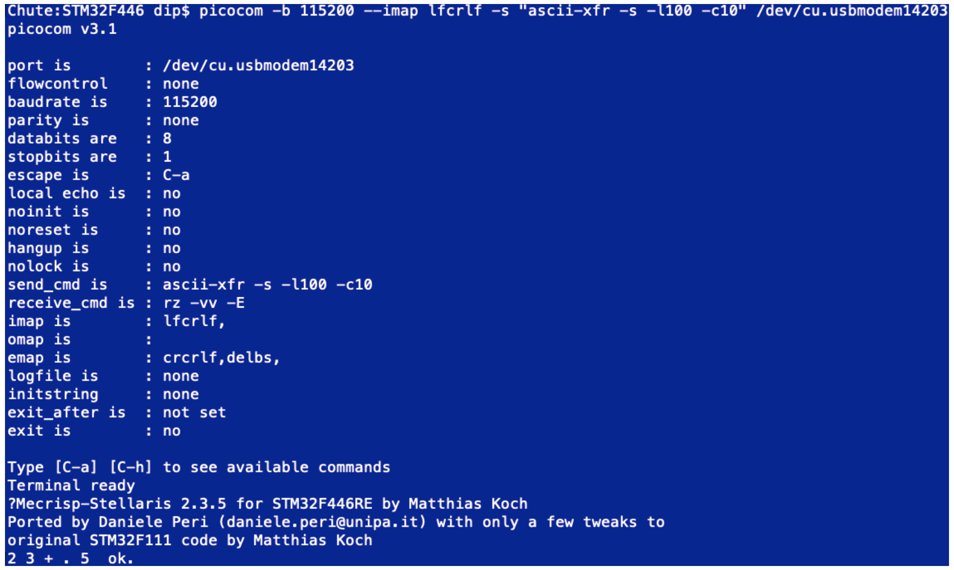
HARDWARE SPECS

- The ST-LINK part provides the host with
  - A Virtual COM Port (UART)
  - A virtual disk interface (Write-only)
  - An In-System Programming interface

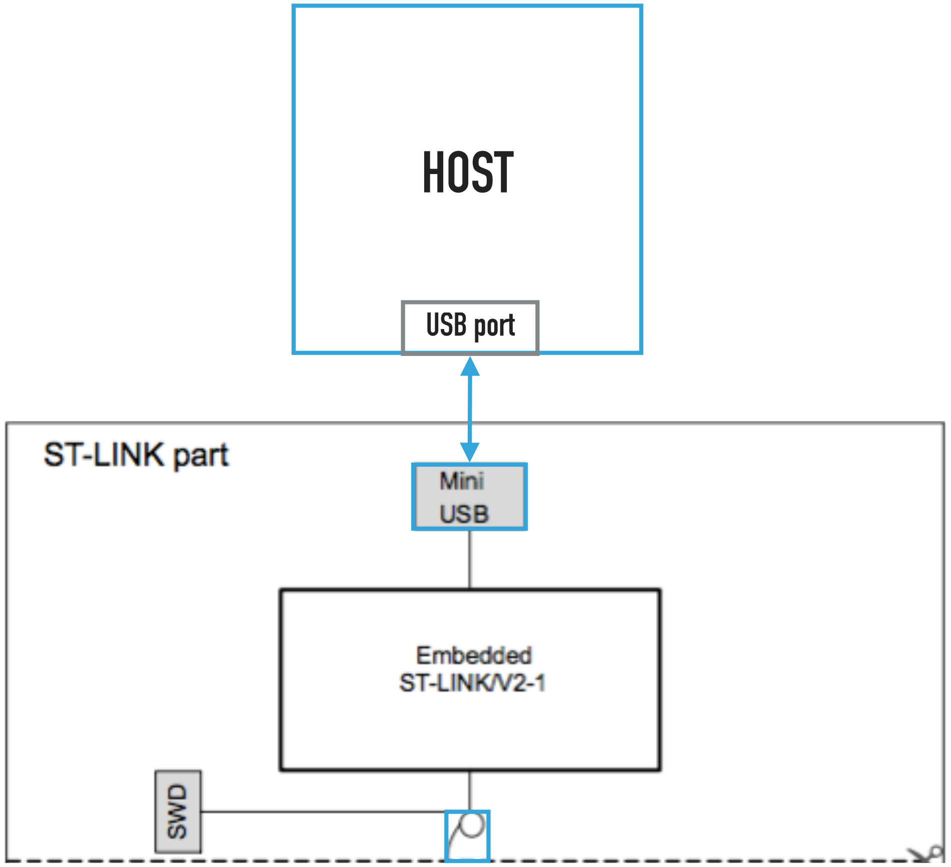


HARDWARE SPECS

- The Virtual COM Port (UART) can be used for
    - Debug
    - I/O
    - Interactive programming
        - Forth

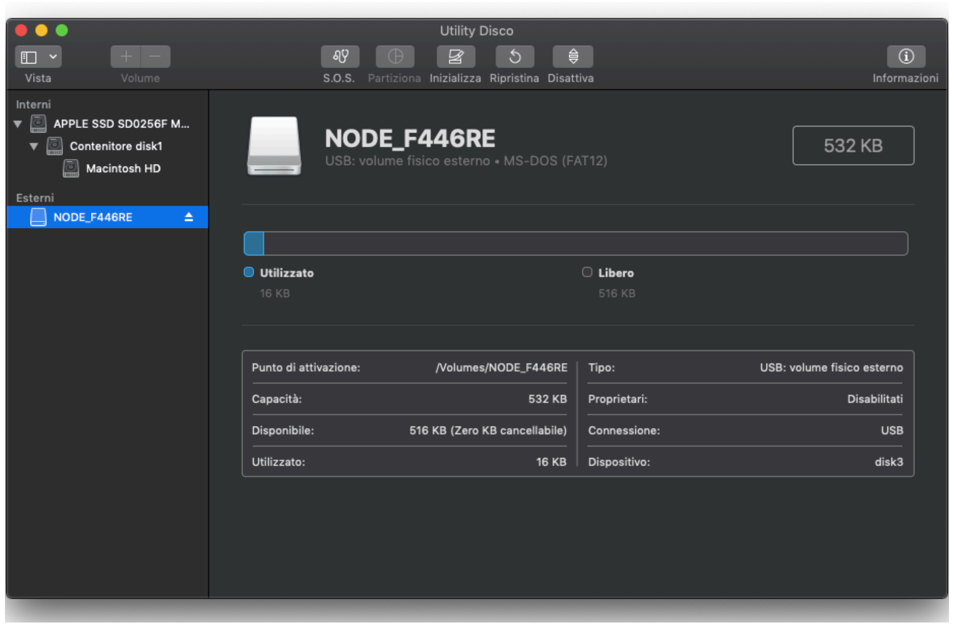


```
Chute:STM32F446 dip$ picocom -b 115200 --imap lfcrlf -s "ascii-xfr -s -l100 -c10" /dev/cu.usbmodem14203
picocom v3.1

port is        : /dev/cu.usbmodem14203
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
hangup is      : no
nolock is      : no
send_cmd is    : ascii-xfr -s -l100 -c10
receive_cmd is : rz -vv -E
imap is        : lfcrlf,
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready
?Mecrisp-Stellaris 2.3.5 for STM32F446RE by Matthias Koch
Ported by Daniele Peri (daniele.peri@unipa.it) with only a few tweaks to
original STM32F111 code by Matthias Koch
2 3 + . 5  ok.
```



HARDWARE SPECS

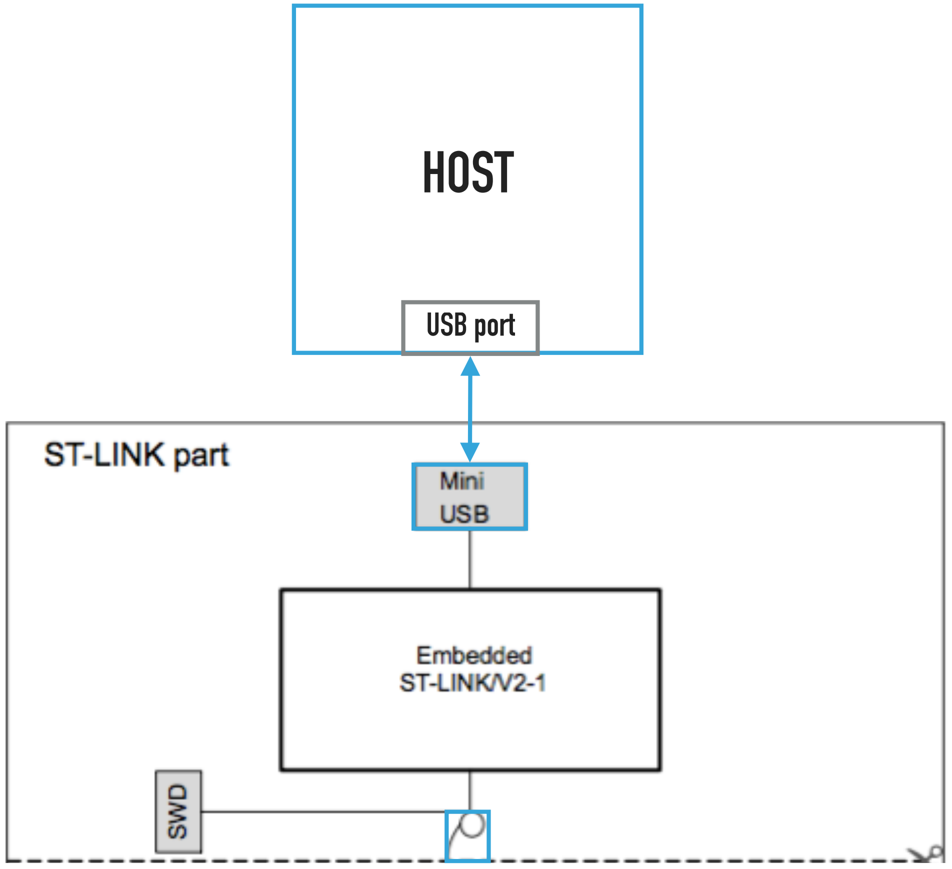
- The virtual disk interface let the user write an image to the flash memory just by dropping it in the virtual filesystem root (mbed)



NODE_F446RE
USB: volume fisico esterno • MS-DOS (FAT12)
532 KB

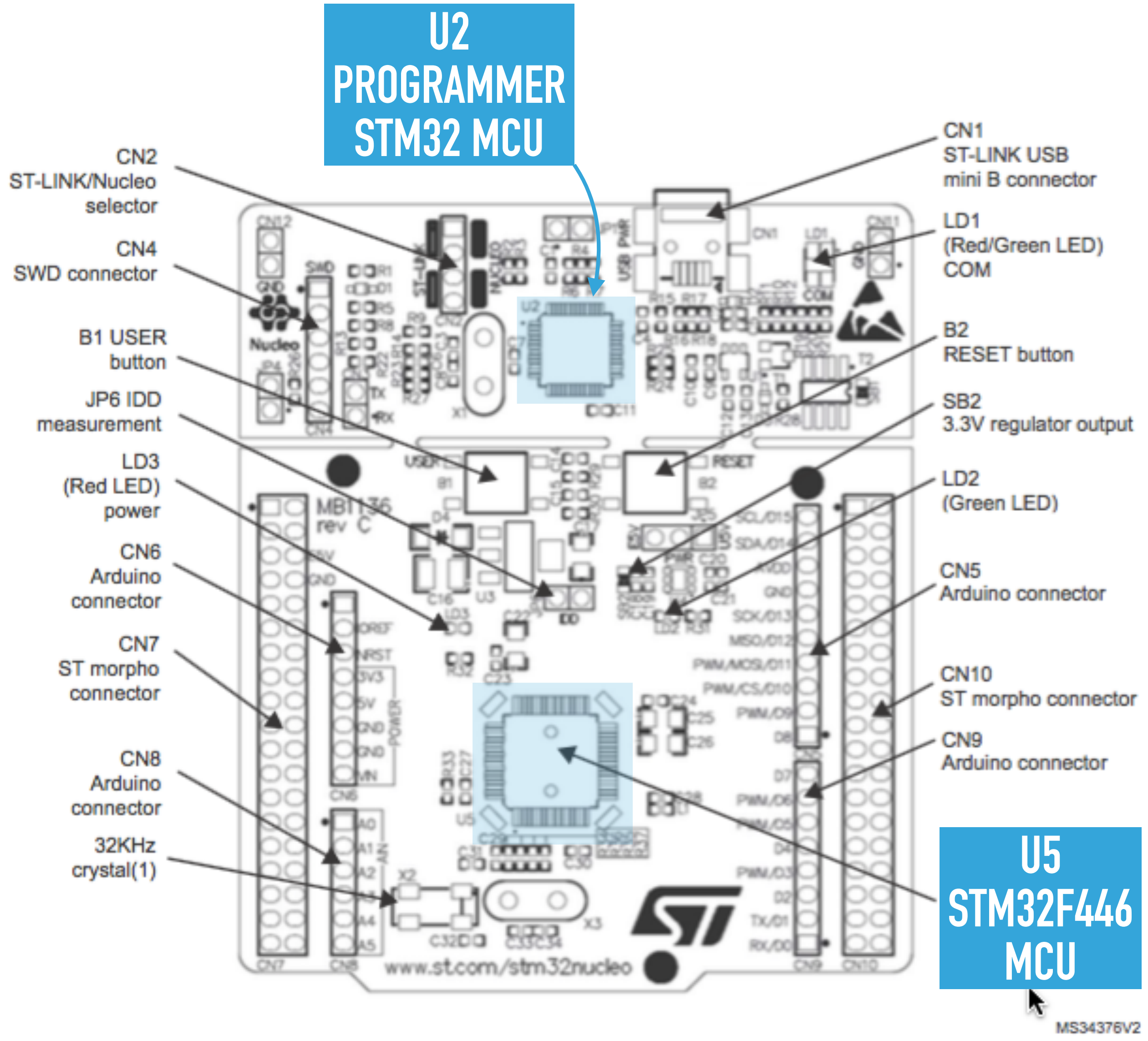
● Utilizzato 16 KB
○ Libero 516 KB

Punto di attivazione:	/Volumes/NODE_F446RE	Tipo:	USB: volume fisico esterno
Capacità:	532 KB	Proprietari:	Disabilitati
Disponibile:	516 KB (Zero KB cancellabile)	Connessione:	USB
Utilizzato:	16 KB	Dispositivo:	disk3



HARDWARE SPECS

- Board components
  - Chips (Ux)
  - Connectors (CNx)
  - Buttons (Bx)
  - LEDs (LEDx)
  - Jumpers (JPx)



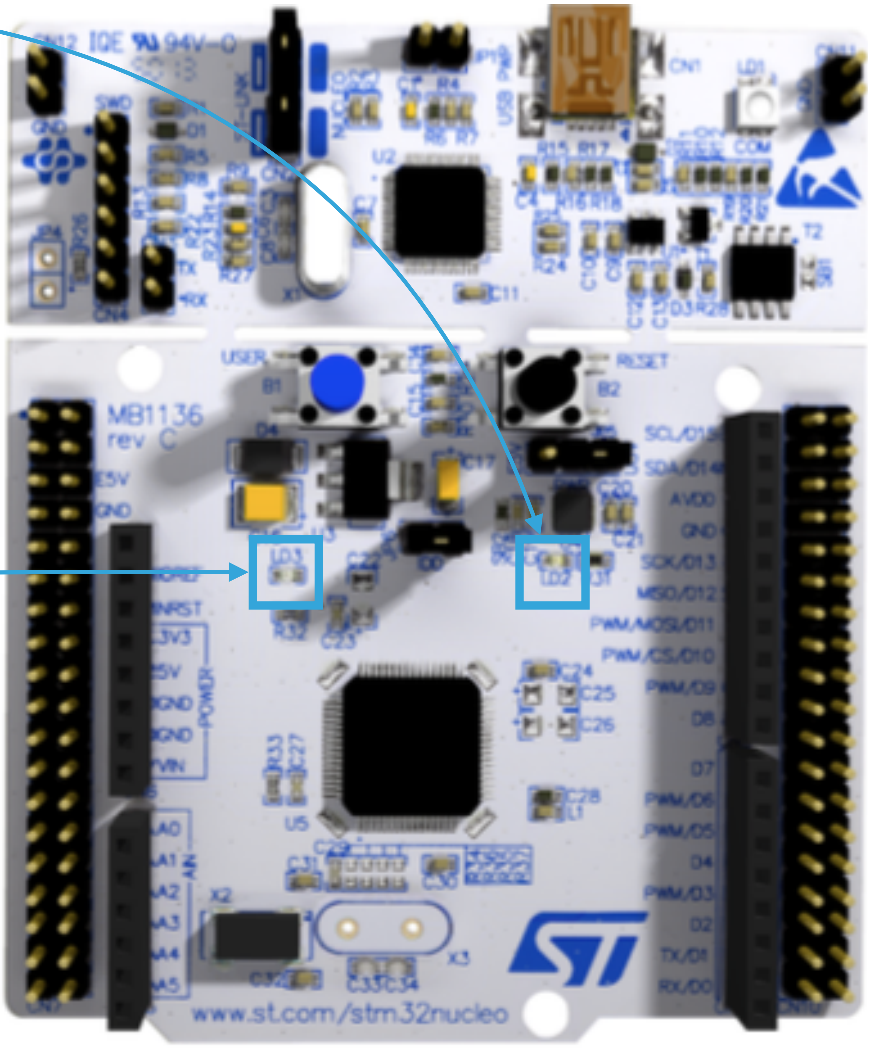
HARDWARE SPECS

- The tricolor LED (green, orange, red) LD1 (COM) provides information about ST-LINK communication status.
- LD1 default color is red. LD1 turns to green to indicate that communication is in progress between the PC and the ST-LINK/V2-1, with the following setup:
    - Slow blinking Red/Off: at power-on before USB initialization
    - Fast blinking Red/Off: after the first correct communication between the PC and ST-LINK/V2-1 (enumeration)
    - Red LED On: when the initialization between the PC and ST-LINK/V2-1 is complete
    - Green LED On: after a successful target communication initialization
    - Blinking Red/Green: during communication with target
    - Green On: communication finished and successful
    - Orange On: Communication failure

UM1724 User manual - (en.DM00105823.pdf)

HARDWARE SPECS

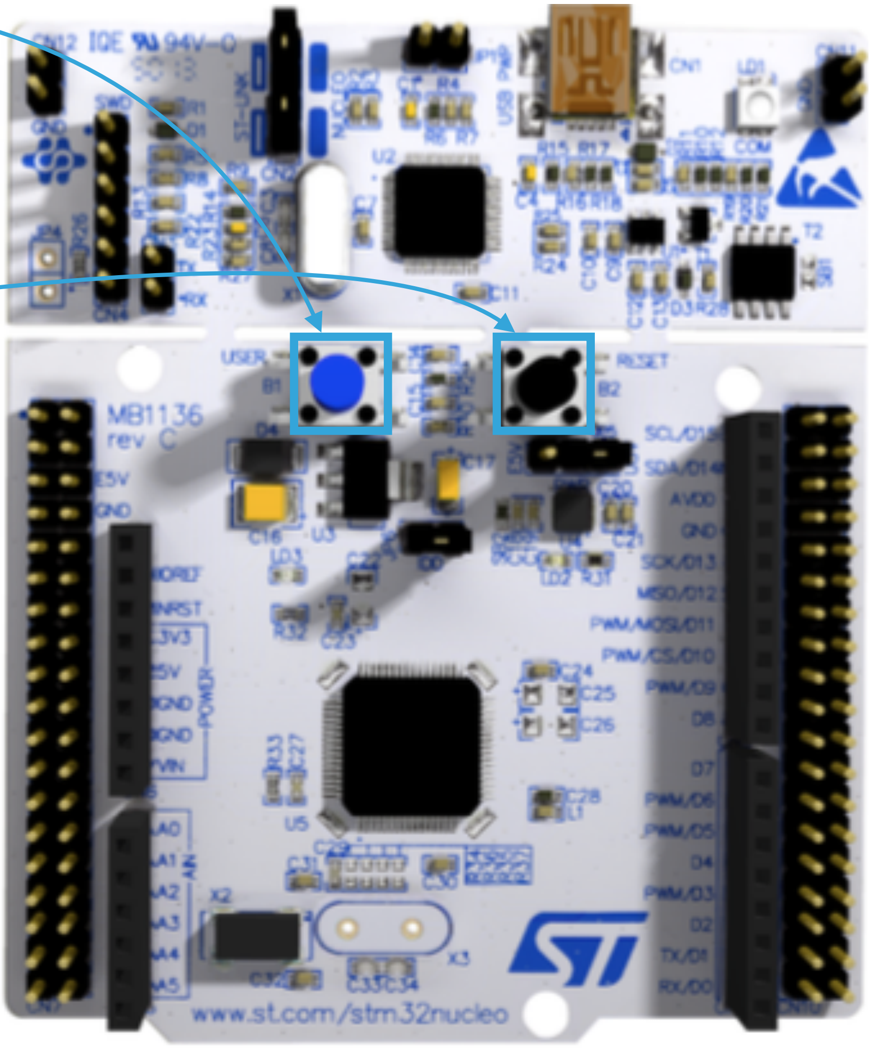
- LD2: the green LED is a user LED connected to Arduino signal D13 corresponding to one (target dependent) of:
  - STM32 I/O PA5 (pin 21)
  - PB13 (pin 34)
- LD2 is on when the I/O is HIGH, is off when the I/O is LOW
- LD3: the red LED indicates that the STM32 part is powered and +5V power is available.



