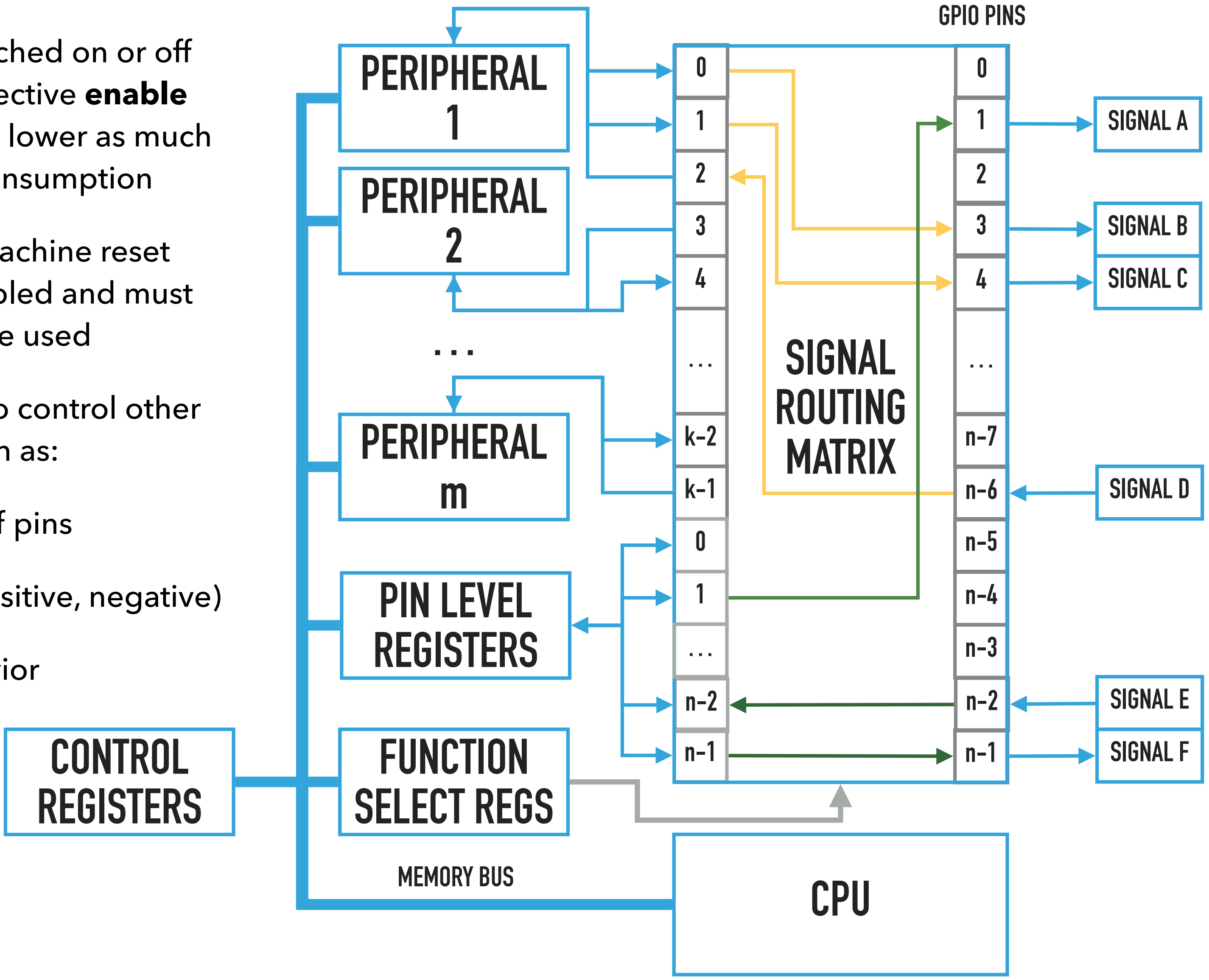
## GENERAL-PURPOSE I/O (GPIO)

- Often, peripherals can be switched on or off by setting or clearing the respective **enable bit** (**EN**) in a control register to lower as much as possible the total energy consumption
    - Usually, at **power up** or machine reset most peripherals are disabled and must be explicitly **enabled** to be used
- Control registers permit also to control other aspects of signal handling such as:
    - Electrical characteristics of pins
    - Signal edge detection (positive, negative)
    - Interrupt triggering behavior



## GENERAL-PURPOSE I/O (GPIO)

- The signal routing matrix physically connects either an I/O internal function line or a pin-level register bit to an external pin
- In the table on the right the function selection registers contains two bits for each pin
  - Options for pin mode are: INPut (`00`), OUTput (`01`), AF0 (`10`), AF1 (`11`)
  - For each pin, AF0 and AF1 are assigned to different subset of functions
  - Only AFs corresponding to white cells are available for each pin

| | | | Functions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Peripheral | | | 1 | | | | 2 | | … | m | |
| pin # | FSEL bits | FUNC | 0 | 1 | 2 | 3 | 4 | 5 | | k-2 | k-1 |
| 0 | 00 | INP | □ | ■ | ■ | □ | ■ | ■ | ■ | ■ | ■ |
| 1 | 00 | INP | ■ | ■ | □ | ■ | ■ | □ | ■ | ■ | ■ |
| 2 | 00 | INP | ■ | □ | ■ | □ | ■ | ■ | ■ | ■ | ■ |
| 3 | 10 | AF0 | X | ■ | ■ | □ | ■ | ■ | ■ | ■ | ■ |
| 4 | 11 | AF1 | □ | X | ■ | ■ | ■ | ■ | ■ | ■ | ■ |
| … | … | … | … | … | … | … | … | … | … | … | … |
| n-7 | 00 | INP | ■ | ■ | ■ | ■ | □ | ■ | ■ | □ | ■ |
| n-6 | 10 | AF0 | ■ | ■ | X | ■ | ■ | □ | ■ | ■ | ■ |
| n-5 | 00 | INP | ■ | ■ | ■ | ■ | □ | ■ | ■ | □ | ■ |
| n-4 | 00 | INP | ■ | ■ | ■ | □ | ■ | ■ | ■ | ■ | □ |
| n-3 | 00 | INP | ■ | ■ | □ | ■ | ■ | ■ | □ | ■ | ■ |
| n-2 | 00 | INP | ■ | ■ | ■ | ■ | □ | ■ | ■ | ■ | □ |
| n-1 | 01 | OUT | ■ | ■ | ■ | ■ | ■ | □ | ■ | □ | ■ |

