COPYRIGHT

DISCLAIMER

This content is provided without warranty or representation of any kind. The use of the content is entirely at your own risk and Stellenbosch University (SU) will have no liability directly or indirectly as a result of this content.

The content must not be assumed to provide complete coverage of the particular study material. Content may be removed or changed without notice.

The video is of a recording with very limited post-recording editing. The video is intended for use only by SU students enrolled in the particular module.

UNIVERSITEIT
iYUNIVESITHI
STELLENBOSCH
UNIVERSITY

100
1918 · 2018

forward together · saam vorentoe · masiye phambili

Computer Systems / Rekenaarstelsels 245 - 2020

Lecture 10

Inputs and Outputs – Part 2
Intrees en Uittrees – Deel 2

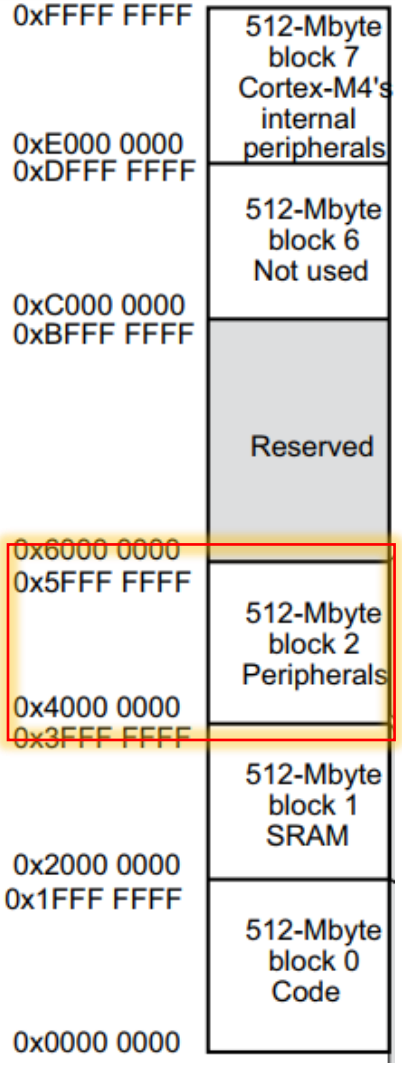
Dr Rensu Theart & Dr Lourens Visagie

Lecture Overview

- Configuring the STM32F4 GPIO
- Using the STM32F4 GPIO
- Displays
  - Bit banging

General-purpose Input/Output (GPIO) / Algemene Intree/Uittree

- Recall that a portion of the address space is dedicated to I/O devices rather than memory.
  - In our case the addresses in range 0x4000 0000 to 0x5FFF FFFF are used for I/O.
- Each I/O device is assigned one or more memory addresses in this range.
- A store to the specified address sends data to the device. A load receives data from the device.
- This method of communicating with I/O devices is called memory-mapped I/O.
  - A load or store may access either memory or an I/O device.



Configuring the STM32F4 GPIO

- Five ports (A..E), each with 16 I/Os (i.e. PA0, PA1,…, PA15, PB0, PB1…)
- For each of the 5 ports, A to E, there are 4 **I/O port control registers** that determine the configuration (x is the port letter, A to E).

1. The **GPIOx_MODER** register is used to select the I/O direction.
   - Input, output
   - Alternative function, analog. We will use this later.
2. The **GPIOx_PUPDR** register I used to activate pull-up/pull-down resistors.
3. The output type register (**GPIOx_OTYPER**) selects between output push-pull or open-drain.
4. The output speed register (**GPIOx_OSPEEDR**) controls how fast the output is changed.

Configuring the STM32F4 GPIO

- port mode register GPIOx_MODER (x = A..E)
- Address offset: 0x00

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

Configuring the STM32F4 GPIO

- Output type register GPIOx_OTYPER (x = A..E)
- Address offset: 0x04

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
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
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 **OTy**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the output type of the I/O port.