UM1724 User manual - (en.DM00105823.pdf)

HARDWARE SPECS

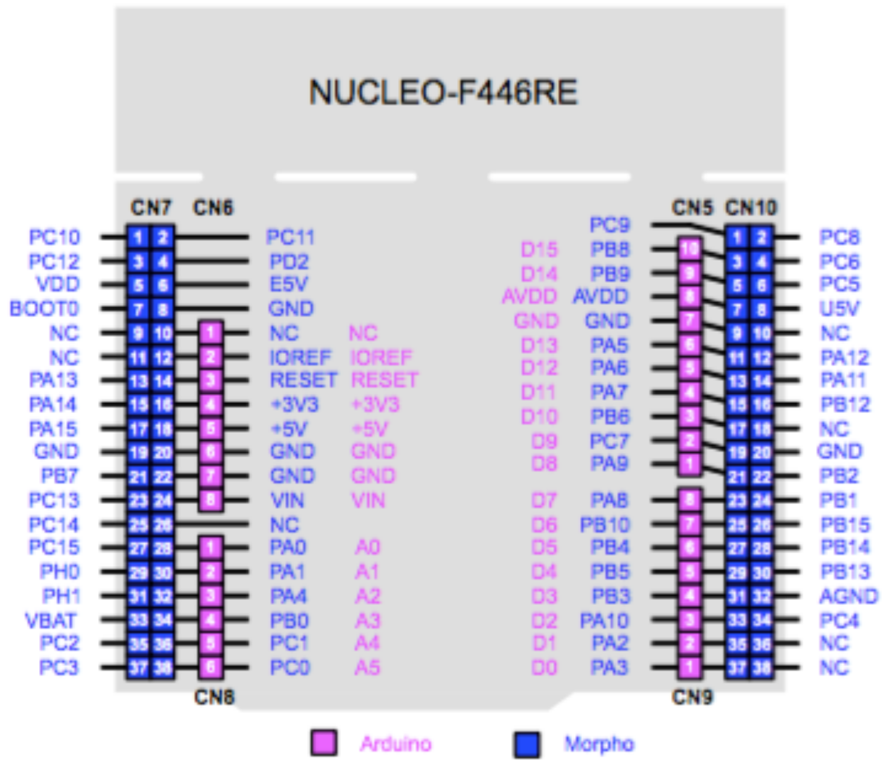
- B1 USER: the user button is connected to the I/O PC13 (pin 2) of the STM32 MCU
- B2 RESET: this push-button is connected to NRST (chip pin), and is used to RESET the STM32 MCU



GPIO

- The SoC is provided with 8 16-pin GPIO ports:
  - GPIOx x=A..H
- Some ports are connected to external pins in:
  - Arduino connectors (CN5, CN6)
  - Proprietary Morpho bus connectors (CN7, CN10)

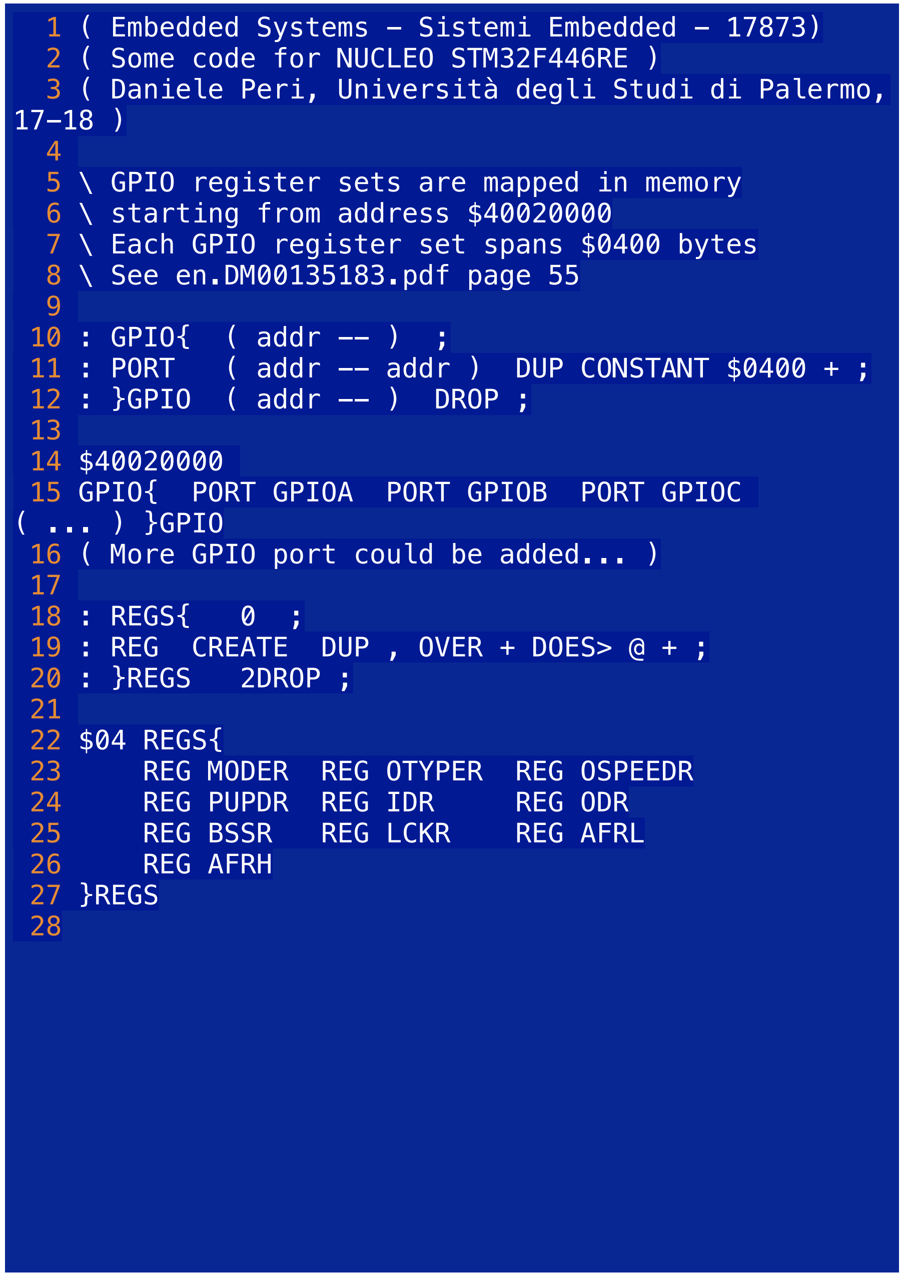
Figure 25. NUCLEO-F446RE



PROGRAMMING ENVIRONMENTS

Several programming environments are available

- Proprietary
  - ST IDE
    - Include Configurator for the entire ST MCU family
- Open Source
  - Arduino IDE
  - GCC C/C++ toolchain
  - Mecrisp-Stellaris Forth Environment
- Web Based
  - Mbed WEB IDE (Mbed OS) - Industry sponsored



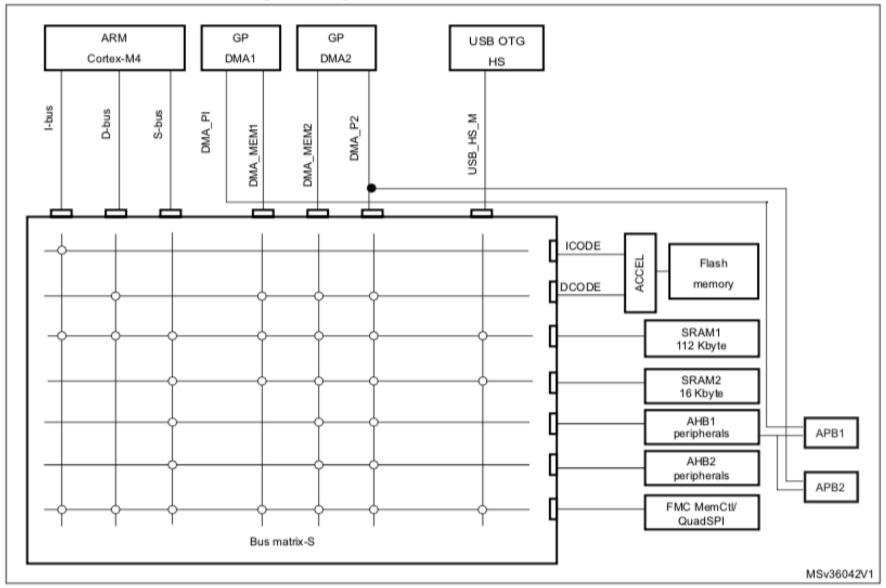
```
( Embedded Systems - Sistemi Embedded - 17873)
( Some code for NUCLEO STM32F446RE )
( Daniele Peri, Università degli Studi di Palermo, 17-18 )

\ GPIO register sets are mapped in memory
\ starting from address $40020000
\ Each GPIO register set spans $0400 bytes
\ See en.DM00135183.pdf page 55

: GPIO{   ( addr -- )  ;
: PORT    ( addr -- addr )  DUP CONSTANT $0400 + ;
: }GPIO   ( addr -- )  DROP ;

$40020000
GPIO{  PORT GPIOA  PORT GPIOB  PORT GPIOC  ( ... ) }GPIO
( More GPIO port could be added... )

: REGS{   0  ;
: REG  CREATE  DUP , OVER + DOES> @ + ;
: }REGS   2DROP ;

$04 REGS{
    REG MODER   REG OTYPER   REG OSPEEDR
    REG PUPDR   REG IDR      REG ODR
    REG BSSR    REG LCKR     REG AFRL
    REG AFRH
}REGS
```

SYSTEM ARCHITECTURE

In STM32F446xx, the main system consists of a **32-bit multilayer AHB bus matrix** that interconnects:

- **Seven masters**:
  - Cortex®-M4 with FPU core I-bus, D-bus and S-bus
  - DMA1 memory bus
  - DMA2 memory bus
  - DMA2 peripheral bus
  - USB OTG HS DMA bus
- **Seven slaves:**
  - Internal Flash memory ICode bus
  - Internal Flash memory DCode bus
  - Main internal SRAM1 (112 KB)
  - Auxiliary internal SRAM2 (16 KB)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals
  - AHB2 peripherals
  - Flexible Memory Controller FMC / QUADSPI

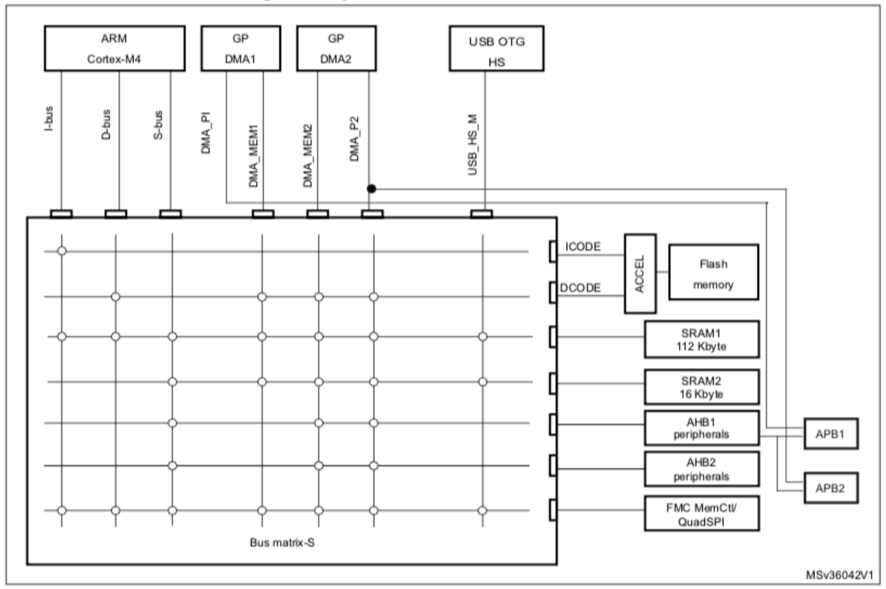
Figure 1. System architecture for STM32F446xx devices



SYSTEM ARCHITECTURE

- **I-bus:** connects the Instruction bus of the Cortex®-M4 with FPU core to the BusMatrix. This bus is used by the core to fetch instructions. The target of this bus is a memory containing code (internal Flash memory/SRAM or external memories through the FMC)
- **D-bus:** connects the databus of the Cortex®-M4 with FPU to the BusMatrix. This bus is used by the core for literal load and debug access. The target of this bus is a memory containing code or data (internal Flash memory or external memories through the FMC).
- **S-bus:** connects the system bus of the Cortex®-M4 with FPU core to a BusMatrix. This bus is used to access data located in a peripheral or in SRAM. Instructions may also be fetch on this bus (less efficient than ICode). The targets of this bus are the internal SRAM, SRAM2, the AHB1 peripherals including the APB peripherals, the AHB2 peripherals and the external memories through the FMC and QUADSPI.
- **DMA memory bus:** connects the DMA memory bus master interface to the BusMatrix. It is used by the DMA to perform transfer to/from memories. The targets of this bus are data memories: internal Flash, internal SRAMs (SRAM1, SRAM2) and external memories through the FMC and QUADSPI.
- **DMA peripheral bus:** connects the DMA peripheral master bus interface to the BusMatrix. This bus is used by the DMA to access AHB peripherals or to perform memory-to-memory transfers. The targets of this bus are the AHB and APB peripherals plus data memories: internal Flash, internal SRAMs (SRAM1, SRAM2) and external memories through the FMC and the QUADSPI.

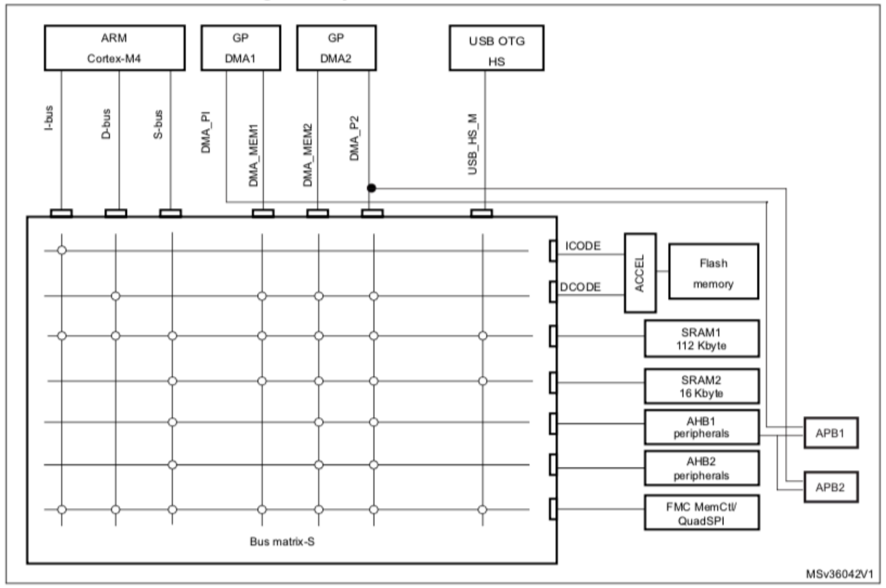
Figure 1. System architecture for STM32F446xx devices



SYSTEM ARCHITECTURE

- **USB OTG HS DMA bus:** connects the USB OTG HS DMA master interface to the BusMatrix. This bus is used by the USB OTG DMA to load/store data to a memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2), internal Flash memory, and external memories through the FMC and QUADSPI
- **BusMatrix:** manages the access arbitration between masters. The arbitration uses a round-robin algorithm
- **AHB/APB bridges (APB):** The two AHB/APB bridges, APB1 and APB2, provide full synchronous connections between the AHB and the two APB buses, allowing flexible selection of the peripheral frequency
  Refer to the device datasheets for more details on APB1 and APB2 maximum frequencies, and to for the address mapping of AHB and APB peripherals.
  After each device reset, all peripheral clocks are disabled (except for the SRAM and Flash memory interface). Before using a peripheral you have to enable its clock in the RCC_AHBxENR or RCC_APBxENR register.
- *Note: When a 16- or an 8-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 16- or 8-bit data to feed the 32-bit vector.*