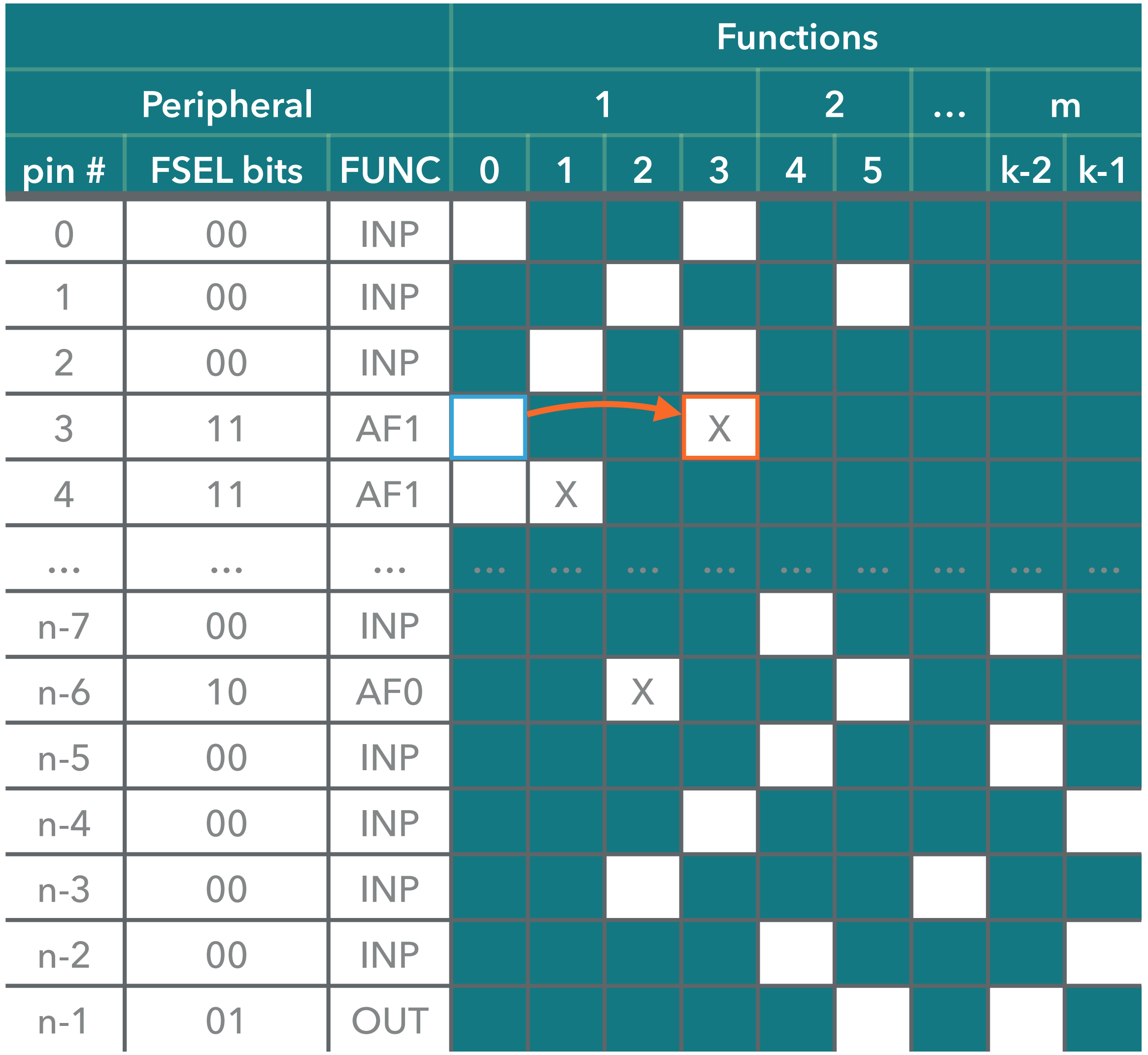
## GENERAL-PURPOSE I/O (GPIO)

- In the table on the right:
  - An X indicates a connection between a function and a pin. Only one white cell in a row can be assigned an X
  - Pin 3 is connected to function 0, pin 4 to function 1 and pin n-6 to function 2
  - The other pins are connected to pin-level registers

| | | | Functions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Peripheral | | | 1 | | | | 2 | | … | m | |
| pin # | FSEL bits | FUNC | 0 | 1 | 2 | 3 | 4 | 5 | | k-2 | k-1 |
| 0 | 00 | INP | ⬜ | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 |
| 1 | 00 | INP | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | 🟦 |
| 2 | 00 | INP | 🟦 | ⬜ | 🟦 | ⬜ | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 |
| 3 | 10 | AF0 | X | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 |
| 4 | 11 | AF1 | ⬜ | X | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 |
| … | … | … | … | … | … | … | … | … | … | … | … |
| n-7 | 00 | INP | 🟦 | 🟦 | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | ⬜ | 🟦 |
| n-6 | 10 | AF0 | 🟦 | 🟦 | X | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | 🟦 |
| n-5 | 00 | INP | 🟦 | 🟦 | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | ⬜ | 🟦 |
| n-4 | 00 | INP | 🟦 | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | 🟦 | 🟦 | ⬜ |
| n-3 | 00 | INP | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 |
| n-2 | 00 | INP | 🟦 | 🟦 | 🟦 | 🟦 | ⬜ | 🟦 | 🟦 | 🟦 | ⬜ |
| n-1 | 01 | OUT | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | ⬜ | 🟦 | ⬜ | 🟦 |