0: Output push-pull (reset state)

1: Output open-drain

Configuring the STM32F4 GPIO

- Output speed register GPIOx_OSPEEDR (x = A..E)
- Address offset: 0x08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [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
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: Fast speed

11: High speed

Configuring the STM32F4 GPIO

- Port pull-up/pull-down register GPIOx_PUPDR (x = A..E)
- Address offset: 0x0C

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

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)	
10	0	SPEED [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

Using the STM32F4 GPIO

- Each GPIO has two 16-bit memory-mapped data registers: input and output data registers.
- **GPIOx_ODR** stores the data to be output, it is read/write accessible.
  - **Write** to this register to output a voltage on the I/O pin.
- The data input through the I/O are stored into the input data register (**GPIOx_IDR**), a read-only register.
  - **Read** from this register to read the voltage state on the I/O pin.
  - Captures the data present on the I/O pin at every AHB clock cycle.

Using the STM32F4 GPIO

- Port input data register GPIOx_IDR (x = A..E)
- Address offset: 0x10

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
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.

Using the STM32F4 GPIO

- Port output data register GPIOx_ODR (x = A..E)
- Address offset: 0x14

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.

Using the STM32F4 GPIO

- The **bit set reset register** (**GPIOx_BSRR**) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR).
- The bit set reset register has twice the size of GPIOx_ODR. Each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BSRR(i) and BSRR(i+16):
  - Writing 1 to BSRR(i) **sets** the corresponding ODR(i) bit.
  - Writing 1 to BSRR(i+16) **resets** the ODR(i) corresponding bit.
- Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR.
- If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the STM32F4 GPIO

- GPIO port bit set/reset register GPIOx_BSRR (x = A..E)
- Address offset: 0x18

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)

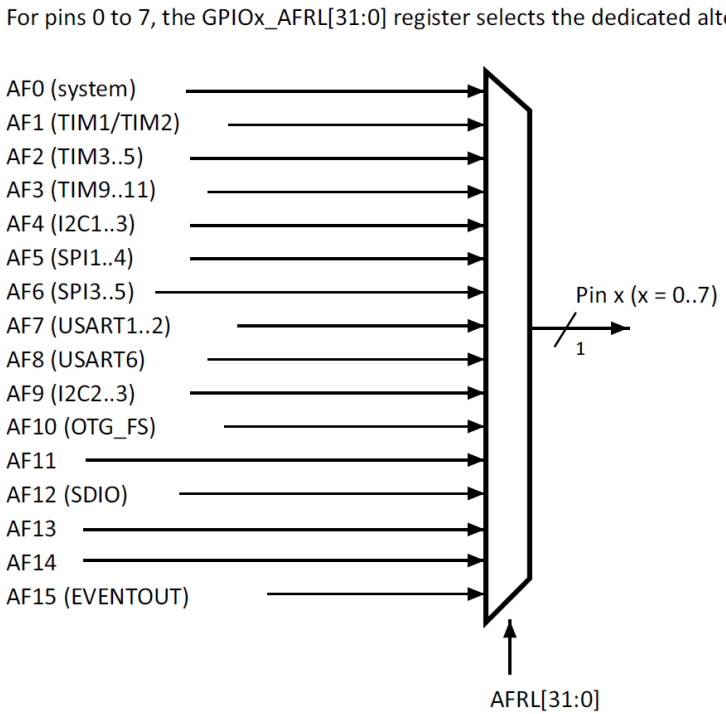
These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