Figure 1. System architecture for STM32F446xx devices



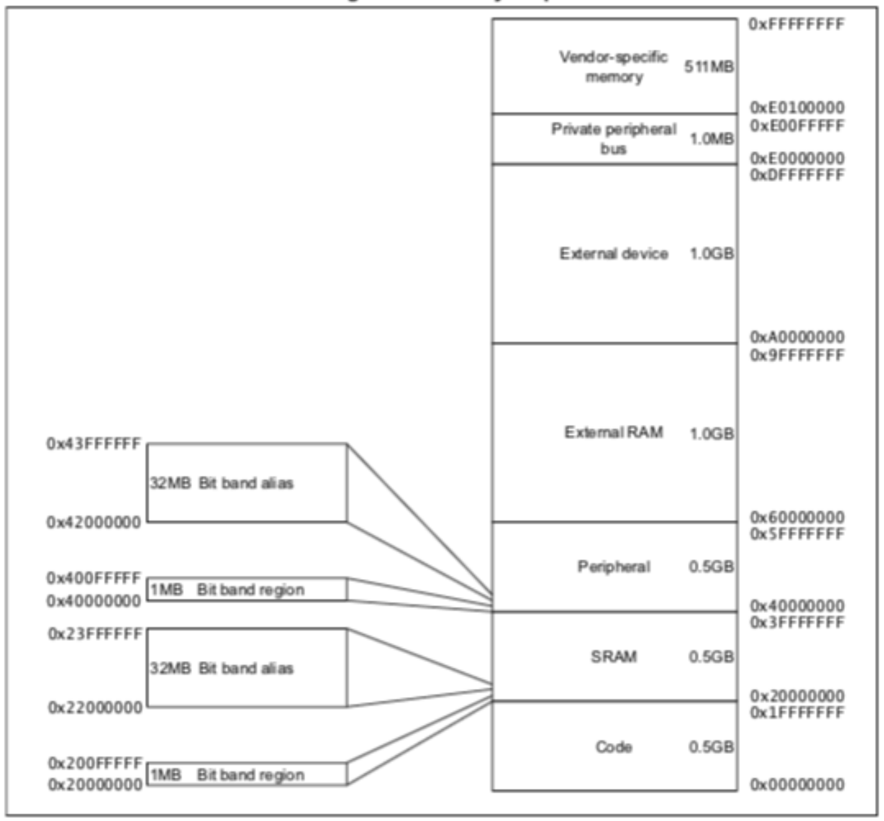
MEMORY ORGANIZATION

- The processor has a fixed memory map that provides up to 4 GB of addressable memory

From the RM0390 Reference manual (p. 54):

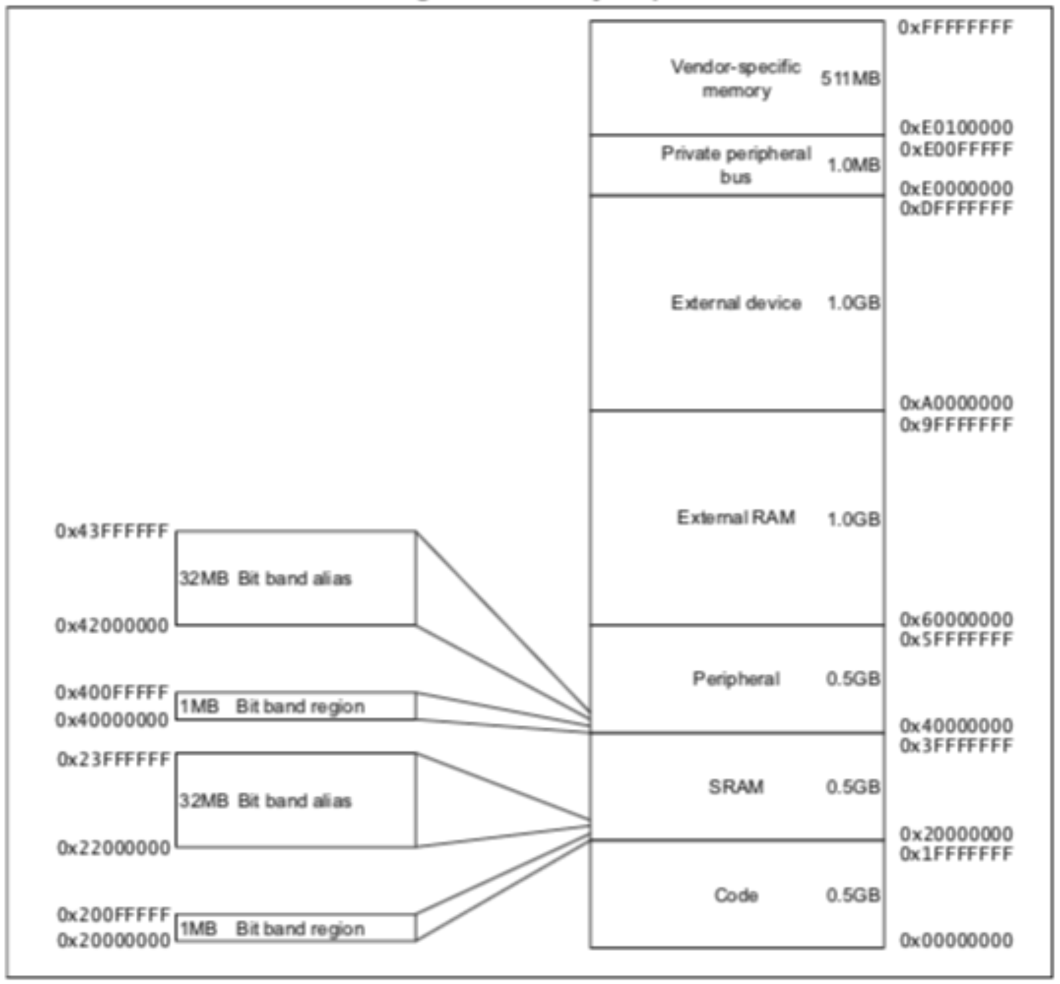
- Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space
- The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant
- The addressable memory space is divided into 8 main blocks, of 512 Mbytes each
- All the memory areas that are not allocated to on-chip memories and peripherals are considered "Reserved"

Figure 8. Memory map



MEMORY ORGANIZATION

Figure 8. Memory map

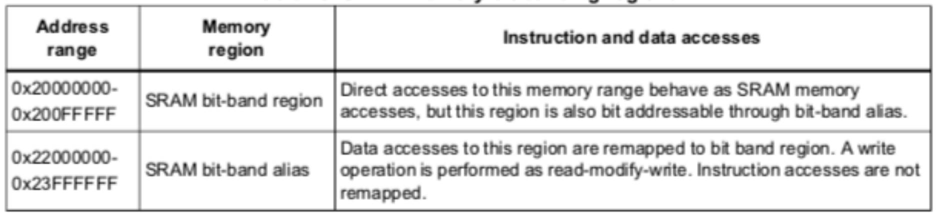


PM0214 Programming manual - (en.DM00046982.pdf), page 27

BIT-BANDING

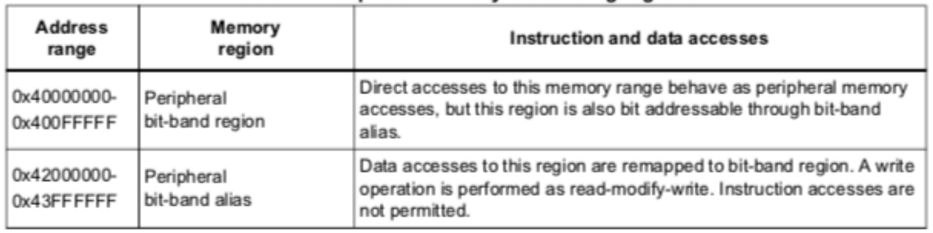
- A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1 Mbyte of the SRAM and peripheral memory regions
- The memory map has two 32 Mbyte alias regions that map to two 1 Mbyte bit-band regions
- Accesses to the 32 Mbyte SRAM alias region map to the 1 Mbyte SRAM bit-band region (Table 13)
- Accesses to the 32 MB peripheral alias region map to the 1 MB peripheral bit-band region (Table 14)

Table 13. SRAM memory bit-banding regions



Address range	Memory region	Instruction and data accesses
0x20000000-0x200FFFFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
0x22000000-0x23FFFFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

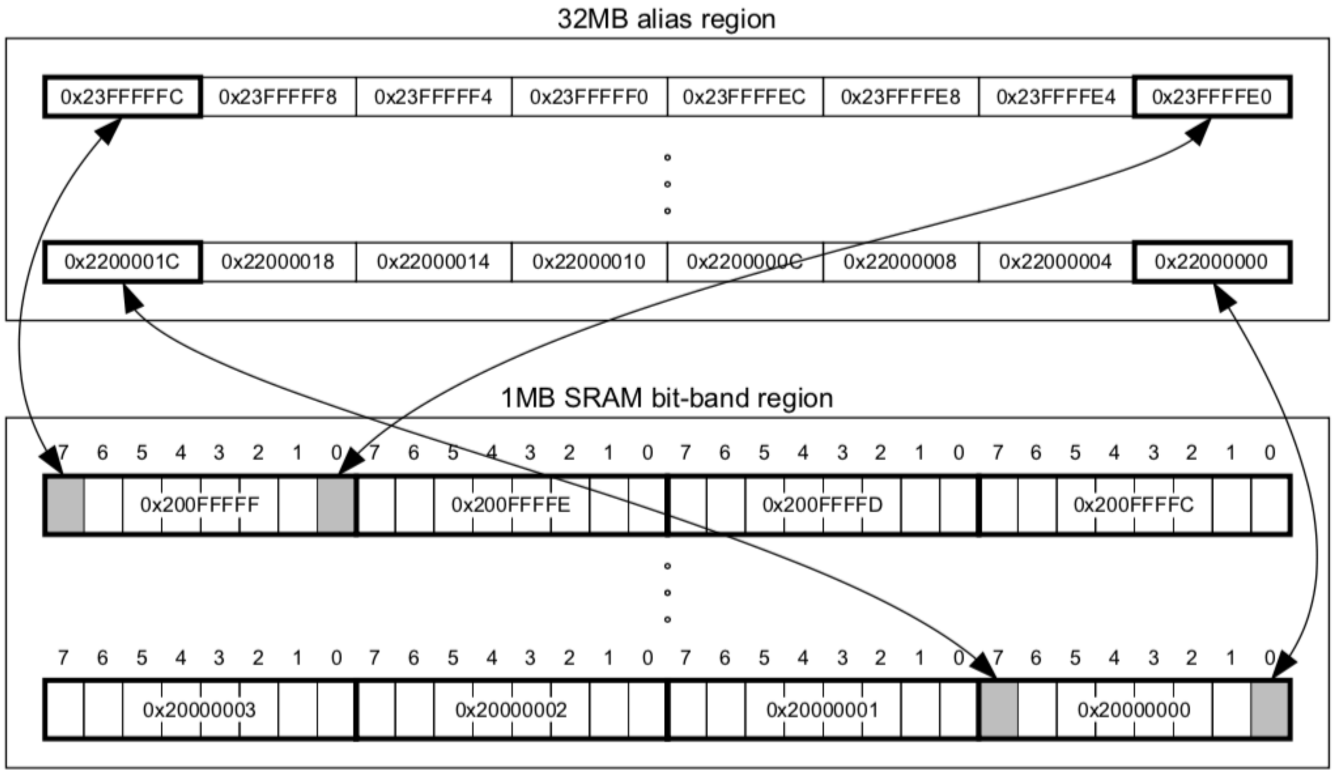
Table 14. Peripheral memory bit-banding regions



Address range	Memory region	Instruction and data accesses
0x40000000-0x400FFFFF	Peripheral bit-band region	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.
0x42000000-0x43FFFFFF	Peripheral bit-band alias	Data accesses to this region are remapped to bit-band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

BIT-BANDING

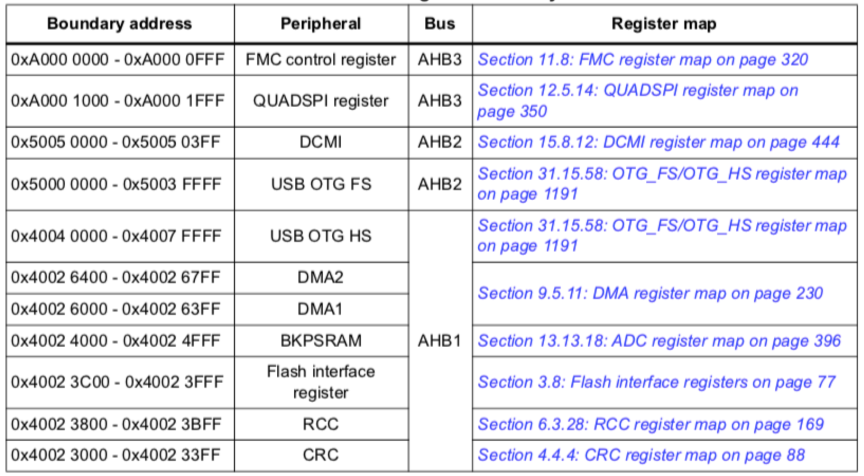
- Accesses to the 32 Mbyte SRAM alias region map to the 1 Mbyte SRAM bit-band region (Table 13)
- Accesses to the 32 MB peripheral alias region map to the 1 MB peripheral bit-band region (Table 14)



MEMORY ORGANIZATION

- Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space
- The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant
- The addressable memory space is divided into 8 main blocks, of 512 Mbytes each.
- All the memory areas that are not allocated to on-chip memories and peripherals are considered "Reserved"

Table 1. STM32F446xx register boundary addresses



Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FMC control register	AHB3	Section 11.8: FMC register map on page 320
0xA000 1000 - 0xA000 1FFF	QUADSPI register	AHB3	Section 12.5.14: QUADSPI register map on page 350
0x5005 0000 - 0x5005 03FF	DCMI	AHB2	Section 15.8.12: DCMI register map on page 444
0x5000 0000 - 0x5003 FFFF	USB OTG FS	AHB2	Section 31.15.58: OTG_FS/OTG_HS register map on page 1191
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1	Section 31.15.58: OTG_FS/OTG_HS register map on page 1191
0x4002 6400 - 0x4002 67FF	DMA2		Section 9.5.11: DMA register map on page 230
0x4002 6000 - 0x4002 63FF	DMA1		
0x4002 4000 - 0x4002 4FFF	BKPSRAM		Section 13.13.18: ADC register map on page 396
0x4002 3C00 - 0x4002 3FFF	Flash interface register		Section 3.8: Flash interface registers on page 77
0x4002 3800 - 0x4002 3BFF	RCC		Section 6.3.28: RCC register map on page 169
0x4002 3000 - 0x4002 33FF	CRC		Section 4.4.4: CRC register map on page 88

MEMORY ORGANIZATION

- Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space
- The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word’s least significant byte and the highest numbered byte the most significant
- The addressable memory space is divided into 8 main blocks, of 512 Mbytes each.
- All the memory areas that are not allocated to on-chip memories and peripherals are considered “Reserved”

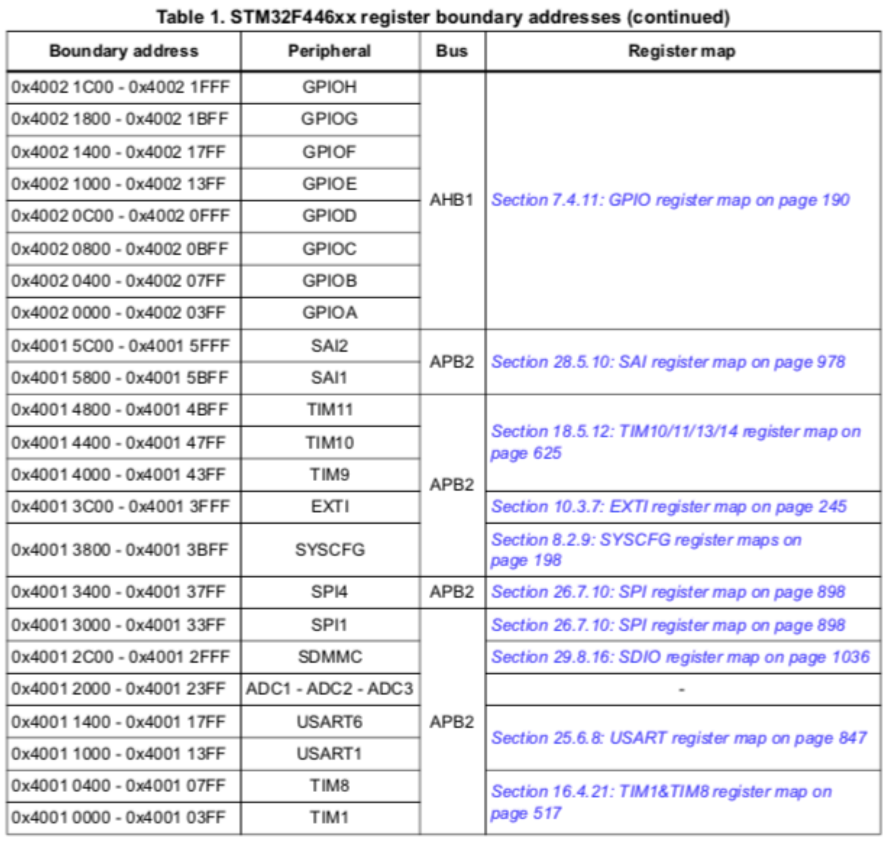


Table 1. STM32F446xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4002 1C00 - 0x4002 1FFF	GPIOH	AHB1	Section 7.4.11: GPIO register map on page 190
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		
0x4001 5C00 - 0x4001 5FFF	SAI2	APB2	Section 28.5.10: SAI register map on page 978
0x4001 5800 - 0x4001 5BFF	SAI1		
0x4001 4800 - 0x4001 4BFF	TIM11	APB2	Section 18.5.12: TIM10/11/13/14 register map on page 625
0x4001 4400 - 0x4001 47FF	TIM10		
0x4001 4000 - 0x4001 43FF	TIM9		
0x4001 3C00 - 0x4001 3FFF	EXTI		Section 10.3.7: EXTI register map on page 245
0x4001 3800 - 0x4001 3BFF	SYSCFG		Section 8.2.9: SYSCFG register maps on page 198
0x4001 3400 - 0x4001 37FF	SPI4	APB2	Section 26.7.10: SPI register map on page 898
0x4001 3000 - 0x4001 33FF	SPI1	APB2	Section 26.7.10: SPI register map on page 898
0x4001 2C00 - 0x4001 2FFF	SDMMC		Section 29.8.16: SDIO register map on page 1036
0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3		-
0x4001 1400 - 0x4001 17FF	USART6		Section 25.6.8: USART register map on page 847
0x4001 1000 - 0x4001 13FF	USART1		
0x4001 0400 - 0x4001 07FF	TIM8		Section 16.4.21: TIM1&TIM8 register map on page 517
0x4001 0000 - 0x4001 03FF	TIM1		

MEMORY ORGANIZATION

Memory organization

The Flash memory (*Embedded Flash memory*) has the following main features:

- Capacity up to 512 KBytes
- 128 bits wide data read
- Byte, half-word, word and double word write
- Sector and mass erase

The Flash memory is organized as follows:

- A main memory block divided into 4 sectors of 16 KBytes, 1 sector of 64 KBytes, and 3 sectors of 128 Kbytes
- System memory from which the device boots in System memory boot mode
- 512 OTP (one-time programmable) bytes for user data
  The OTP area contains 16 additional bytes used to lock the corresponding OTP data block.
- Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode
- Low-power modes