## GENERAL-PURPOSE I/O (GPIO)

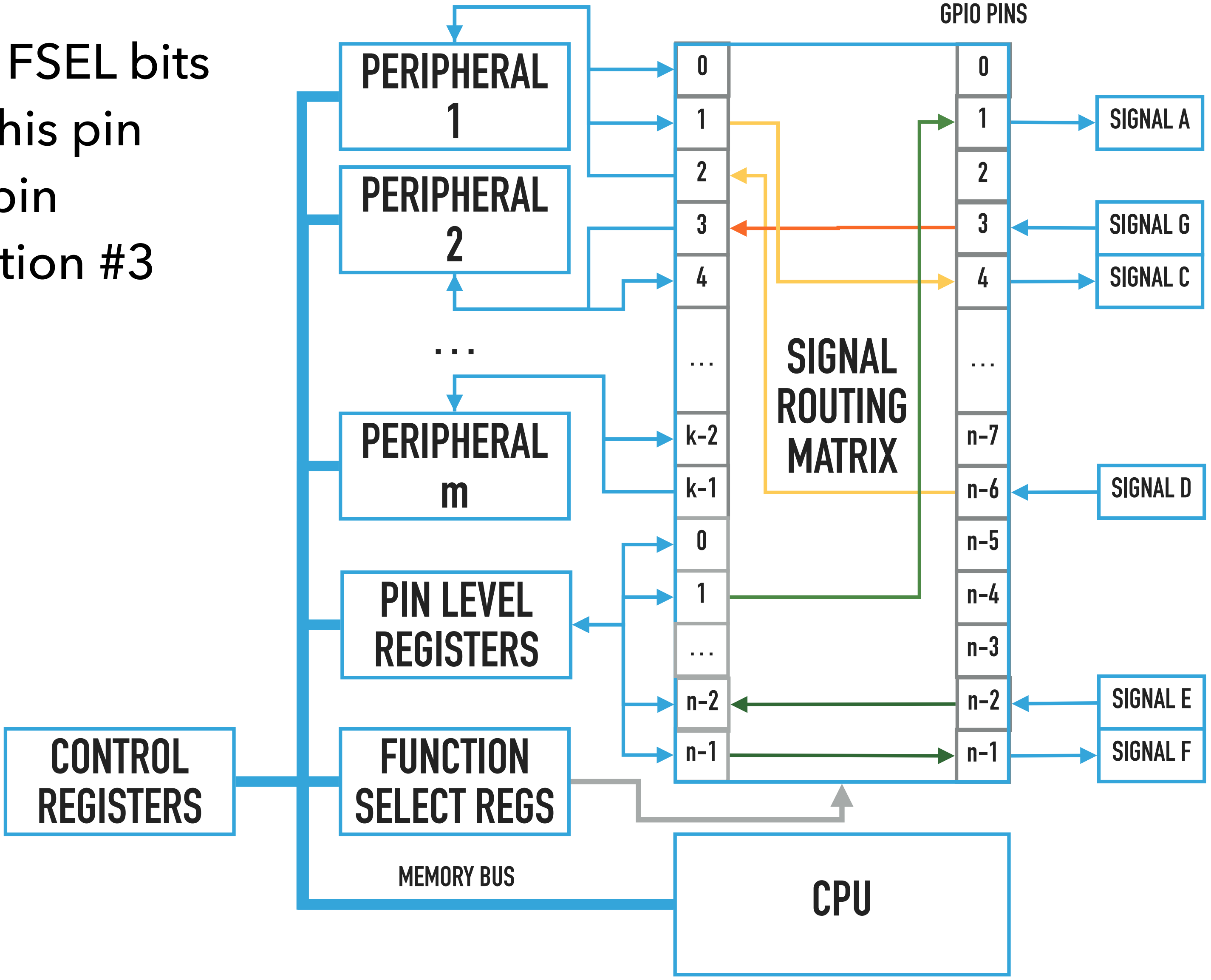
- If the FSEL bits at row #3 are changed to 11, FUNC becomes AF1 and the X moves from the cell at column 0 to the cell at column 3



| | | | Functions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Peripheral | | | 1 | | | | 2 | | … | m | |
| pin # | FSEL bits | FUNC | 0 | 1 | 2 | 3 | 4 | 5 | | k-2 | k-1 |
| 0 | 00 | INP | ⬜ | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 1 | 00 | INP | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | 🟩 |
| 2 | 00 | INP | 🟩 | ⬜ | 🟩 | ⬜ | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 3 | 11 | AF1 | ⬜ | 🟩 | 🟩 | X | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 4 | 11 | AF1 | ⬜ | X | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| … | … | … | … | … | … | … | … | … | … | … | … |
| n-7 | 00 | INP | 🟩 | 🟩 | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | ⬜ | 🟩 |
| n-6 | 10 | AF0 | 🟩 | 🟩 | X | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | 🟩 |
| n-5 | 00 | INP | 🟩 | 🟩 | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | ⬜ | 🟩 |
| n-4 | 00 | INP | 🟩 | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | 🟩 | 🟩 | ⬜ |
| n-3 | 00 | INP | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 |
| n-2 | 00 | INP | 🟩 | 🟩 | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 | 🟩 | ⬜ |
| n-1 | 01 | OUT | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | ⬜ | 🟩 | ⬜ | 🟩 |