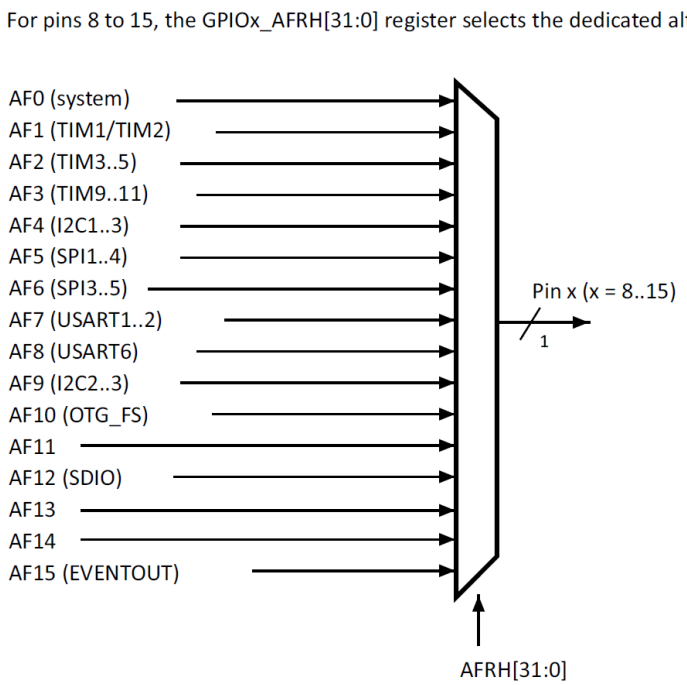
0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

Using the STM32F4 GPIO

- Two registers (**GPIOx_AFRL** and **GPIOx_AFRH**) are provided to select one out of the sixteen **alternate function** inputs/outputs available for each I/O.
- All the alternative functions are multiplexed together.
- Register contents are the select lines to the multiplexer that determines the pin's function.





Using the STM32F4 GPIO

- It is possible to freeze the GPIO control registers by applying a specific write sequence to the **GPIOx_LCKR** register.
  - The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

- Remember, using **External interrupt/event controller (EXTI)** a GPIO can also be set up to trigger an interrupt.

Using the STM32F4 GPIO

- Memory map of port registers

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

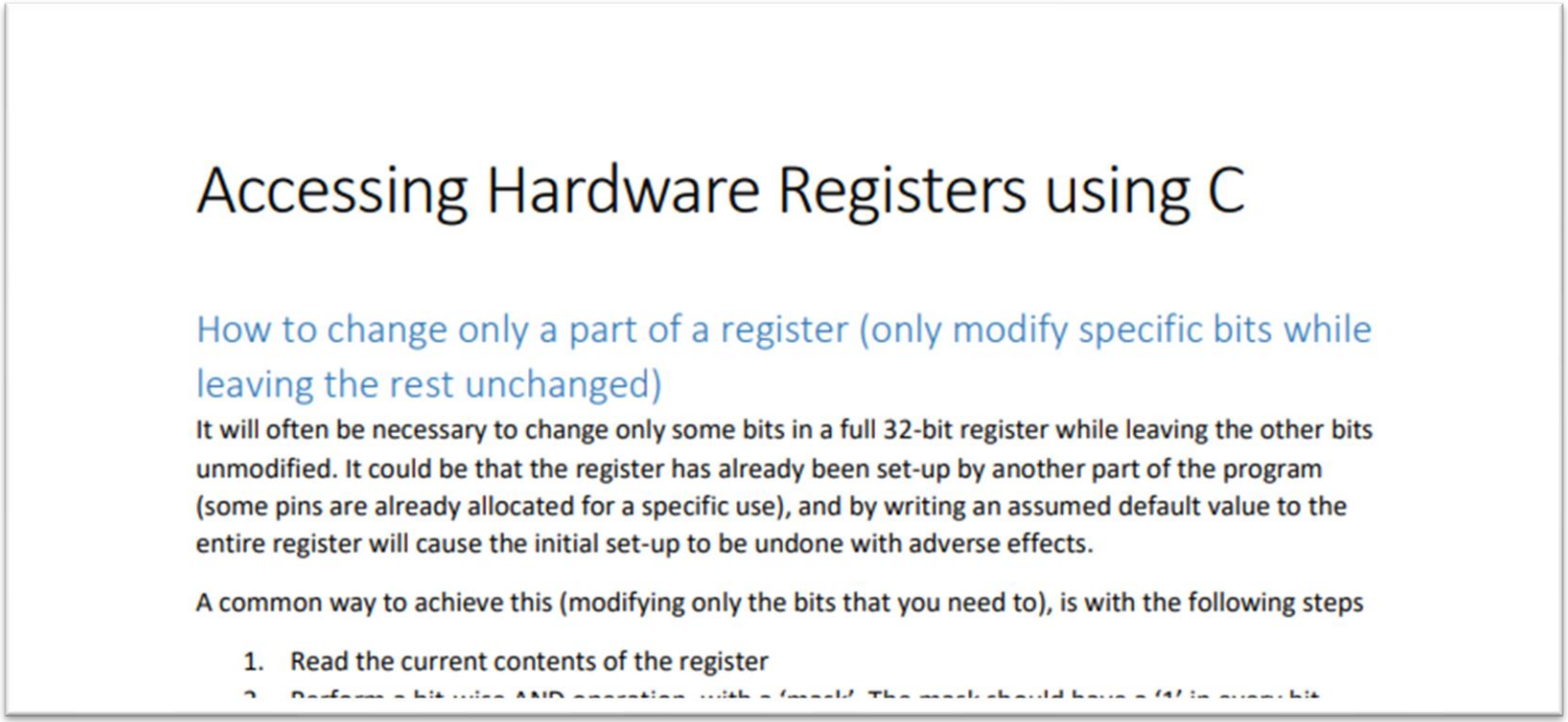
Offset	Register	Description
0 (0x00)	GPIOx_MODER	Mode register
4 (0x04)	GPIOx_OTYPER	Output type register
8 (0x08)	GPIOx_OSPEEDR	Output speed register
12 (0x0C)	GPIOx_PUPDR	Pull-up/pull-down register
16 (0x10)	GPIOx_IDR	Input data register
20 (0x14)	GPIOx_ODR	Output data register
24 (0x18)	GPIOx_BSRR	Set/reset register