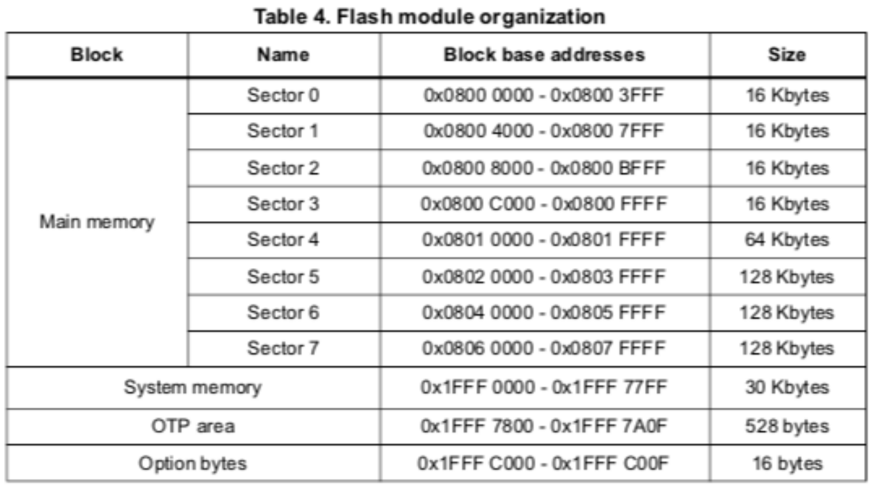
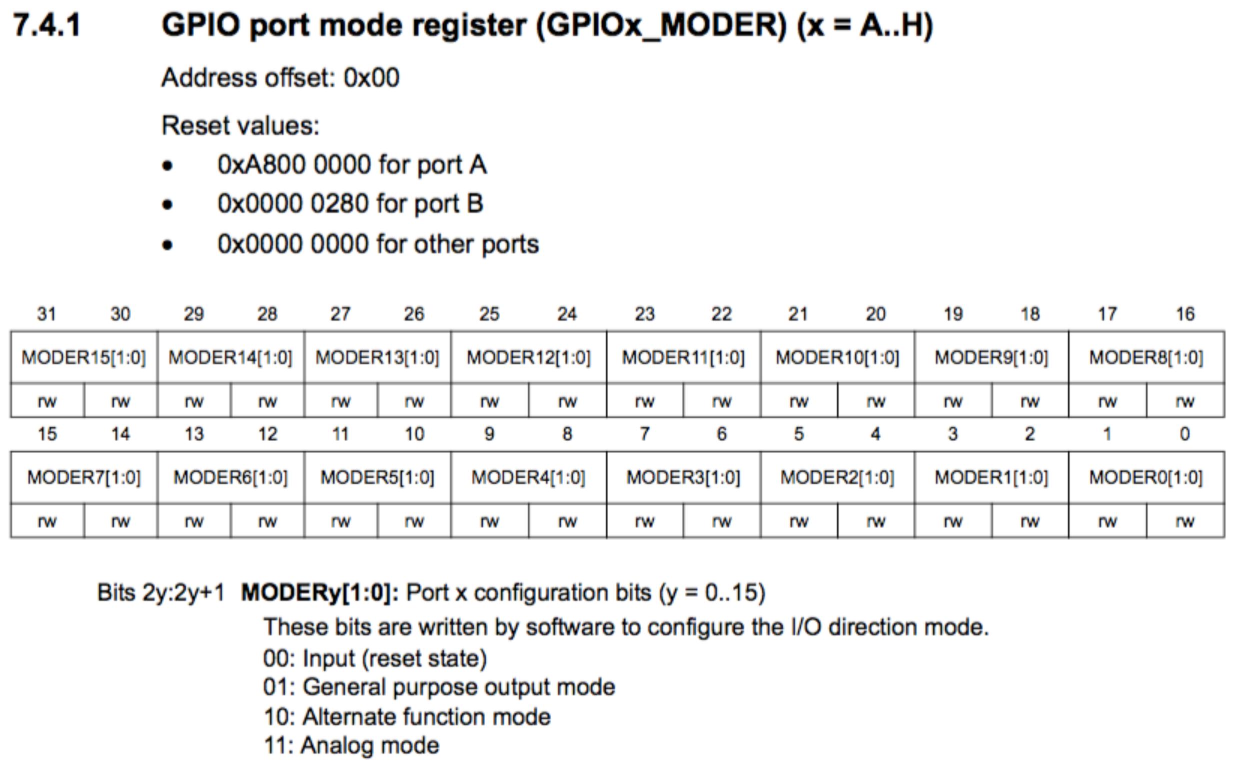


Table 4. Flash module organization

Block	Name	Block base addresses	Size
Main memory	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbytes
	Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbytes
	Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbytes
	Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbytes
	Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbytes
	Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbytes
	Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbytes
	Sector 7	0x0806 0000 - 0x0807 FFFF	128 Kbytes
System memory		0x1FFF 0000 - 0x1FFF 77FF	30 Kbytes
OTP area		0x1FFF 7800 - 0x1FFF 7A0F	528 bytes
Option bytes		0x1FFF C000 - 0x1FFF C00F	16 bytes

GPIO

- The I/O direction of each pin can be configured through a two-bit value in the port MODER register



7.4.1 GPIO port mode register (GPIOx_MODER) (x = A..H)

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

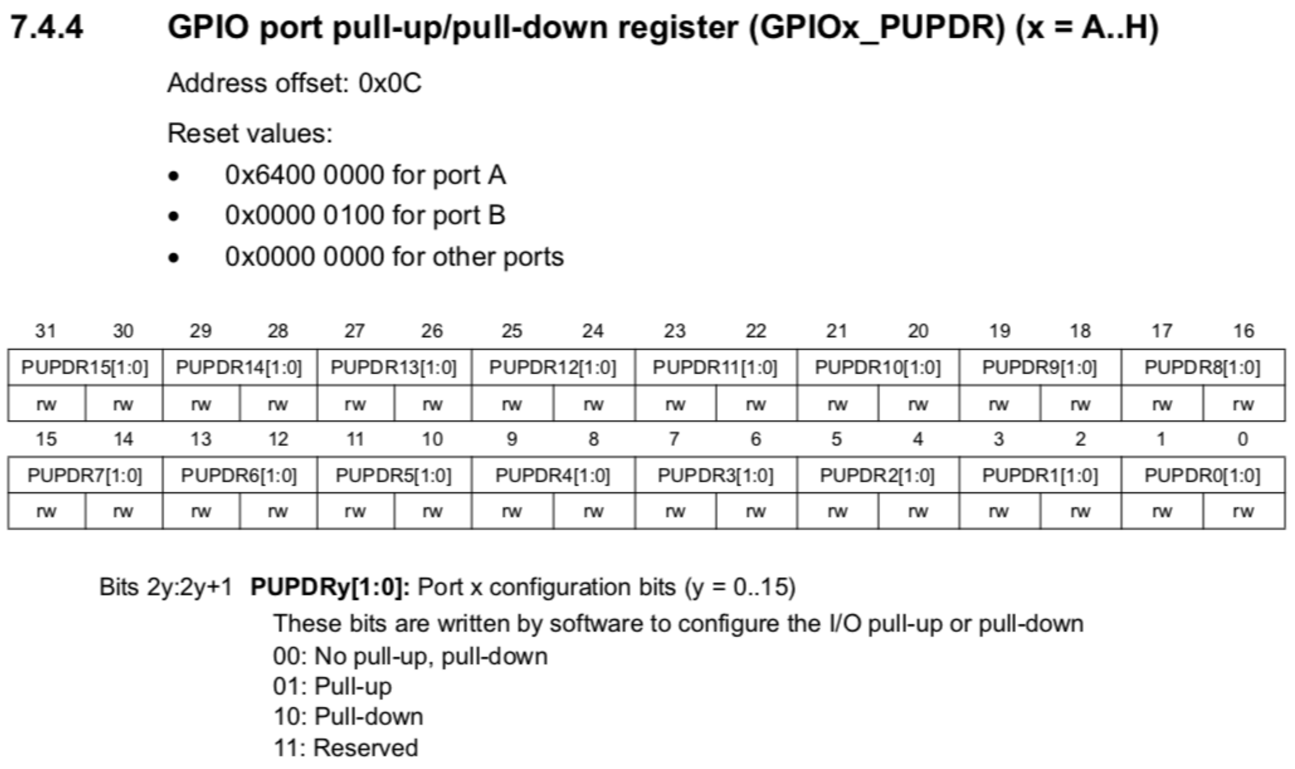
Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

RM0390 Reference manual - (en.DM00135183.pdf), page 184

GPIO

- Electrical connection characteristics of each pin can be configured through a two-bit value in the port PUPDR register



7.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..H)

Address offset: 0x0C

Reset values:

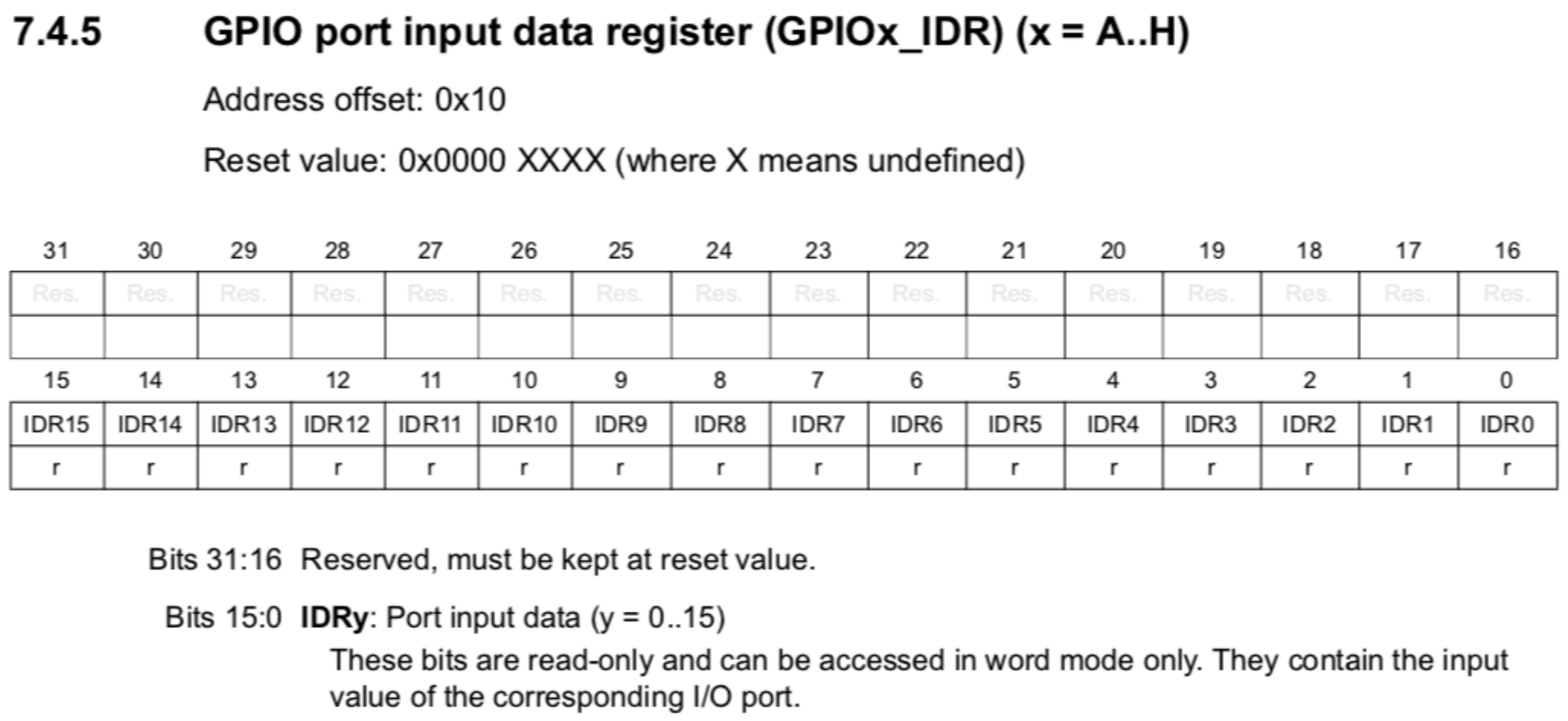
- 0x6400 0000 for port A
- 0x0000 0100 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **PUPDRy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

GPIO

- The IDR register of a port holds the digital input values of its pins
  - The whole word must be read to read even a single pin value



7.4.5 GPIO port input data register (GPIOx_IDR) (x = A..H)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

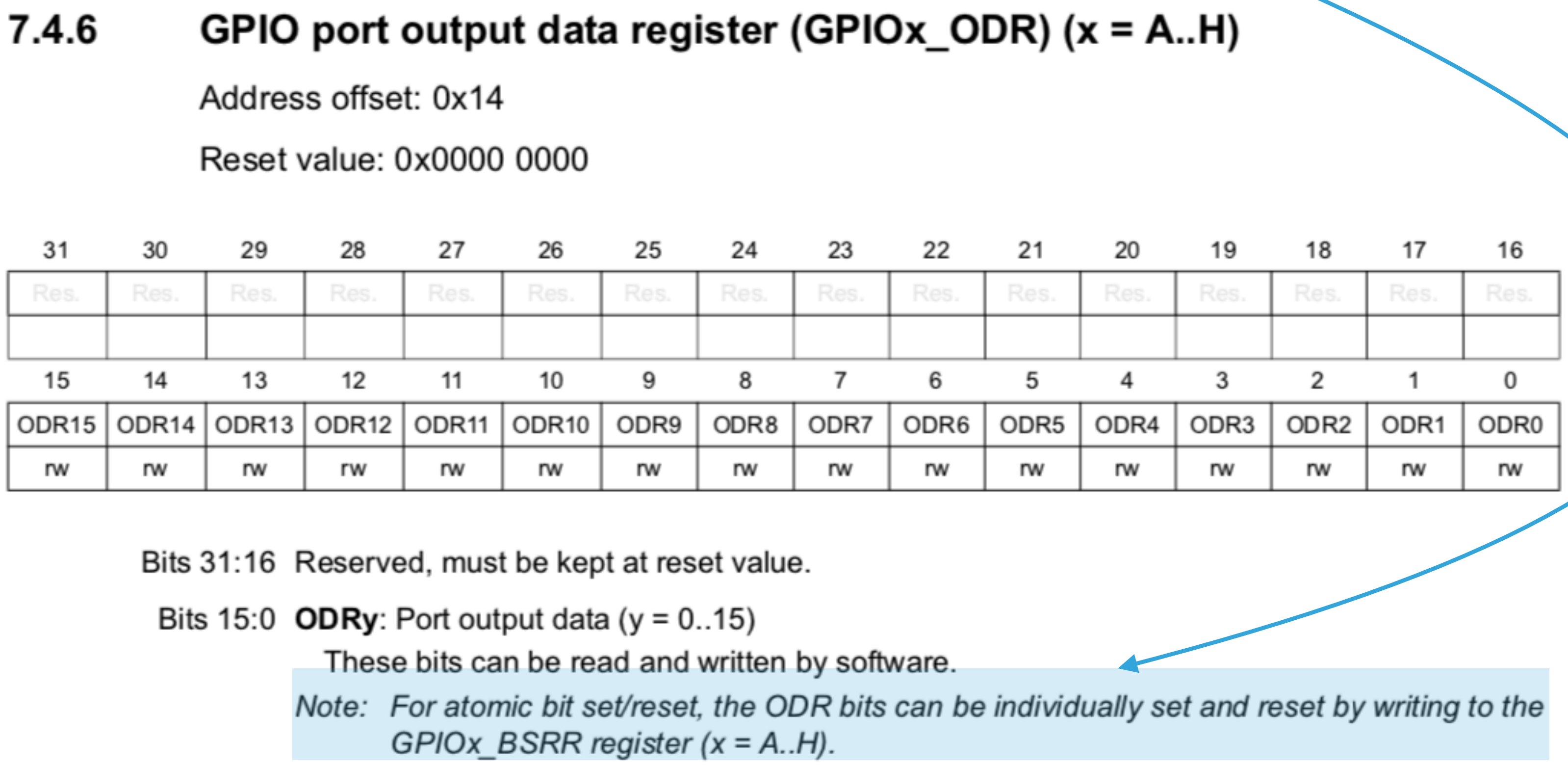
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **IDRy**: Port input data (y = 0..15)
These bits are read-only and can be accessed in word mode only. They contain the input value of the corresponding I/O port.

GPIO

- The ODR register of a port gives read and write access to the digital output values of its pins
  - The whole word must be accessed to change the value of even a single pin value



7.4.6 GPIO port output data register (GPIOx_ODR) (x = A..H)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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.

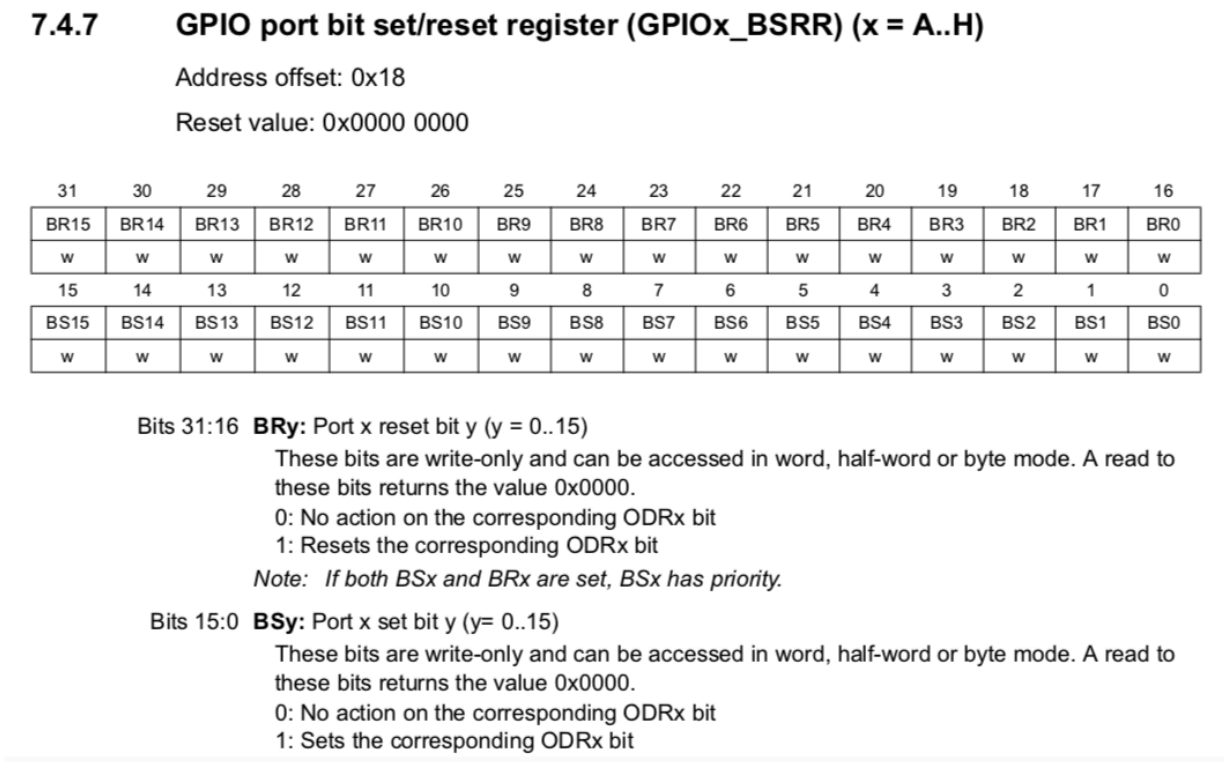
Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..H).