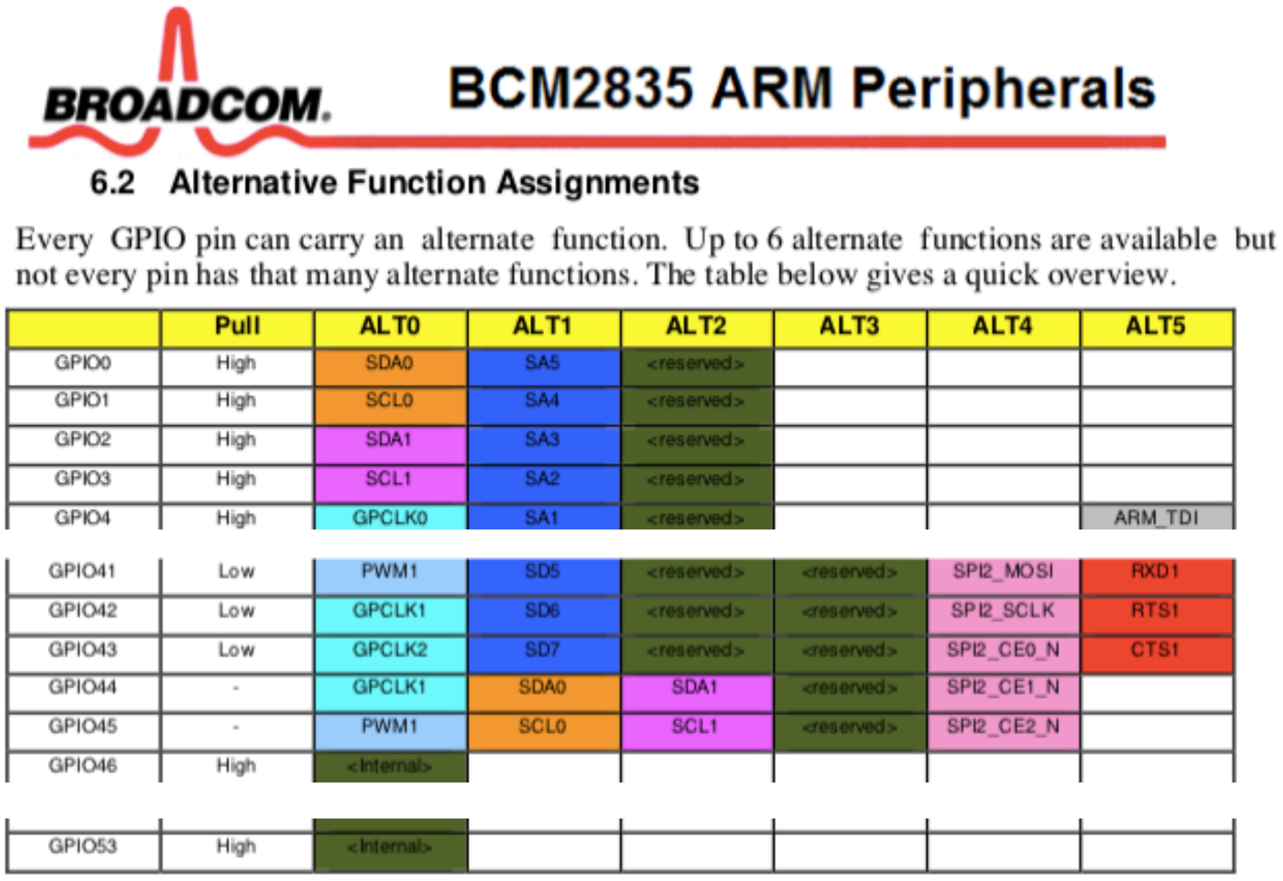
## GENERAL-PURPOSE I/O (GPIO)

- The change in the FSEL bits for pin #3 makes this pin become an input pin connected to function #3



## GENERAL-PURPOSE I/O (GPIO)

- For example, the SoC used in the Raspberry provides 8 configuration choices for each pin:
  - Input, output, ALT0, ALT1, ALT2, ALT3, ALT4, and ALT5
- Not all the possible ALTx functions for each pin are necessarily connected to a peripheral
- The set of all functions of the internal peripherals are spread all over the GPIO pins
- A table in the specifications provides the possible choices for each pin
  - For instance
    - GPIO4 may only be configured as GPCLK0, SA1 or ARM_TDI
    - GPIO41 may only configured as PWM1, SD5, SPI2_MOSI, or RXD1