Using the STM32F4 GPIO

- Setting things up in STM32CubeIDE.

Using the STM32F4 GPIO

- Accessing Hardware Registers using C (under Resources on SunLearn)



Accessing Hardware Registers using C

How to change only a part of a register (only modify specific bits while leaving the rest unchanged)

It will often be necessary to change only some bits in a full 32-bit register while leaving the other bits unmodified. It could be that the register has already been set-up by another part of the program (some pins are already allocated for a specific use), and by writing an assumed default value to the entire register will cause the initial set-up to be undone with adverse effects.

A common way to achieve this (modifying only the bits that you need to), is with the following steps

1. Read the current contents of the register

Using the STM32F4 GPIO

- **Example:** Read pin state from PE4
- GPIOE addresses start at 0x40021000, GPIOx_IDR offset is 16 (0x10), so GPIOE_IDR address is 0x40021010

Assembly

```
LDR     r0, =0x40021010
LDR     r1, [r0]    ; read the entire 16-bit port
LSR     r1, r1, #4 ; move bit 4 to least-significant position
AND     r1, r1, #1 ; clear all bits except least-significant bit
```

C

```
uint32_t pe4 = (*((uint32_t*) 0x40021010) >> 4 ) & 0x01;
```

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
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

Using the STM32F4 GPIO

- **Example:** Configure pin PD13 as output, **without changing** the configuration of **other pins**
- GPIOD addresses start at 0x40020C00, GPIOx_MODER offset is 0. For output configuration for PD13, Set bits 27,26 to 0b01

Assembly

```
LDR     r0, =0x40020C00
LDR     r1, [r0]           ; read the register
BIC     r1, r1, #0xC000000 ; clear only bits 27&26
ORR     r1, r1, #0x4000000 ; set bits 27,26 to 01
STR     r1, [r0]           ; set register value
```