RM0390 Reference manual - (en.DM00135183.pdf), page 187

GPIO

Output pin Bit Set/Reset

- Registers GPIOx_BSRR allows for controlling digital outputs without bit mask operations; these registers work similarly to the GPSETn/GPCLRn registers in the Raspberry Pi SOCs



7.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..H)

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

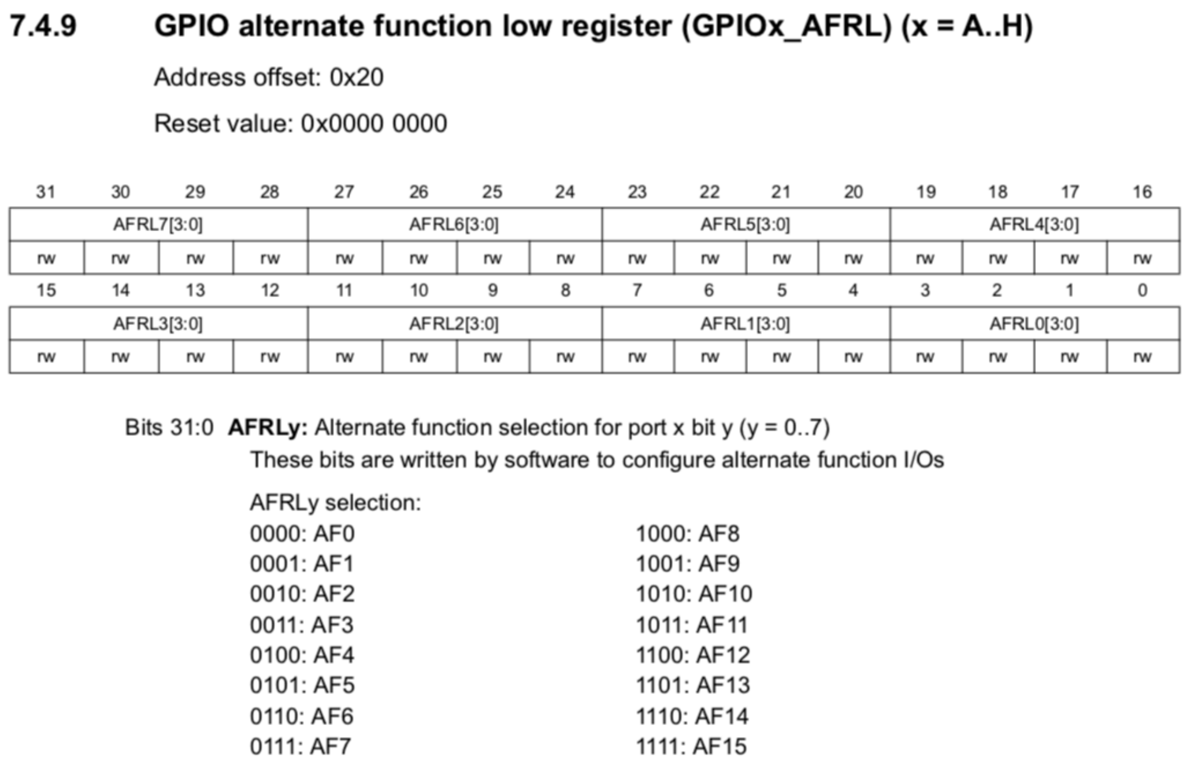
1: Sets the corresponding ODRx bit

RM0390 Reference manual - (en.DM00135183.pdf) page 187

GPIO

GPIO and Alternate Functions

- When in Alternate Function mode, a GPIO pin can be configured through either the **<u>AFRL (lower 8 pins)</u>** or AFRH (highest 8 pins) register to express a specific function.



7.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A..H)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
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
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRLy:** Alternate function selection for port x bit y (y = 0..7)
These bits are written by software to configure alternate function I/Os

AFRLy selection:

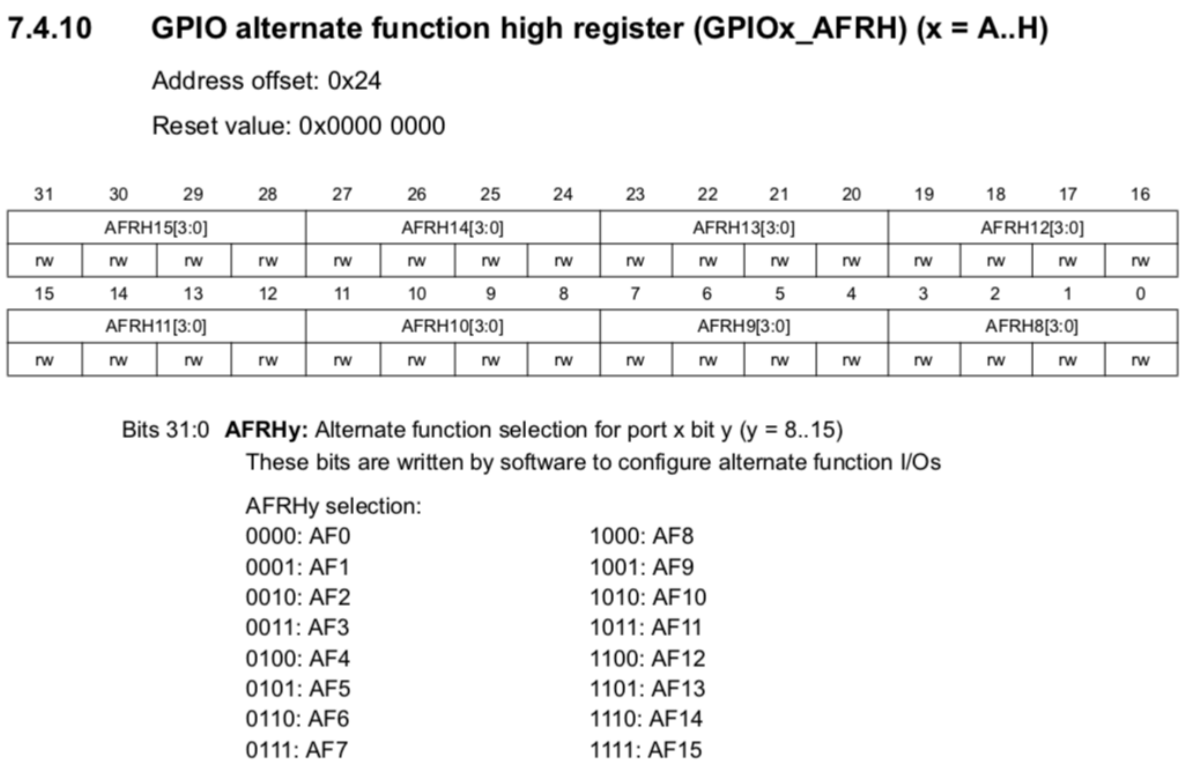
0000: AF0	1000: AF8
0001: AF1	1001: AF9
0010: AF2	1010: AF10
0011: AF3	1011: AF11
0100: AF4	1100: AF12
0101: AF5	1101: AF13
0110: AF6	1110: AF14
0111: AF7	1111: AF15

RM0390 Reference manual - (en.DM00135183.pdf) page 189

GPIO

GPIO and Alternate Functions

- When in Alternate Function mode, a GPIO pin can be configured through either the AFRL (lower 8 pins) or **<u>AFRH (highest 8 pins)</u>** register to express a specific function.



7.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A..H)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
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
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRHy:** Alternate function selection for port x bit y (y = 8..15)

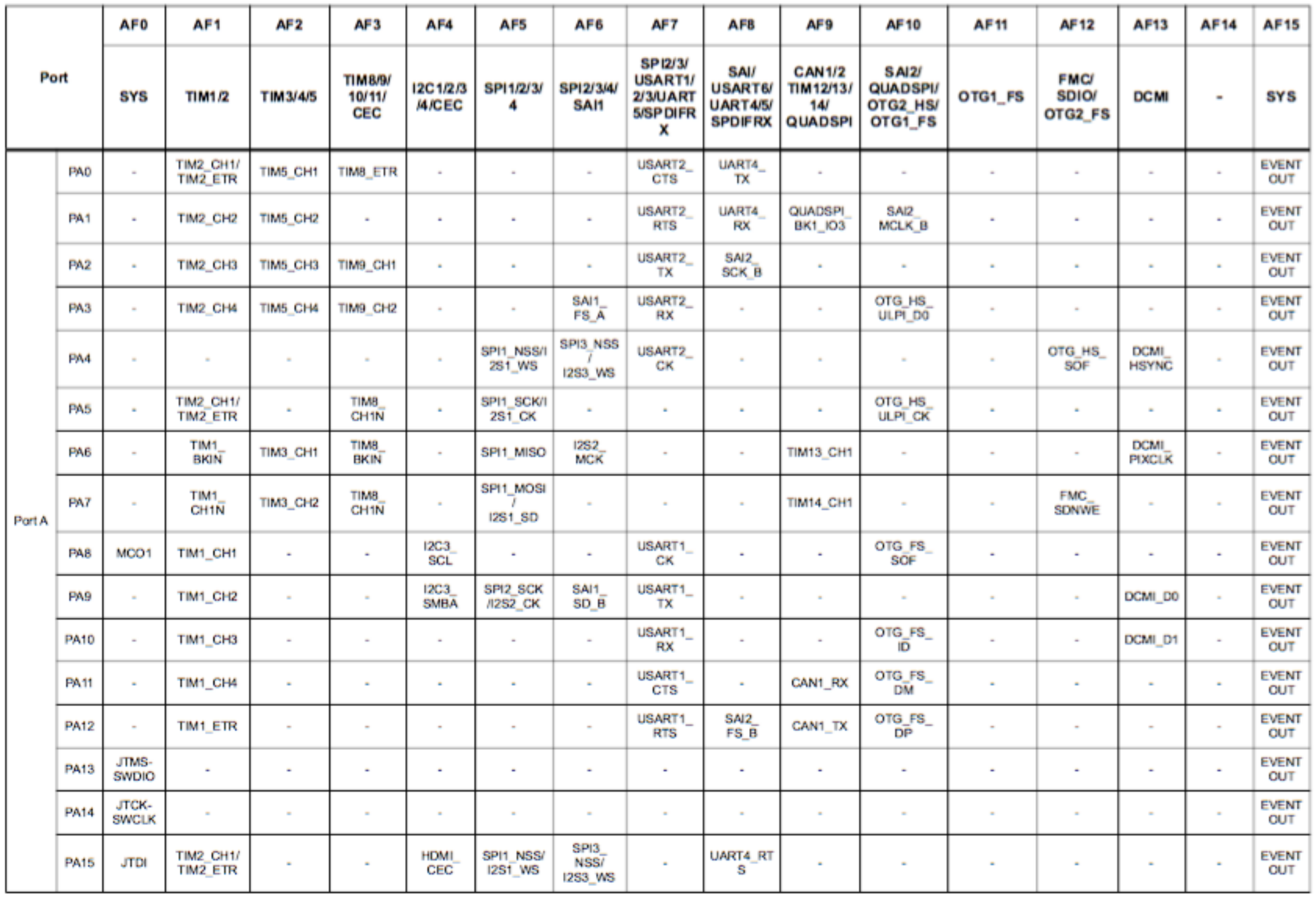
These bits are written by software to configure alternate function I/Os

AFRHy selection:

0000: AF0	1000: AF8
0001: AF1	1001: AF9
0010: AF2	1010: AF10
0011: AF3	1011: AF11
0100: AF4	1100: AF12
0101: AF5	1101: AF13
0110: AF6	1110: AF14
0111: AF7	1111: AF15

GPIO

Table 11. Alternate function

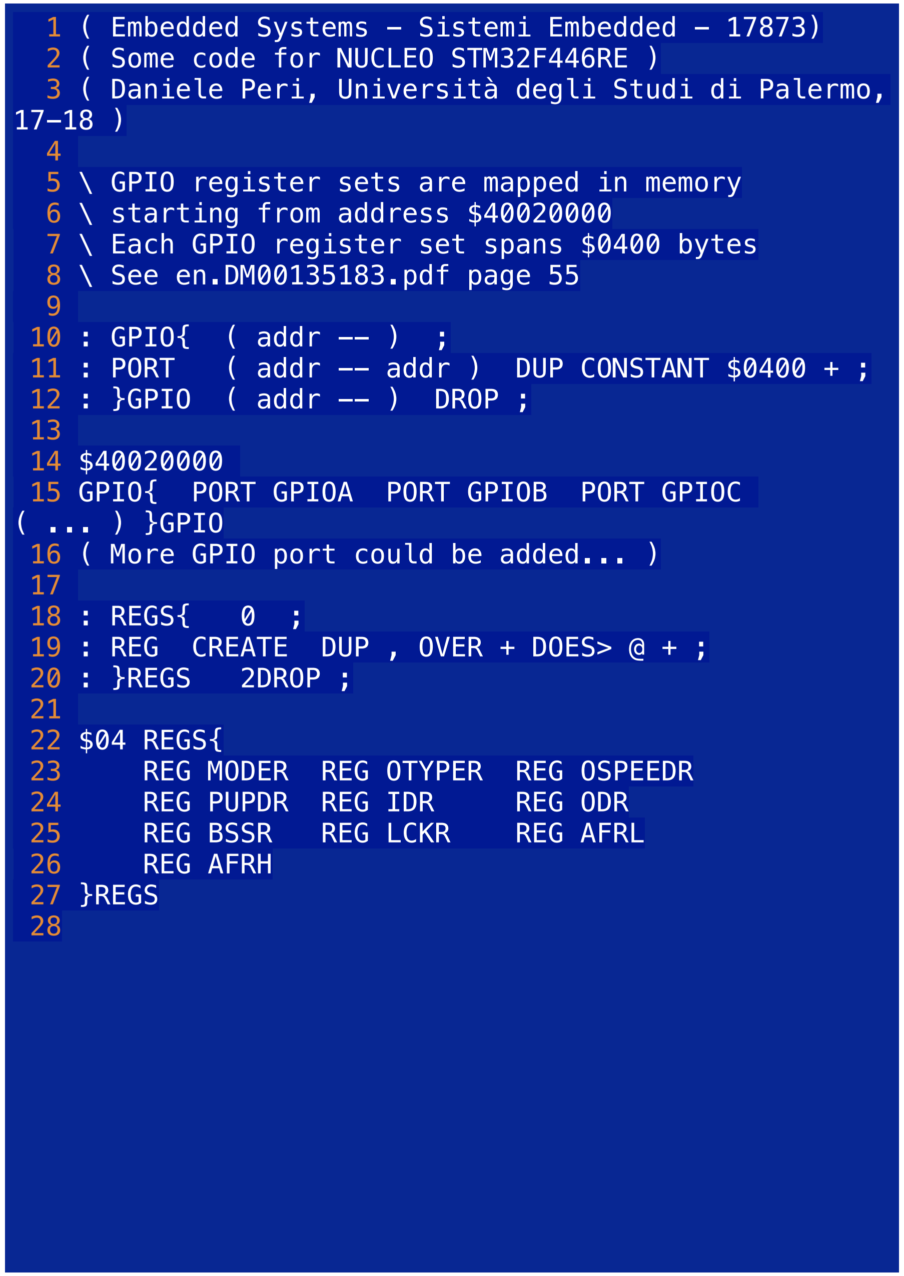


Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
		SYS	TIM1/2	TIM3/4/5	TIM8/9/ 10/11/ CEC	I2C1/2/3 /4/CEC	SPI1/2/3/ 4	SPI2/3/4/ SAI1	SPI2/3/ USART1/ 2/3/UART 5/SPDIFR X	SAI/ USART6/ UART4/5/ SPDIFRX	CAN1/2 TIM12/13/ 14/ QUADSPI	SAI2/ QUADSPI/ OTG2_HS/ OTG1_FS	OTG1_FS	FMC/ SDIO/ OTG2_FS	DCMI	-	SYS
Port A	PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	TIM8_ETR	-	-	-	USART2_ CTS	UART4_ TX	-	-	-	-	-	-	EVENT OUT
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_ RTS	UART4_ RX	QUADSPI_ BK1_IO3	SAI2_ MCLK_B	-	-	-	-	EVENT OUT
	PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	-	-	USART2_ TX	SAI2_ SCK_B	-	-	-	-	-	-	EVENT OUT
	PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	-	SAI1_ FS_A	USART2_ RX	-	-	OTG_HS_ ULPI_D0	-	-	-	-	EVENT OUT
	PA4	-	-	-	-	-	SPI1_NSS/I 2S1_WS	SPI3_NSS / I2S3_WS	USART2_ CK	-	-	-	-	OTG_HS_ SOF	DCMI_ HSYNC	-	EVENT OUT
	PA5	-	TIM2_CH1/ TIM2_ETR	-	TIM8_ CH1N	-	SPI1_SCK/I 2S1_CK	-	-	-	-	OTG_HS_ ULPI_CK	-	-	-	-	EVENT OUT
	PA6	-	TIM1_ BKIN	TIM3_CH1	TIM8_ BKIN	-	SPI1_MISO	I2S2_ MCK	-	-	TIM13_CH1	-	-	-	DCMI_ PIXCLK	-	EVENT OUT
	PA7	-	TIM1_ CH1N	TIM3_CH2	TIM8_ CH1N	-	SPI1_MOSI / I2S1_SD	-	-	-	TIM14_CH1	-	-	FMC_ SDNWE	-	-	EVENT OUT
	PA8	MCO1	TIM1_CH1	-	-	I2C3_ SCL	-	-	USART1_ CK	-	-	OTG_FS_ SOF	-	-	-	-	EVENT OUT
	PA9	-	TIM1_CH2	-	-	I2C3_ SMBA	SPI2_SCK /I2S2_CK	SAI1_ SD_B	USART1_ TX	-	-	-	-	-	DCMI_D0	-	EVENT OUT
	PA10	-	TIM1_CH3	-	-	-	-	-	USART1_ RX	-	-	OTG_FS_ ID	-	-	DCMI_D1	-	EVENT OUT
	PA11	-	TIM1_CH4	-	-	-	-	-	USART1_ CTS	-	CAN1_RX	OTG_FS_ DM	-	-	-	-	EVENT OUT
	PA12	-	TIM1_ETR	-	-	-	-	-	USART1_ RTS	SAI2_ FS_B	CAN1_TX	OTG_FS_ DP	-	-	-	-	EVENT OUT
	PA13	JTMS- SWDIO	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA14	JTCK- SWCLK	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA15	JTDI	TIM2_CH1/ TIM2_ETR	-	-	HDMI_ CEC	SPI1_NSS/ I2S1_WS	SPI3_ NSS/ I2S3_WS	-	UART4_RT S	-	-	-	-	-	-	EVENT OUT