BROADCOM. BCM2835 ARM Peripherals

6.2 Alternative Function Assignments

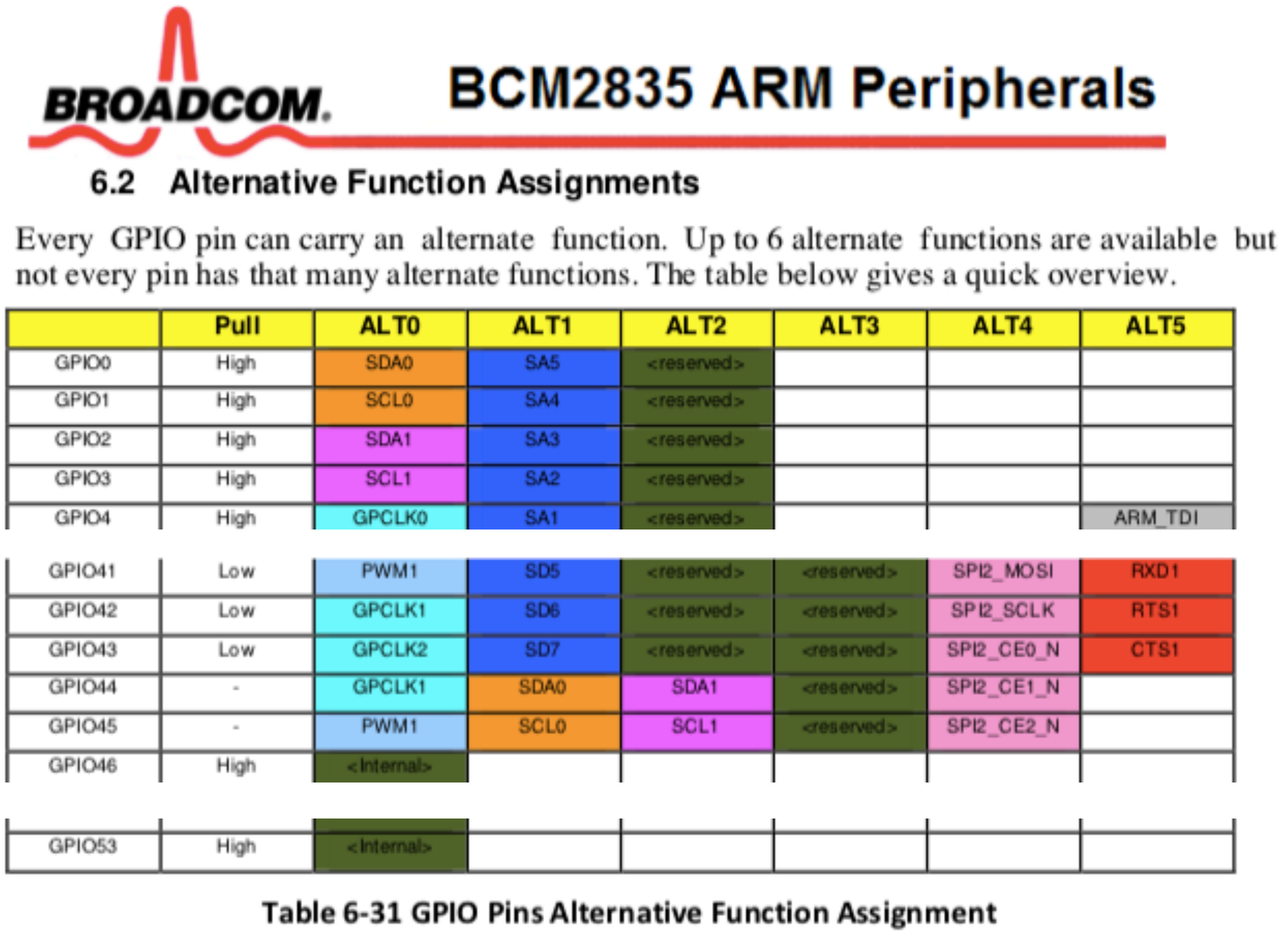
Every GPIO pin can carry an alternate function. Up to 6 alternate functions are available but not every pin has that many alternate functions. The table below gives a quick overview.

| | Pull | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 |
|---|---|---|---|---|---|---|---|
| GPIO0 | High | SDA0 | SA5 | <reserved> | | | |
| GPIO1 | High | SCL0 | SA4 | <reserved> | | | |
| GPIO2 | High | SDA1 | SA3 | <reserved> | | | |
| GPIO3 | High | SCL1 | SA2 | <reserved> | | | |
| GPIO4 | High | GPCLK0 | SA1 | <reserved> | | | ARM_TDI |
| GPIO41 | Low | PWM1 | SD5 | <reserved> | <reserved> | SPI2_MOSI | RXD1 |
| GPIO42 | Low | GPCLK1 | SD6 | <reserved> | <reserved> | SPI2_SCLK | RTS1 |
| GPIO43 | Low | GPCLK2 | SD7 | <reserved> | <reserved> | SPI2_CE0_N | CTS1 |
| GPIO44 | - | GPCLK1 | SDA0 | SDA1 | <reserved> | SPI2_CE1_N | |
| GPIO45 | - | PWM1 | SCL0 | SCL1 | <reserved> | SPI2_CE2_N | |
| GPIO46 | High | <Internal> | | | | | |
| GPIO53 | High | <Internal> | | | | | |

Table 6-31 GPIO Pins Alternative Function Assignment

## GPIO

- Functions from the same peripherals are indicated with the same color and must be available on different pins
  - SDA0 is available on GPIO0 while SCL0 on GPIO1 as both are needed for the peripheral operation
- Functions of a peripheral can be replicated in other pin sets
  - SDA0 is also available on GPIO44 and SCL0 is available on GPIO45
  - This way, for instance, by enabling SDA0 on GPIO44, SCL0 on GPIO45, SA5 on GPIO0 and SA4 on GPIO1, all these functions can coexist in the same application