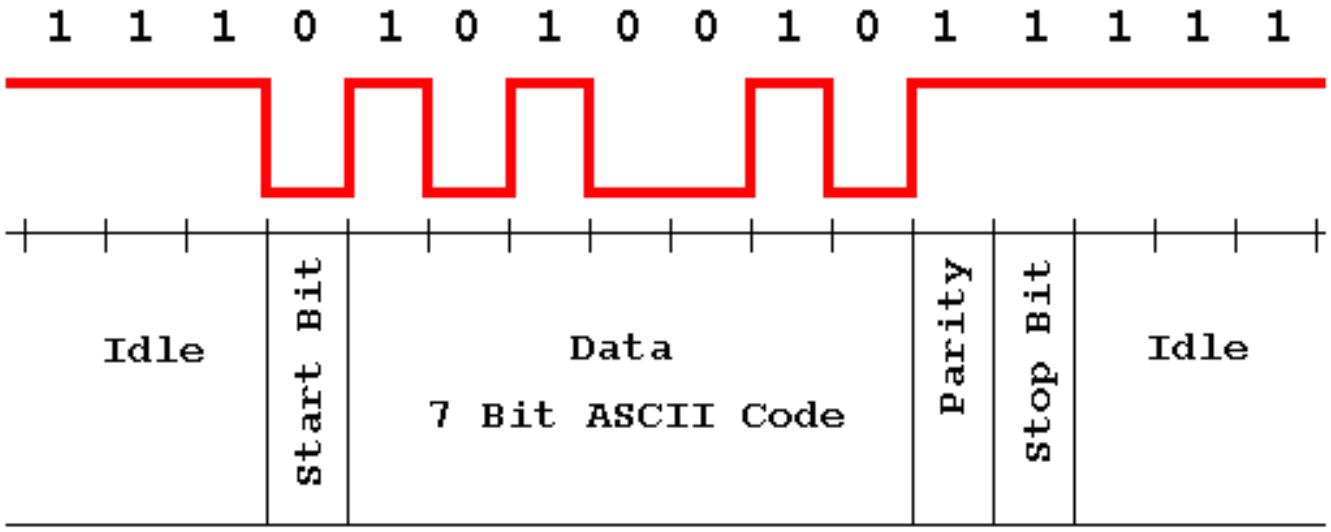
C

```
*((uint32_t*) 0x40020c00) = (*((uint32_t*) 0x40020c00) & ~0xc000000 ) + 0x4000000;
```

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]	

Using the STM32F4 GPIO

- Can we use GPIOs for communication?
- Signal changes over time – basis of inter-device communication.

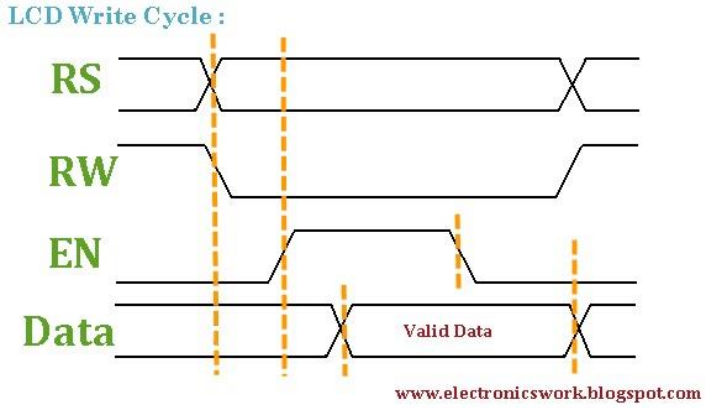


- We have several communication interfaces in hardware, such as SPI, I2C, UART. More about that later on…
- When transmitting or receiving signals with software using the GPIOs instead of dedicated hardware, we call this **bit banging.**
  - This is necessary for some applications such as LCD screens.

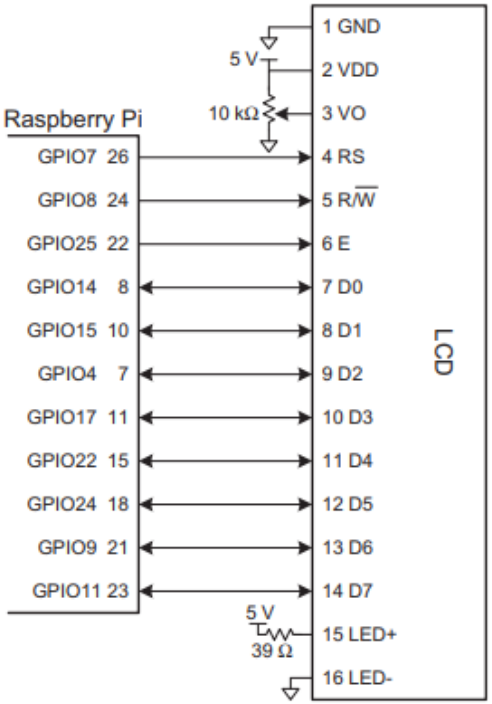
Displays / Vertooneenhede

LCD Character Display

- Parallel GPIO interface
- Write data or instruction to display, on every falling edge of E (enable) signal
- Requires specific timing between signal changes