STM32F446xC/E Datasheet - (en.DM00141306.pdf) page 59

HARDWARE ABSTRACTION LAYERS

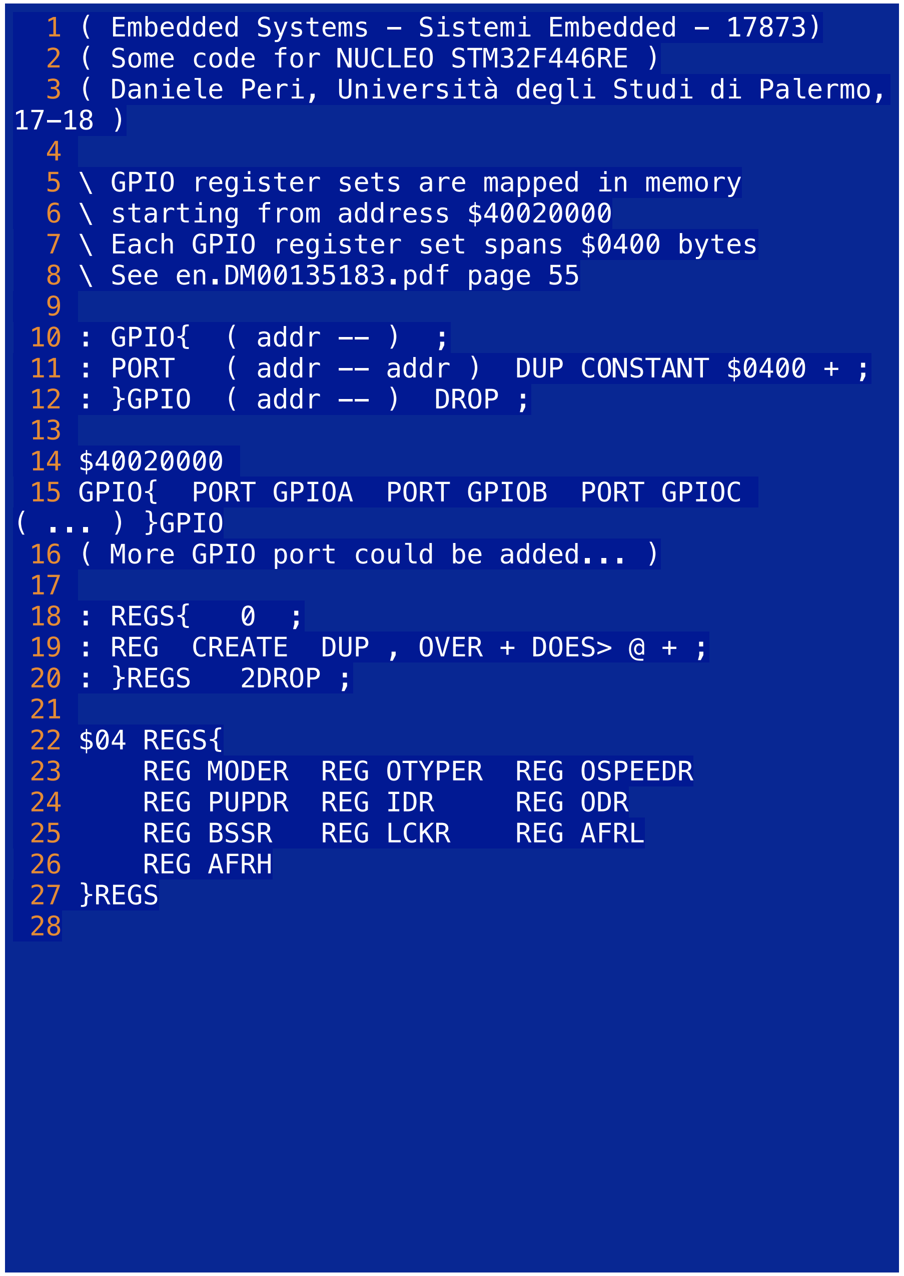
- Often information is spread across several documents
- HAL definitions can be made clear to collect consolidated knowledge about devices in a executable and testable way
- FORTH is extremely useful to interactively test what data sheet report and to define hardware driving code easily



```
( Embedded Systems - Sistemi Embedded - 17873)
( Some code for NUCLEO STM32F446RE )
( Daniele Peri, Università degli Studi di Palermo, 17-18 )

\ GPIO register sets are mapped in memory
\ starting from address $40020000
\ Each GPIO register set spans $0400 bytes
\ See en.DM00135183.pdf page 55

: GPIO{   ( addr -- )  ;
: PORT    ( addr -- addr )  DUP CONSTANT $0400 + ;
: }GPIO   ( addr -- )  DROP ;

$40020000
GPIO{  PORT GPIOA  PORT GPIOB  PORT GPIOC ( ... ) }GPIO
( More GPIO port could be added... )

: REGS{   0  ;
: REG  CREATE  DUP , OVER + DOES> @ + ;
: }REGS   2DROP ;

$04 REGS{
    REG MODER   REG OTYPER   REG OSPEEDR
    REG PUPDR   REG IDR      REG ODR
    REG BSSR    REG LCKR     REG AFRL
    REG AFRH
}REGS
```

HARDWARE ABSTRACTION LAYERS

- Let’s define a HAL in FORTH using Mecrisp-Stellaris on the Nucleo-64 STM32F446
  - The code begins with GPIO definitions
  - Then GPIO words to interact with the on-board I/O devices (LD2 LED, B1 Button) are defined
  - The HAL can also be used to drive external I/O devices



```
( Embedded Systems - Sistemi Embedded - 17873)
( Some code for NUCLEO STM32F446RE )
( Daniele Peri, Università degli Studi di Palermo, 17-18 )

\ GPIO register sets are mapped in memory
\ starting from address $40020000
\ Each GPIO register set spans $0400 bytes
\ See en.DM00135183.pdf page 55

: GPIO{  ( addr -- )  ;
: PORT   ( addr -- addr )  DUP CONSTANT $0400 + ;
: }GPIO  ( addr -- )  DROP ;

$40020000
GPIO{  PORT GPIOA  PORT GPIOB  PORT GPIOC  ( ... ) }GPIO
( More GPIO port could be added... )

: REGS{   0  ;
: REG  CREATE  DUP , OVER + DOES> @ + ;
: }REGS   2DROP ;

$04 REGS{
    REG MODER  REG OTYPER  REG OSPEEDR
    REG PUPDR  REG IDR     REG ODR
    REG BSSR   REG LCKR    REG AFRL
    REG AFRH
}REGS

```

HARDWARE ABSTRACTION LAYERS

```
( Embedded Systems - Sistemi Embedded - 17873)
( Some code for NUCLEO STM32F446RE )
( Daniele Peri, Università degli Studi di Palermo, 17-18 )

\ GPIO register sets are mapped in memory
\ starting from address $40020000
\ Each GPIO register set spans $0400 bytes
\ See en.DM00135183.pdf page 55

: GPIO{  ( addr -- )  ;
: PORT   ( addr -- addr )  DUP CONSTANT $0400 + ;
: }GPIO  ( addr -- )  DROP ;

$40020000
GPIO{  PORT GPIOA  PORT GPIOB  PORT GPIOC ( ... ) }GPIO
( More GPIO port definitions could be added... )

: REGS{   0  ;
: REG  CREATE  DUP , OVER + DOES> @ + ;
: }REGS   2DROP ;

$04 REGS{
    REG MODER  REG OTYPER  REG OSPEEDR
    REG PUPDR  REG IDR     REG ODR
    REG BSSR   REG LCKR    REG AFRL
    REG AFRH
}REGS

```

Nucleo64-STM32F446RE-simple-HAL.f

HARDWARE ABSTRACTION LAYERS

```
: DELAY   1000 * 0 DO LOOP ;
: 1BIT   $1 1 ;
: 2BIT   $3 2 ;
: 4BIT   $F 4 ;
: MASK  ( index mask width -- offset_mask ) ROT * LSHIFT ;
: PIN   SWAP 1BIT MASK SWAP ;
: MODE   OVER 2BIT MASK SWAP ;
: MODE@   MODE @ AND SWAP 2 * RSHIFT ;
: MODE!   >R R@ MODE @ SWAP NOT AND ROT 3 AND ROT 2 * LSHIFT OR R> ! ;
: BIT  ( mask addr -- addr value mask )  DUP @ ROT ;
: SET  ( addr value mask -- )  OR SWAP ! ;
: CLEAR  ( addr value mask -- )  NOT AND SWAP ! ;
: TRUTH  ( addr value mask -- value )  AND 0<> NIP ;
: OUT@   ODR PIN BIT TRUTH ;
: OUT!   ODR PIN BIT SWAP OVER NOT AND >R ROT AND R> OR SWAP ! ;

\ Examples:
\
\ 5 GPIOA MODE@ .
\ Shows current mode for pin 5 of GPIO Port A (controlling LED2)
\
\ 13 GPIOC MODE@ .
\ Shows current mode for pin 13 of GPIO Port C (connected to Button B1 USER)
\
\ 1 5 GPIOA MODE!
\ Sets Output mode (1) for pin 5 of GPIO Port A (LED2)
\
```

Nucleo64-STM32F446RE-simple-HAL.f

HARDWARE ABSTRACTION LAYERS

```
\ 5 GPIOA ODR PIN BIT SET
\ Sets pin 5 of GPIO Port A to high (LED2 turns ON)
\
\ TRUE 5 GPIOA OUT!
\ Sets pin 5 of GPIO Port A to high (LED2 turns ON)
\

: BUTTON   $2000 GPIOC IDR ; \ Same as 13 GPIOC IDR PIN
: RELEASED  BIT TRUTH ;
: PRESSED   RELEASED NOT ;
: ?BUTTON   BUTTON PRESSED ;
: CLICKED   BEGIN 2DUP PRESSED UNTIL  BEGIN 2DUP RELEASED UNTIL 2DROP ;
: LED   $20 GPIOA ODR ;       \ Same as 5 GPIOA ODR PIN
: ON    ( mask addr -- ) BIT SET ;
: OFF   ( mask addr -- ) BIT CLEAR ;
: BLINK   2DUP  OFF  1000 DELAY  2DUP ON  1000 DELAY OFF ;

\ Examples:
\
\ LED ON
\ LED BLINK
\

( Test code: wait for button click then blink the LED, do it n times )
: TEST  ( n -- ) 0 DO  BUTTON CLICKED  LED BLINK  LOOP ;

\ Configuration:
1 5 GPIOA MODE!
```

Nucleo64-STM32F446RE-simple-HAL.f

ARDUINO CONNECTIVITY

- Connectors CN5, CN6, CN8 and CN9 provide hardware compatibility with Arduino shields and applications

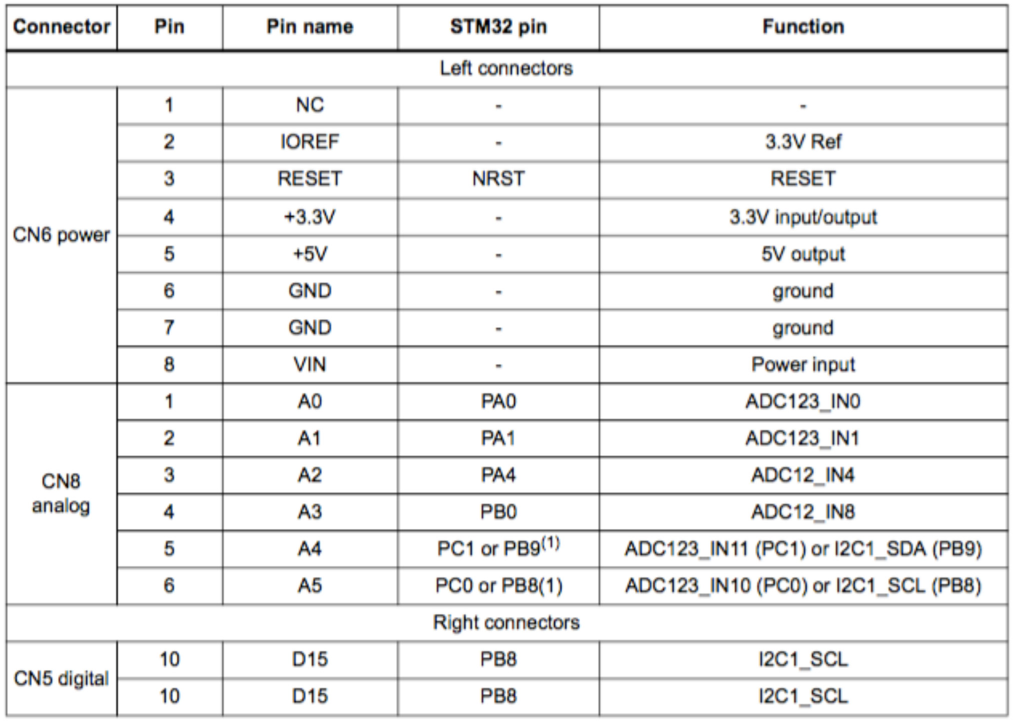


Table 19. Arduino connectors on NUCLEO-F446RE

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC123_IN0
	2	A1	PA1	ADC123_IN1
	3	A2	PA4	ADC12_IN4
	4	A3	PB0	ADC12_IN8
	5	A4	PC1 or PB9(1)	ADC123_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC123_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	10	D15	PB8	I2C1_SCL

ARDUINO CONNECTIVITY

- Connectors CN5, CN6, CN8 and CN9 provide hardware compatibility with Arduino shields and applications

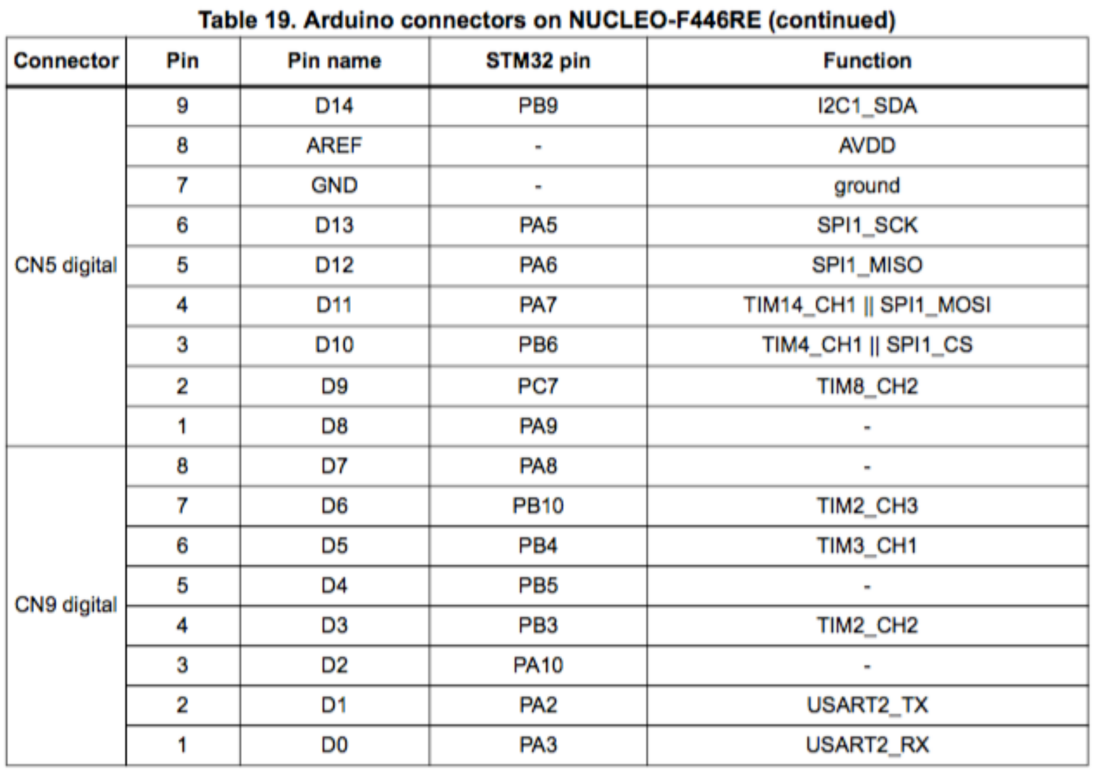


Table 19. Arduino connectors on NUCLEO-F446RE (continued)

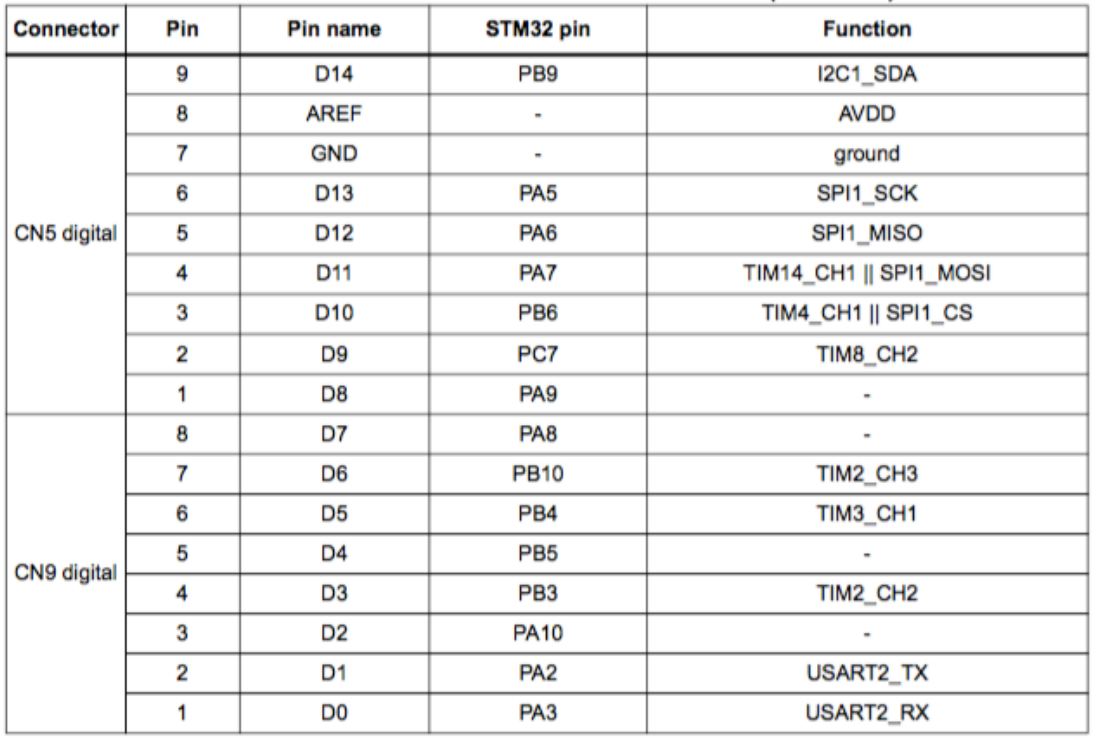
Connector	Pin	Pin name	STM32 pin	Function
CN5 digital	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM14_CH1 || SPI1_MOSI
	3	D10	PB6	TIM4_CH1 || SPI1_CS
	2	D9	PC7	TIM8_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