BROADCOM. **BCM2835 ARM Peripherals**

**6.2 Alternative Function Assignments**

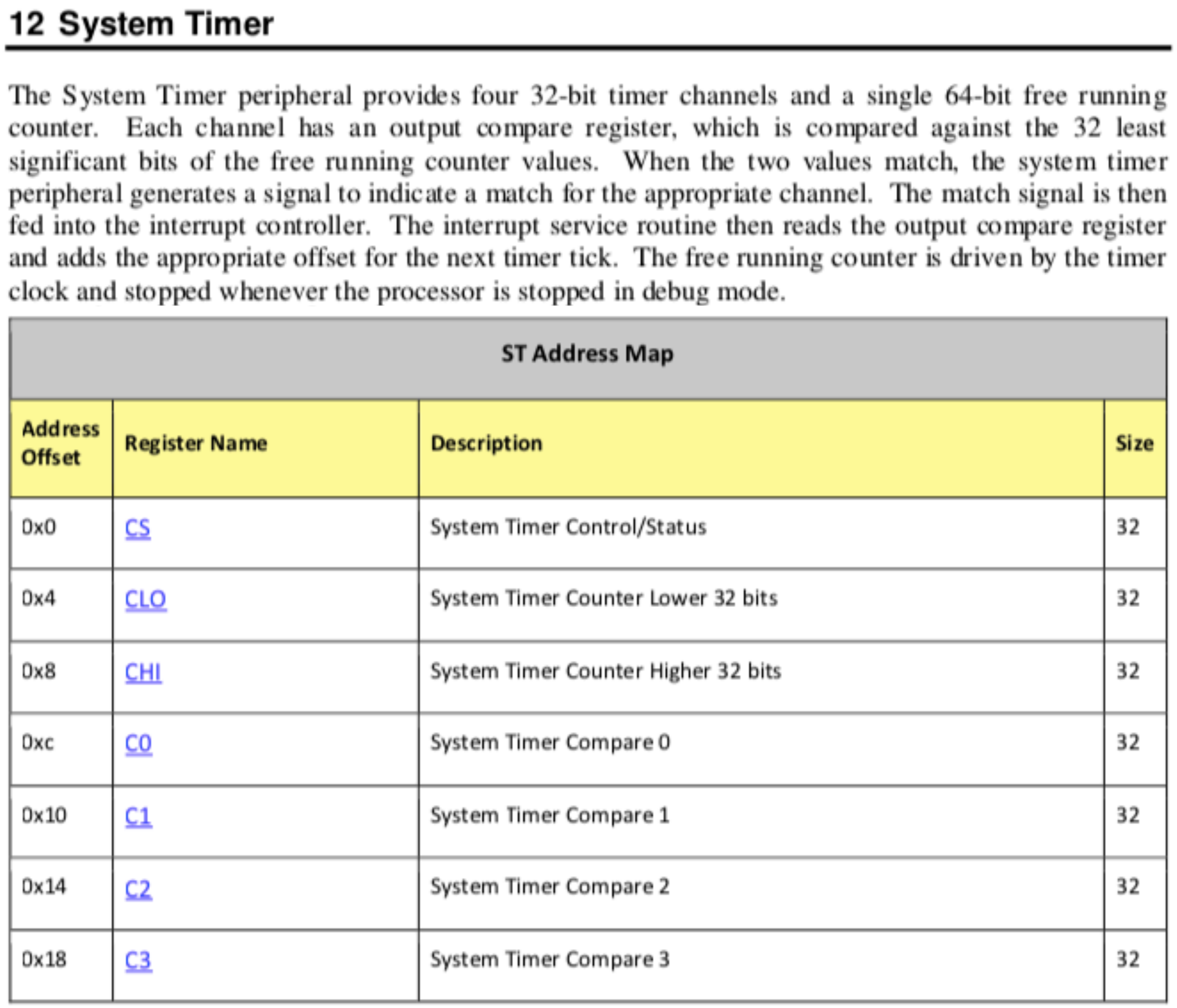
Every GPIO pin can carry an alternate function. Up to 6 alternate functions are available but not every pin has that many alternate functions. The table below gives a quick overview.

| | Pull | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 |
|---|---|---|---|---|---|---|---|
| GPIO0 | High | SDA0 | SA5 | <reserved> | | | |
| GPIO1 | High | SCL0 | SA4 | <reserved> | | | |
| GPIO2 | High | SDA1 | SA3 | <reserved> | | | |
| GPIO3 | High | SCL1 | SA2 | <reserved> | | | |
| GPIO4 | High | GPCLK0 | SA1 | <reserved> | | | ARM_TDI |
| GPIO41 | Low | PWM1 | SD5 | <reserved> | <reserved> | SPI2_MOSI | RXD1 |
| GPIO42 | Low | GPCLK1 | SD6 | <reserved> | <reserved> | SPI2_SCLK | RTS1 |
| GPIO43 | Low | GPCLK2 | SD7 | <reserved> | <reserved> | SPI2_CE0_N | CTS1 |
| GPIO44 | - | GPCLK1 | SDA0 | SDA1 | <reserved> | SPI2_CE1_N | |
| GPIO45 | - | PWM1 | SCL0 | SCL1 | <reserved> | SPI2_CE2_N | |
| GPIO46 | High | <Internal> | | | | | |
| GPIO53 | High | <Internal> | | | | | |

**Table 6-31 GPIO Pins Alternative Function Assignment**

## PERIPHERAL REGISTERS

- Peripheral of many different kinds can be included in SoCs
- All provide a set of registers located each one at a fixed offset from a base address
- The memory mapping circuitry, either a simple decoding logic or an MMU, defines the base address
    - Sometimes the mapping circuitry allows for changing the base address
- The System Timer of the Pi 3 B+ has 5 32-bit registers and one 64-bit register
- The 64-bit register, exposed to the memory bus as two consequent 32-bit memory words, provides a value counting elapsed µs since system started



### 12 System Timer

The System Timer peripheral provides four 32-bit timer channels and a single 64-bit free running counter. Each channel has an output compare register, which is compared against the 32 least significant bits of the free running counter values. When the two values match, the system timer peripheral generates a signal to indicate a match for the appropriate channel. The match signal is then fed into the interrupt controller. The interrupt service routine then reads the output compare register and adds the appropriate offset for the next timer tick. The free running counter is driven by the timer clock and stopped whenever the processor is stopped in debug mode.