PIN ASSIGNMENT		
Pin no.	Symbol	Function
1	Vss	Power supply(GND)
2	Vdd	Power supply(+)
3	Vo	Contrast Adjust
4	RS	Register select signal
5	R/$\overline{W}$	Data read / write
6	E	Enable signal
7	DB0	Data bus line
8	DB1	Data bus line
9	DB2	Data bus line
10	DB3	Data bus line
11	DB4	Data bus line
12	DB5	Data bus line
13	DB6	Data bus line
14	DB7	Data bus line
15	A	Power supply for LED B/L (+)
16	K	Power supply for LED B/L (–)



Displays / Vertooneenhede

LCD Character Display

● INSTRUCTIONS

Instruction	Code										Description	Executed Time(max.)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	1.64mS
Cursor Art Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home positio (Address 0). Also returns the display bing shifted to the original position. DD RAM contents remain unchanged.	1.64mS
Entry Mode Set	0	0	0	0	0	0	0	1	1/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 µS
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor NO/OFF (C), and blink of cursor position character (B).	40 µS
Cursor /Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents.	40 µS
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F)	40 µS
CG RAM Address Set	0	0	0	1	ACG						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 µS
DD RAM Address Set	0	0	1	ADD							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 µS
Busy Flag/ Address Read	0	1	BF	AC							Reads Busy flag (FB) indicating internal operation is being performed and reads address counter counts.	0 µS
CG RAM / DD RAM Data Write	1	0	WRITE DATA								Writes data into DD RAM or CG RAM.	40 µS
CG RAM / DD RAM Data Read	1	1	READ DATA								Reads data from DD RAM or CG RAM.	40 µS

LCD Character Display - Code implementation

```
void lcd_outputbyte(uint8_t rs, uint8_t data)
{
    // R/W = low, RS = rs
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_1, 0);  // R/W
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, rs); // RS

    HAL_Delay(1);

    // set data pin states
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, data & 0x01);         // D0
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, (data >> 1) & 0x01);  // D1
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, (data >> 2) & 0x01);  // D2
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, (data >> 3) & 0x01);  // D3
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, (data >> 4) & 0x01);  // D4
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, (data >> 5) & 0x01);  // D5
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, (data >> 6) & 0x01);  // D6
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, (data >> 7) & 0x01);  // D7

    // Pulse enable
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_0, 0);  // E
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_0, 1);  // E
    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_0, 0);  // E
}
```

```
void lcd_refresh(char* output)
{
    // set display address to 0
    lcd_outputbyte(0, 0x80);
    HAL_Delay(2);  // at least 1.6ms
    char* p_char = output;
    for (int i = 0; i < 8; i++)
    {
        if(*p_char != '\0')
        {
            lcd_outputbyte(1, *p_char);
            p_char++;
        }
        else
        {
            lcd_outputbyte(1, ' ');
        }
    }

    // set display address to second set of 8 chars
    lcd_outputbyte(0, 0xA8);
    HAL_Delay(2);  // at least 1.6ms
    for (int i = 8; i < 16; i++)
    {
        if(*p_char != '\0')
        {
            lcd_outputbyte(1, *p_char);
            p_char++;
        }
        else
        {
            lcd_outputbyte(1, ' ');
        }
    }
}
```

Display position DDRAM address

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

2-Line display mode

Displays / Vertooneenhede

- Bit banging can even be done manually (by-hand)

Toggle switches

https://www.youtube.com/watch?v=hZRL8luuPb8