UM1724 User manual - (en.DM00105823.pdf) - page 48

ARDUINO CONNECTIVITY

- Connectors CN5, CN6, CN8 and CN9 provide hardware compatibility with Arduino shields and applications

Table 19. Arduino connectors on NUCLEO-F446RE (continued)



Connector	Pin	Pin name	STM32 pin	Function
CN5 digital	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM14_CH1 || SPI1_MOSI
	3	D10	PB6	TIM4_CH1 || SPI1_CS
	2	D9	PC7	TIM8_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

UM1724 User manual - (en.DM00105823.pdf) - page 48

ARDUINO CONNECTIVITY

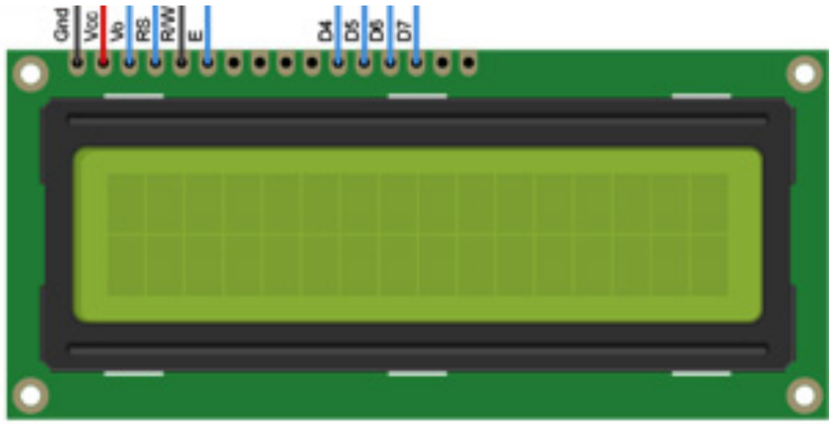
- It is easy to define in FORTH a simple software layer to use the same symbols as found in the specifications of the Arduino application

```
( Embedded Systems - Sistemi Embedded - 17873)
( Some code for Nucleo STM32F446RE )
( Daniele Peri, Università degli Studi di Palermo, 17-18 )

\ Must be INCLUDEd after Nucleo64-STM32F446RE-simple-HAL.f

\ Definitions for a few Arduino pins on
\ connectors CN5 and CN9:
\

: D2   10 GPIOA ;
: D3    3 GPIOB ;
: D4    5 GPIOB ;
: D5    4 GPIOB ;
: D6   10 GPIOB ;
: D7    8 GPIOA ;
: D11   7 GPIOA ;
: D12   6 GPIOA ;
: D13   5 GPIOA ;

\ Examples:
\
\ TRUE D13 OUT!
\ Sets Arduino D13 pin (connector CN5) high
\ This is another way to turn the LD2 LED on.
\
```

arduino.f

GPIO AND BUSES

- The LCD 1602 is a display module
  - 16x2 dot-matrix (5x8 pixels) characters
- It is controlled through a parallel interface with
  - 8-bit/4-bit data bus
  - 3 control signals
- A parallel bus can be set up using a group of GPIOs

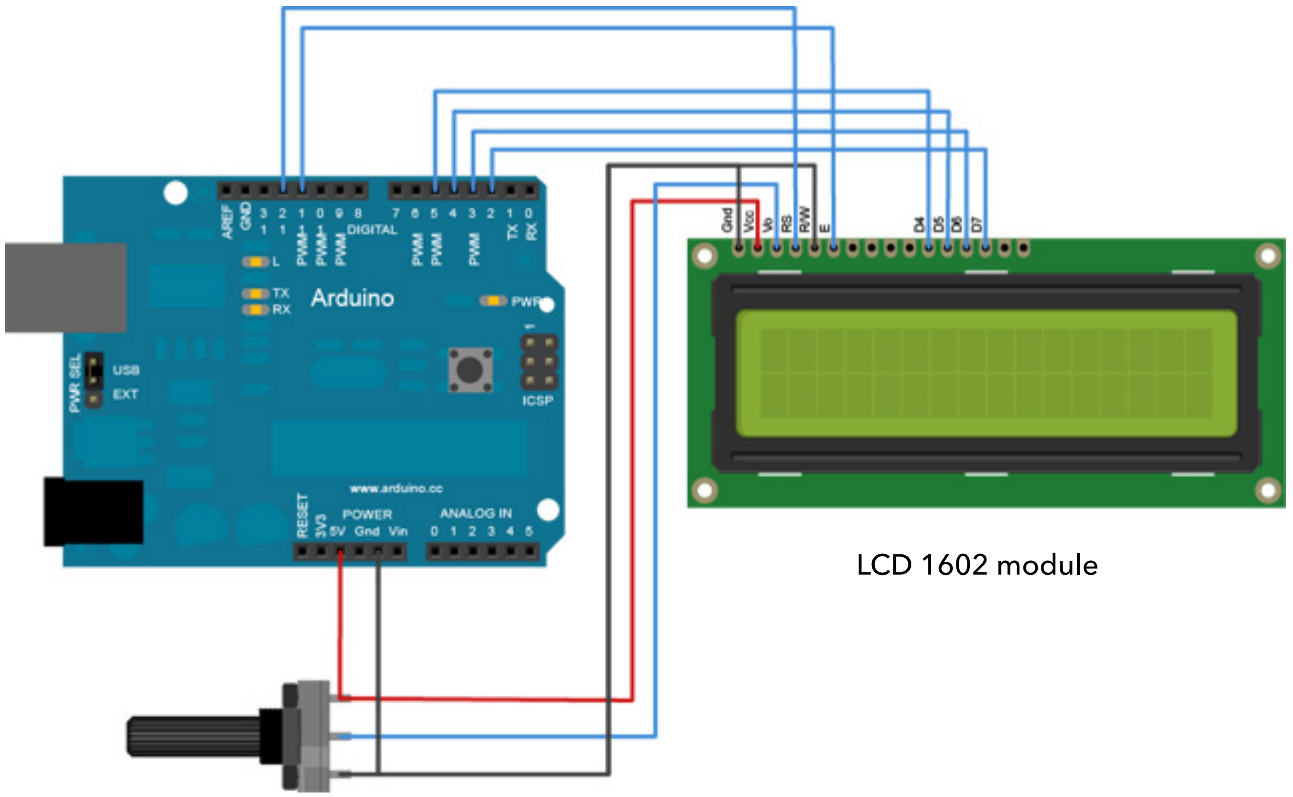


LCD 1602 module

LCD 1602 Arduino Application (from the web)

GPIO AND BUSES

- In a typical Arduino application using the LCD 1602 module
  - Six GPIO pins are used to control the module (write is constantly asserted)

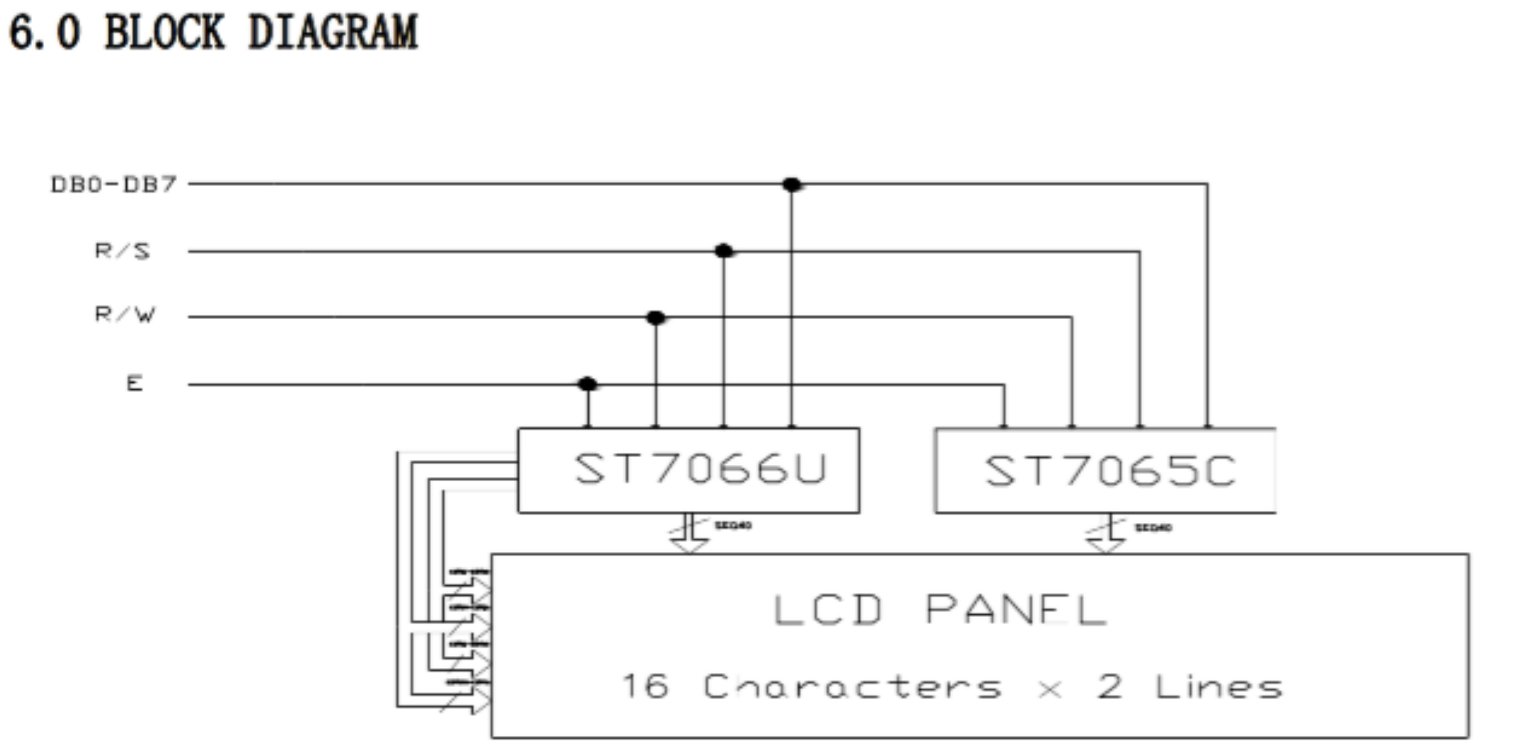


LCD 1602 module

LCD 1602 Arduino Application (from the web)

GPIO AND BUSES

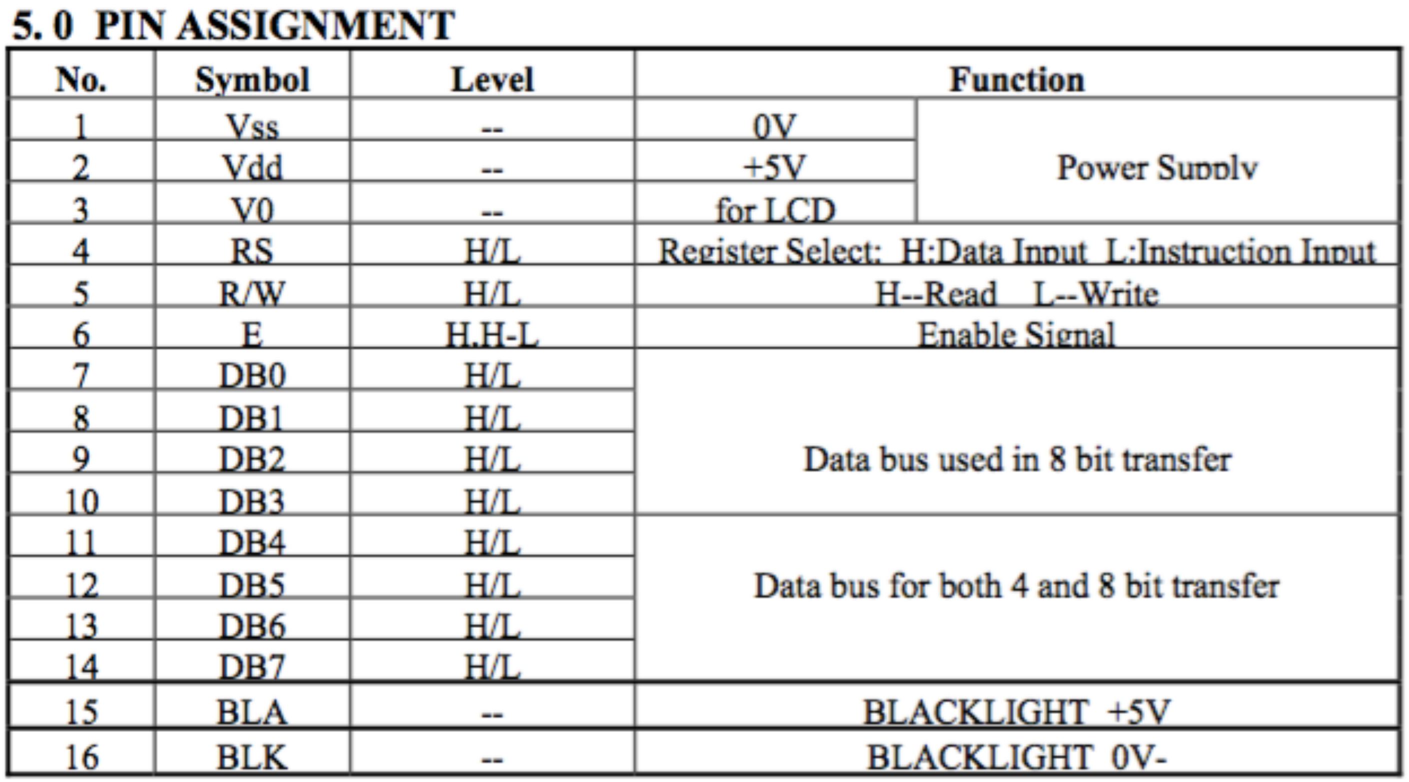
- LCD 1602 schematics from the datasheet
  - The interface signals reach the two controller chips that drive the LCD Panel



LCD Module 1602A-1 specs - (eone-1602a1.pdf)

GPIO AND BUSES

- LCD 1602 schematics from the datasheet



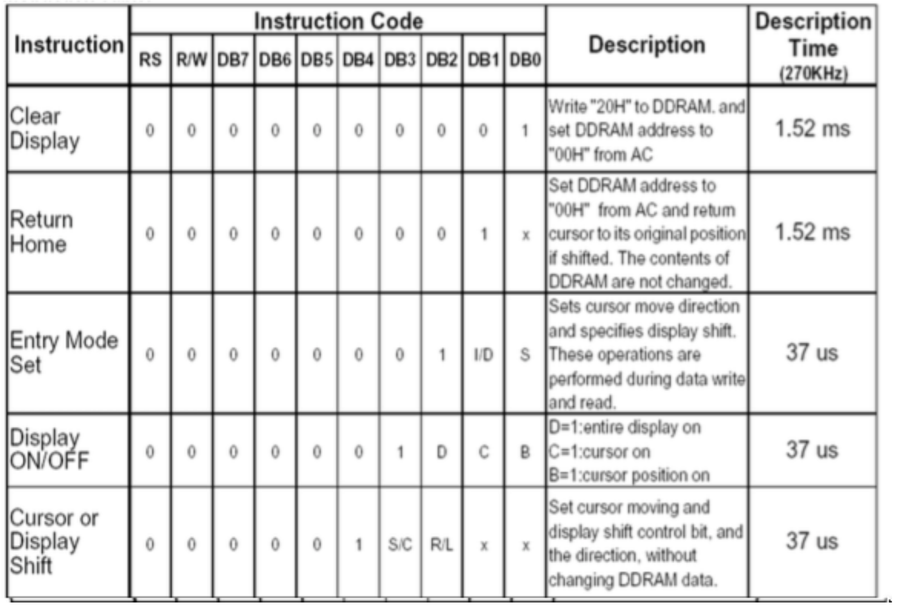
5. 0 PIN ASSIGNMENT

No.	Symbol	Level	Function	
1	Vss	--	0V	Power Supply
2	Vdd	--	+5V	
3	V0	--	for LCD	
4	RS	H/L	Register Select: H:Data Input L:Instruction Input	
5	R/W	H/L	H--Read L--Write	
6	E	H,H-L	Enable Signal	
7	DB0	H/L	Data bus used in 8 bit transfer	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L	Data bus for both 4 and 8 bit transfer	
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		
15	BLA	--	BLACKLIGHT +5V	
16	BLK	--	BLACKLIGHT 0V-	

LCD Module 1602A-1 specs - (eone-1602a1.pdf)

GPIO AND BUSES

- The LCD 1602 module is controlled through instructions (four categories) to:
  - <u>set display format, data length, scrolling modality</u>
  - set internal RAM address
  - perform data transfer from/to internal RAM
  - access status flags

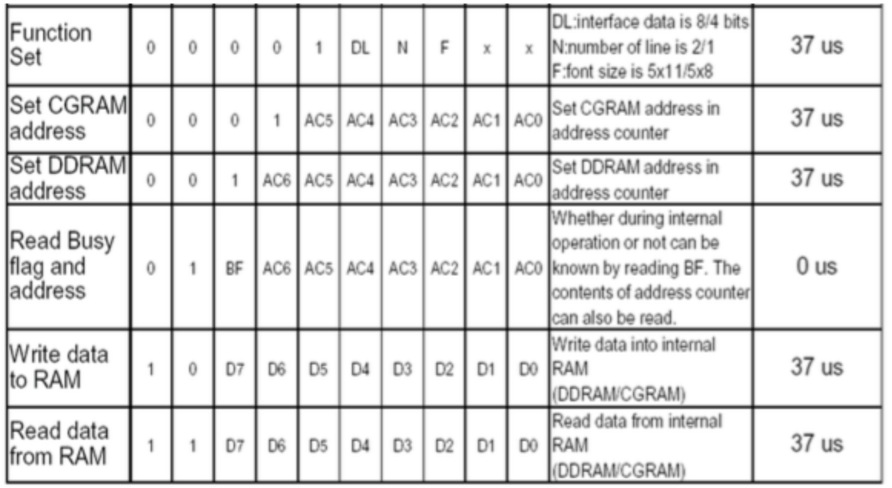


Instruction	Instruction Code										Description	Description Time (270KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM. and set DDRAM address to "00H" from AC	1.52 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.52 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 us
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	37 us
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	37 us

LCD Module 1602A-1 specs - (eone-1602a1.pdf)

GPIO AND BUSES

- The LCD 1602 module is controlled through instructions (four categories) to:
    - <u>set display format, data length, scrolling modality</u>
    - set internal RAM address
    - perform data transfer from/to internal RAM
    - access status flags