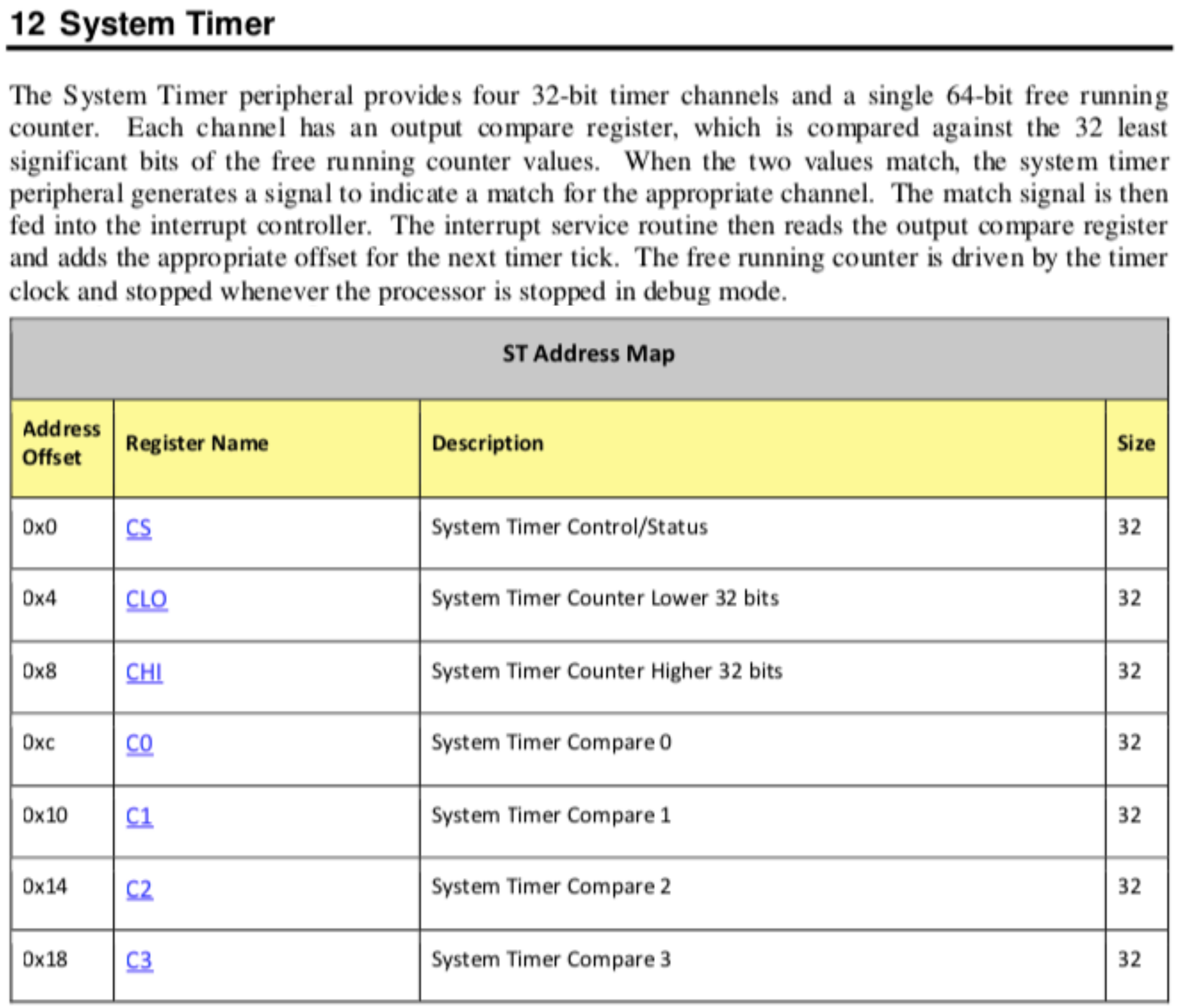
ST Address Map

| Address Offset | Register Name | Description | Size |
|---|---|---|---|
| 0x0 | CS | System Timer Control/Status | 32 |
| 0x4 | CLO | System Timer Counter Lower 32 bits | 32 |
| 0x8 | CHI | System Timer Counter Higher 32 bits | 32 |
| 0xc | C0 | System Timer Compare 0 | 32 |
| 0x10 | C1 | System Timer Compare 1 | 32 |
| 0x14 | C2 | System Timer Compare 2 | 32 |
| 0x18 | C3 | System Timer Compare 3 | 32 |

BCM2837-ARM-Peripherals.-.Revised.-.V2-1 **p. 172** - (BCM2837-ARM-Peripherals.-.Revised.-.V2-1.pdf)

## PERIPHERAL REGISTERS

- Even if the System Timer is apparently not tied to the outer environment, it provides input values to the CPU
- In the ARM addressing space of the Pi 3 B+ with disabled MMU, the base address of the system timer is **`0x3F003000`**
- To measure time CPU can just read the start time from register **`CLO–CHI`** and then enter a busy loop that reads the current register value, subtracts the start value and compares the result until the latter exceeds a given value (**polling**)
- More efficient ways to use the timer involve registers **`C0–C3`** and the ability of the timer to trigger interrupts



### 12 System Timer

The System Timer peripheral provides four 32-bit timer channels and a single 64-bit free running counter. Each channel has an output compare register, which is compared against the 32 least significant bits of the free running counter values. When the two values match, the system timer peripheral generates a signal to indicate a match for the appropriate channel. The match signal is then fed into the interrupt controller. The interrupt service routine then reads the output compare register and adds the appropriate offset for the next timer tick. The free running counter is driven by the timer clock and stopped whenever the processor is stopped in debug mode.

**ST Address Map**

| Address Offset | Register Name | Description | Size |
|---|---|---|---|
| 0x0 | CS | System Timer Control/Status | 32 |
| 0x4 | CLO | System Timer Counter Lower 32 bits | 32 |
| 0x8 | CHI | System Timer Counter Higher 32 bits | 32 |
| 0xc | C0 | System Timer Compare 0 | 32 |
| 0x10 | C1 | System Timer Compare 1 | 32 |
| 0x14 | C2 | System Timer Compare 2 | 32 |
| 0x18 | C3 | System Timer Compare 3 | 32 |

BCM2837-ARM-Peripherals.-.Revised.-.V2-1 **p. 172** - (BCM2837-ARM-Peripherals.-.Revised.-.V2-1.pdf)

## BARE METAL PROGRAMMING - ARM ASSEMBLY - EXAMPLE 02-1

- Blinking LED using the System Timer
- The address of the word containing the lower 32 bits of the system clock value is defined as the symbol **SYSTIMER_CLO**

```
/* Example bm01: Blinking LED
 *
 * Target: Raspberry Pi 2/3
 *
 */

/* Pi 2/3
 * GPIO register addresses
 */
GPFSEL2=0x3F200008
GPSET0 =0x3F20001C
GPCLR0 =0x3F200028
SYSTIMER_CLO =0x3F003004

```

```
.global _start
_start:
    mov sp, #0x8000000
    push {ip, lr}

    bl led_pin_enable

1:  bl led_on
    mov r0, #0x00400000 /* duration=64*65536 us = ~ 4s */
    bl delay_us
    bl led_off
    mov r0, #0x00100000 /* duration=16*65536 us = ~ 1s */
    bl delay_us
    b 1b

/* led_pin_enable
 * args: none
 */
led_pin_enable:
    ldr r0,=GPFSEL2
    ldr r0,[r0]
    bic r1, r0, #0x38000000
    orr r1, r1, #0x08000000
    ldr r0,=GPFSEL2
    str r1, [r0]
    bx lr

/* led_on
 *
 * args: none
 */
led_on:
    ldr r0,=GPSET0
    mov r1, #0x20000000   /* GPIO pin 29 on (LED ON) */
    str r1, [r0]
    bx lr