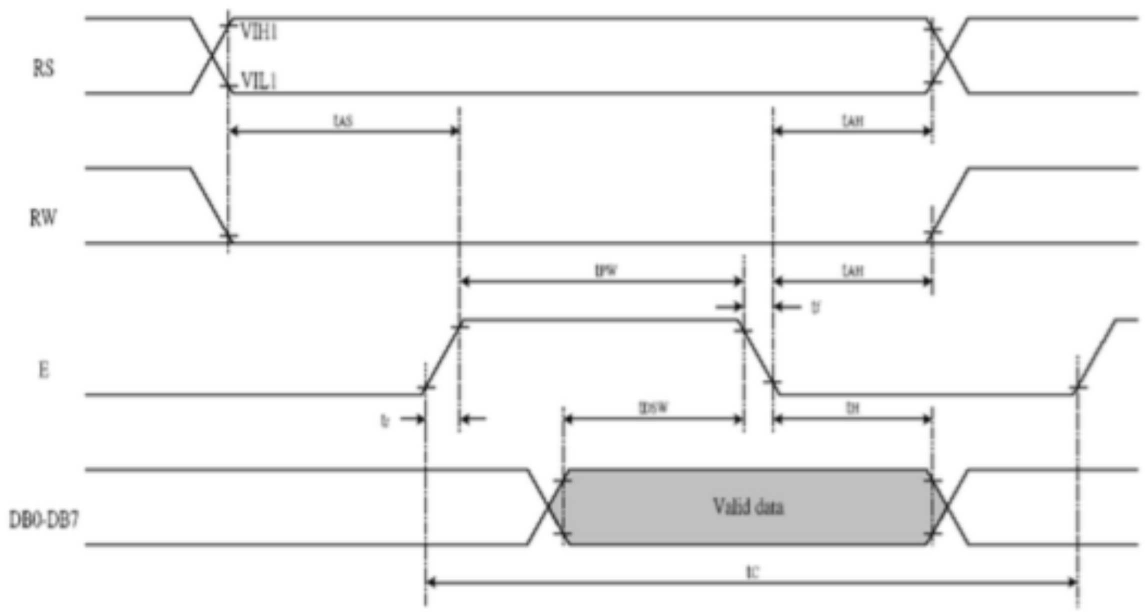
Function Set	0	0	0	0	1	DL	N	F	x	x	DL:interface data is 8/4 bits N:number of line is 2/1 F:font size is 5x11/5x8	37 us
Set CGRAM address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	37 us
Set DDRAM address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	37 us
Read Busy flag and address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 us
Write data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	37 us
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	37 us

LCD Module 1602A-1 specs - (eone-1602a1.pdf)

GPIO AND BUSES

- Writing to the controller chips

Writing data from MPU to ST7066U

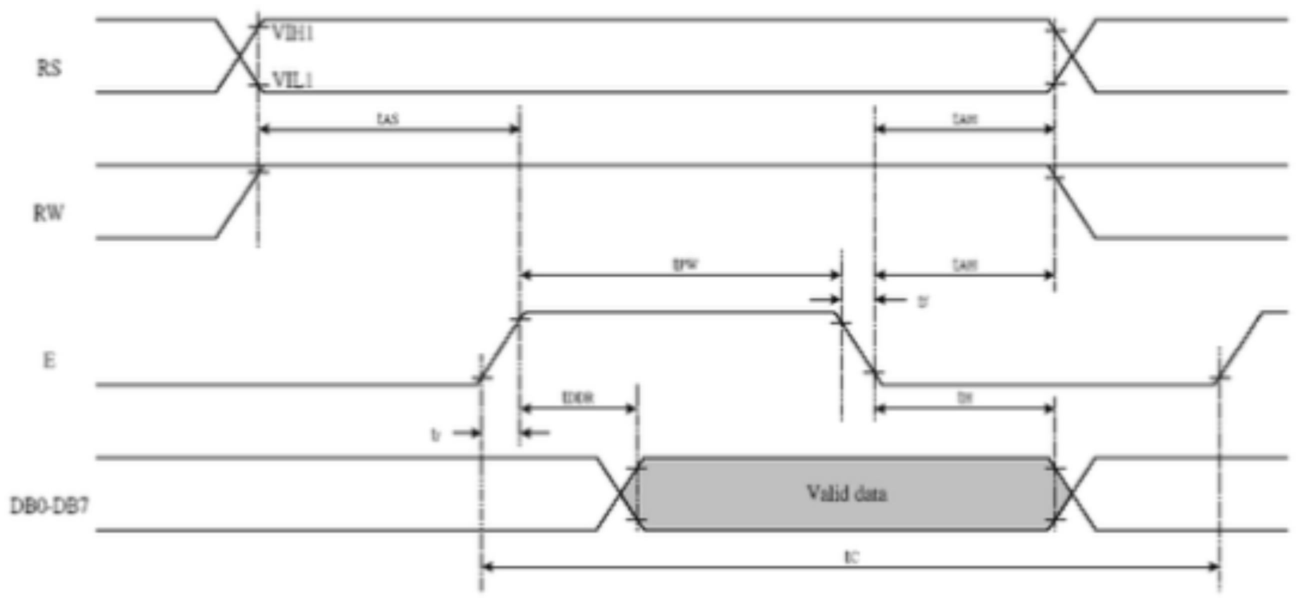


LCD Module 1602A-1 specs - (eone-1602a1.pdf)

GPIO AND BUSES

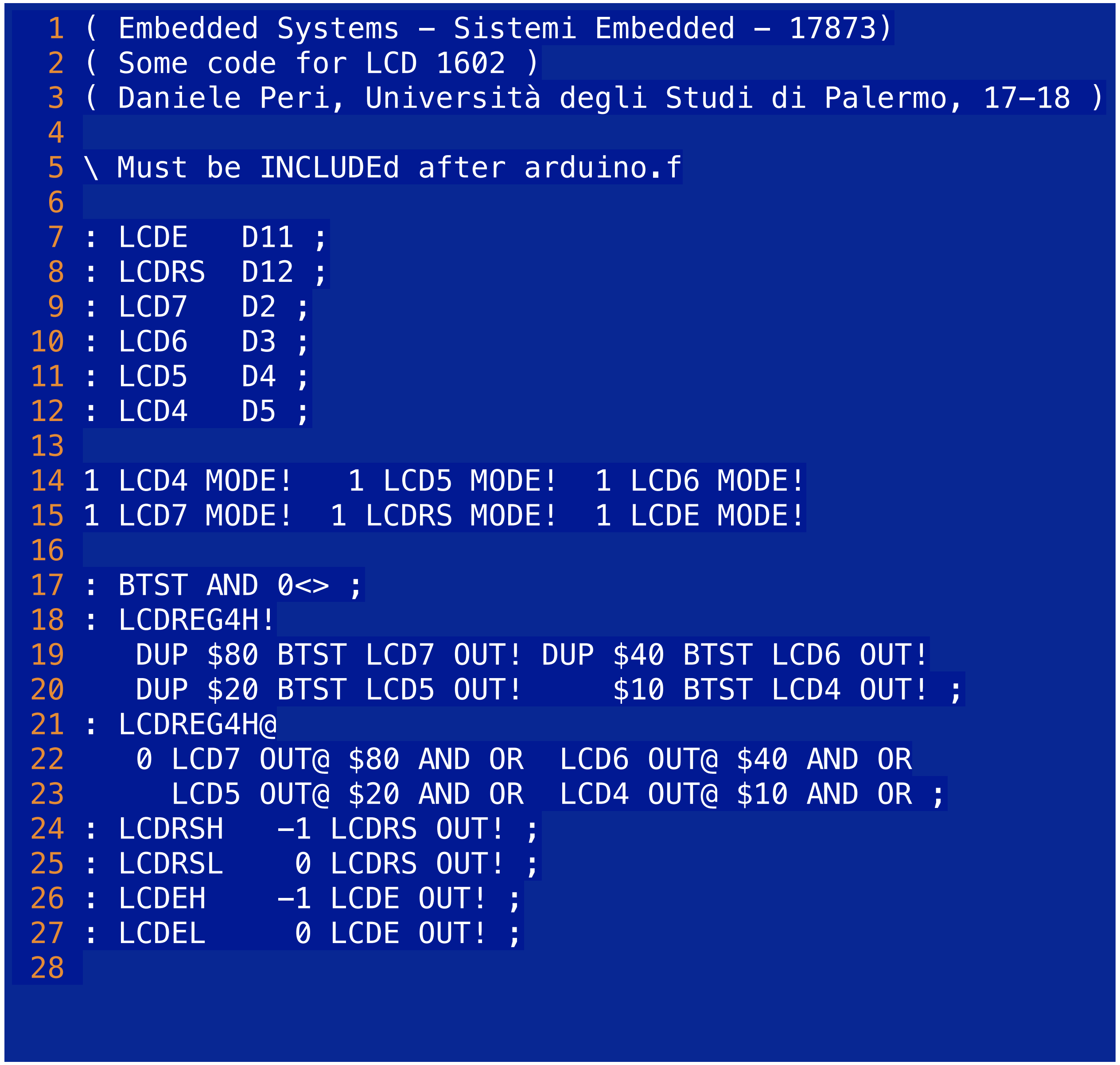
- Reading from the controller chips

Reading data from ST7066U to MPU



LCD Module 1602A-1 specs - (eone-1602a1.pdf)

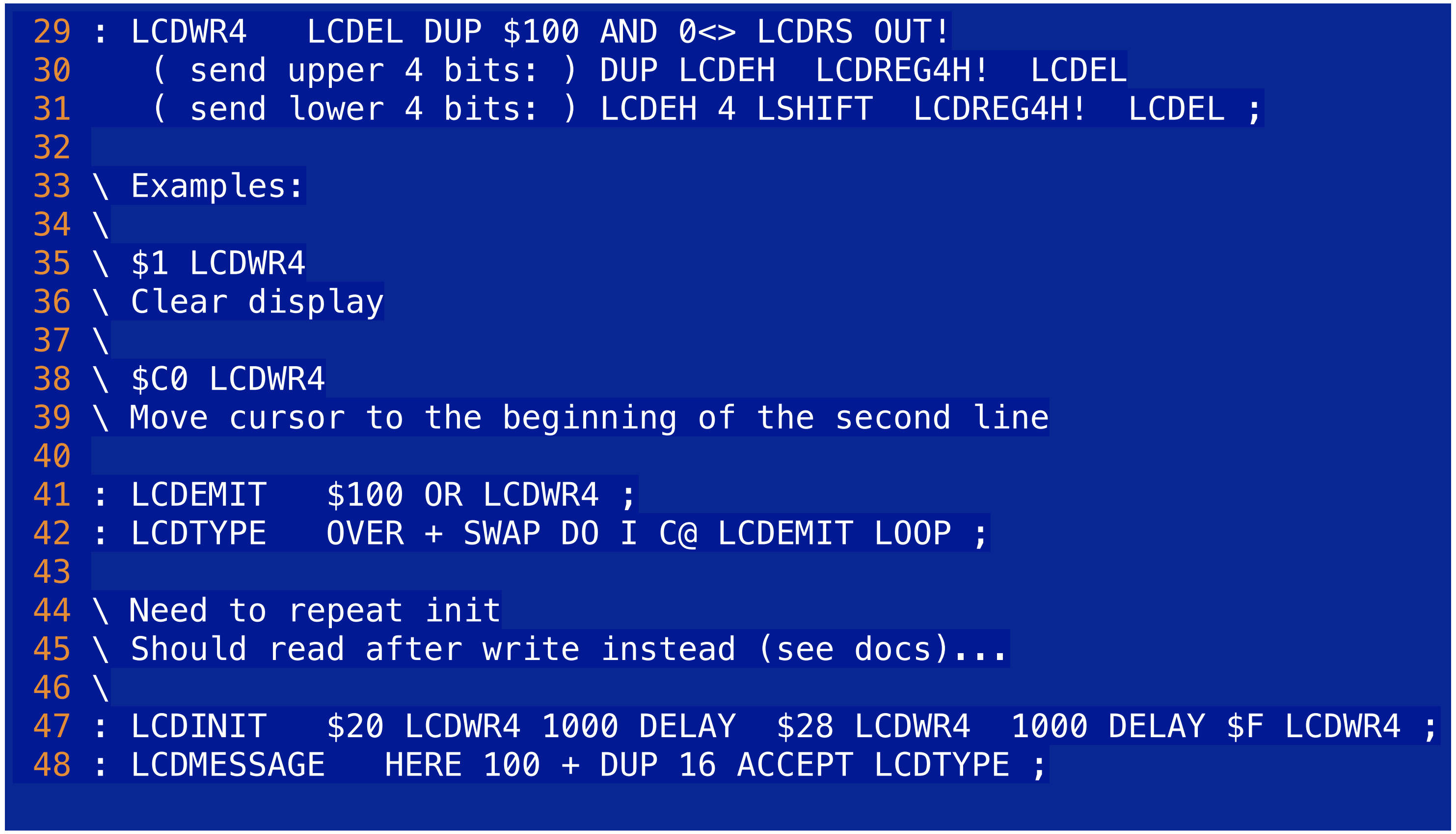
LCD 1602



```
( Embedded Systems – Sistemi Embedded – 17873)
( Some code for LCD 1602 )
( Daniele Peri, Università degli Studi di Palermo, 17–18 )

\ Must be INCLUDEd after arduino.f

: LCDE   D11 ;
: LCDRS  D12 ;
: LCD7   D2 ;
: LCD6   D3 ;
: LCD5   D4 ;
: LCD4   D5 ;

1 LCD4 MODE!   1 LCD5 MODE!  1 LCD6 MODE!
1 LCD7 MODE!  1 LCDRS MODE!  1 LCDE MODE!

: BTST AND 0<> ;
: LCDREG4H!
   DUP $80 BTST LCD7 OUT! DUP $40 BTST LCD6 OUT!
   DUP $20 BTST LCD5 OUT!     $10 BTST LCD4 OUT! ;
: LCDREG4H@
   0 LCD7 OUT@ $80 AND OR  LCD6 OUT@ $40 AND OR
     LCD5 OUT@ $20 AND OR  LCD4 OUT@ $10 AND OR ;
: LCDRSH   -1 LCDRS OUT! ;
: LCDRSL    0 LCDRS OUT! ;
: LCDEH    -1 LCDE OUT! ;
: LCDEL     0 LCDE OUT! ;
```

LCD-1602.f

NUCLEO-64 STM32F446



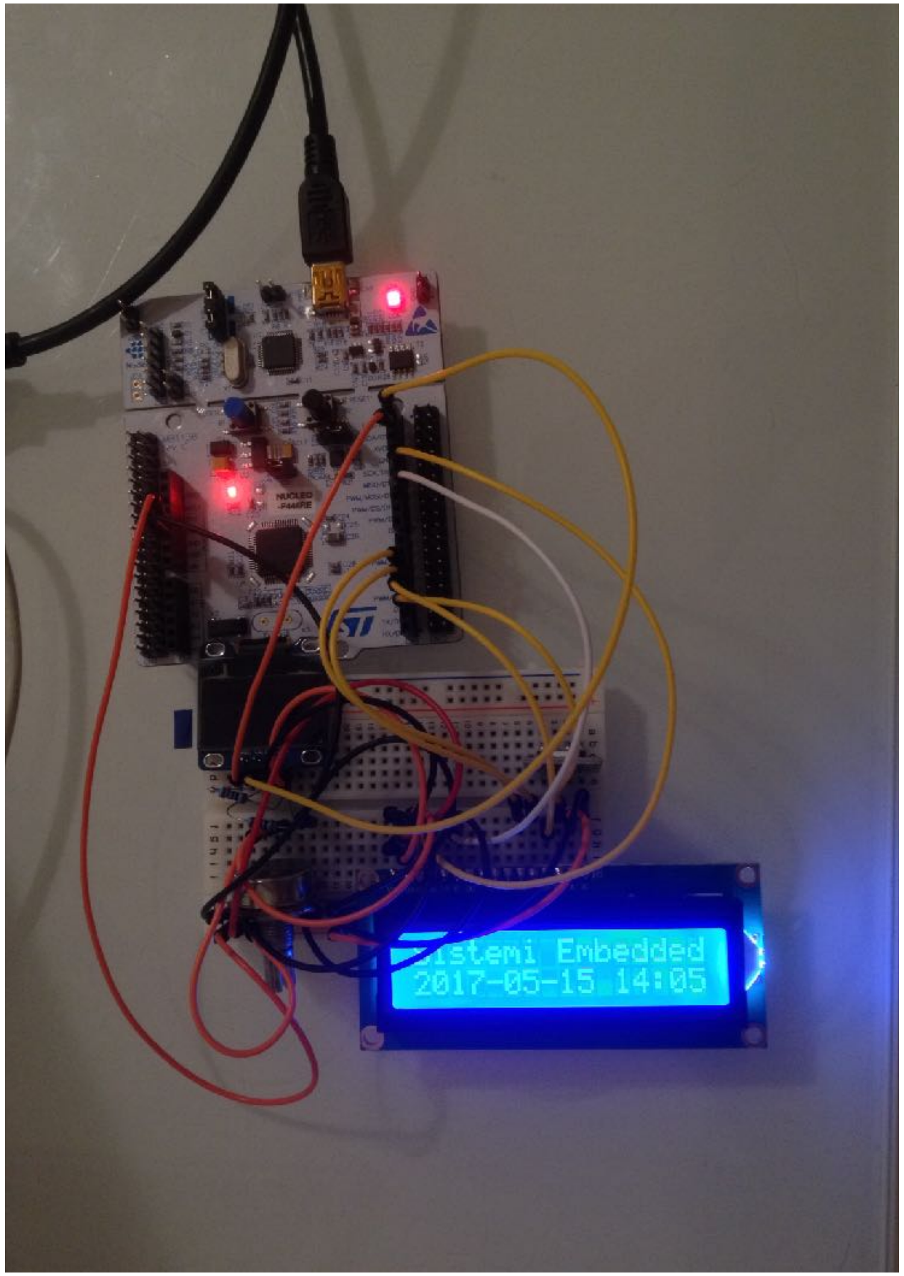
```
: LCDWR4   LCDEL DUP $100 AND 0<> LCDRS OUT!
   ( send upper 4 bits: ) DUP LCDEH  LCDREG4H!  LCDEL
   ( send lower 4 bits: ) LCDEH 4 LSHIFT  LCDREG4H!  LCDEL ;

\ Examples:
\
\ $1 LCDWR4
\ Clear display
\
\ $C0 LCDWR4
\ Move cursor to the beginning of the second line

: LCDEMIT   $100 OR LCDWR4 ;
: LCDTYPE   OVER + SWAP DO I C@ LCDEMIT LOOP ;

\ Need to repeat init
\ Should read after write instead (see docs)...
\
: LCDINIT   $20 LCDWR4 1000 DELAY  $28 LCDWR4  1000 DELAY $F LCDWR4 ;
: LCDMESSAGE   HERE 100 + DUP 16 ACCEPT LCDTYPE ;
```

LCD-1602.f

NUCLEO-64 STM32F446

- Writing a message to the display

```
$1 LCDWR4  ok.
LCDMESSAGE Sistemi Embedded  ok.
$C0 LCDWR4  ok.
LCDMESSAGE 2017-05-15 14:05  ok.
```



LCD-1602.f