```

ARM-Assembly/4-Bare metal/RPi3Bp/bm-02-blinking_led-1/blinking_led.s

## BARE METAL PROGRAMMING – ARM ASSEMBLY – EXAMPLE 02-1

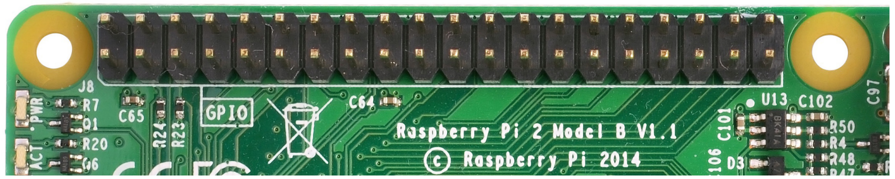
- Blinking LED using the System Timer
- In the new **`delay_us`**:
    - Register **`r4`** is saved as it is used to hold the lower 32 bits of the start time value
    - The full 64-bit time value is read from registers **`CLO-CHI`** with the **`ldrd`** instruction
    - The lower 32 bits of the just read time value are stored into **`r4`**
    - A loop begins:
        - The full 64-bit current time is read into **`r2-r3`**
        - The value in the lower 32 bits of the start time is subtracted from the lower 32 bits of current time to obtain the elapsed time
        - The elapsed time is compared to the delay value
        - If the elapsed time is less than or equal to the delay value, execution restarts from the beginning of the loop

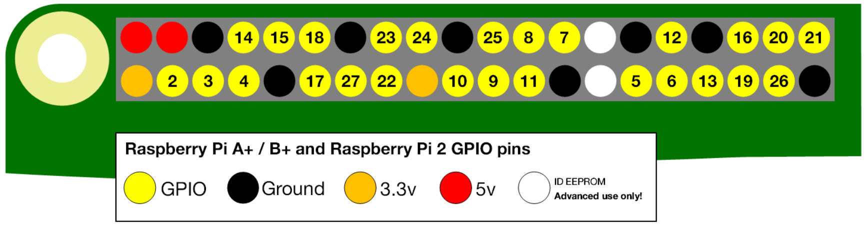
```
/* led_off
 *
 * args: none
 */
led_off:
    ldr r0,=GPCLR0
    mov r1, #0x20000000  /* GPIO pin 29 off (LED OFF) */
    str r1, [r0]
    bx lr

/*
 * delay_us
 *
 * args (r0) delay (~microseconds)
 */
delay_us:
    delay   .req r0
    time_lo .req r2
    time_hi .req r3
    elapsed_time_lo .req r2
    start_time_lo .req r4
    push {r4, lr}           /* Save r4 along with lr */
    ldr r1,=SYSTIMER_CLO
    ldrd time_lo, time_hi, [r1] /* Read current time (64-bit value) */
    mov start_time_lo, time_lo
1:  ldrd time_lo, time_hi, [r1] /* Read current time (64-bit value) */
    sub elapsed_time_lo, time_lo, start_time_lo
    cmp elapsed_time_lo, delay
    bls 1b
    pop {r4, lr}
    .unreq start_time_lo
    .unreq elapsed_time_lo
    .unreq delay
    .unreq time_hi
    .unreq time_lo
    bx lr
```

ARM-Assembly/4-Bare metal/RPi3Bp/bm-02-blinking_led-1/blinking_led.s

## RASPBERRY PI 3 B+ & 4 B+ GPIO CONNECTOR

- Voltages
  - Two 5V pins and two 3V3 pins are present on the board, as well as a number of ground pins (0V), which are not configurable. The remaining pins are all general purpose 3V3 pins, meaning outputs are set to 3V3 and inputs are 3V3-tolerant.
- Outputs
  - A GPIO pin designated as an output pin can be set to high (3V3) or low (0V).
- Inputs
  - A GPIO pin designated as an input pin can be read as high (3V3) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

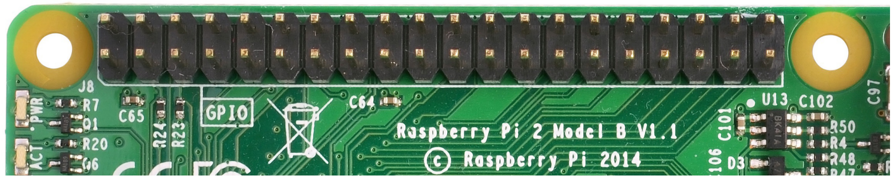


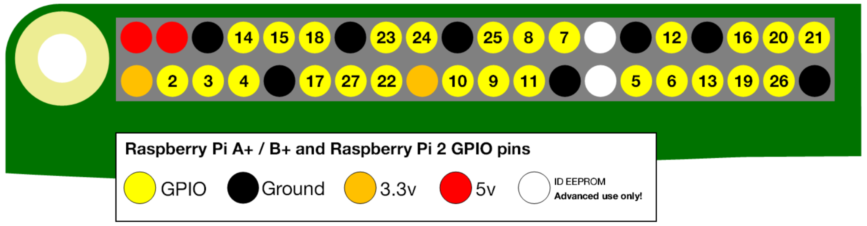


https://www.raspberrypi.org/documentation/usage/gpio/

## RASPBERRY PI 3 B+ & 4 B+ GPIO CONNECTOR

- PWM (pulse-width modulation)
  - Software PWM available on all pins
  - Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19
- SPI
  - SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
  - SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)
- I2C
  - Data: (GPIO2); Clock (GPIO3)
  - EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)
    - A boot ROM can be connected to these pins (white colored in the drawing)
- Serial (UART)
  - TX (GPIO14); RX (GPIO15)

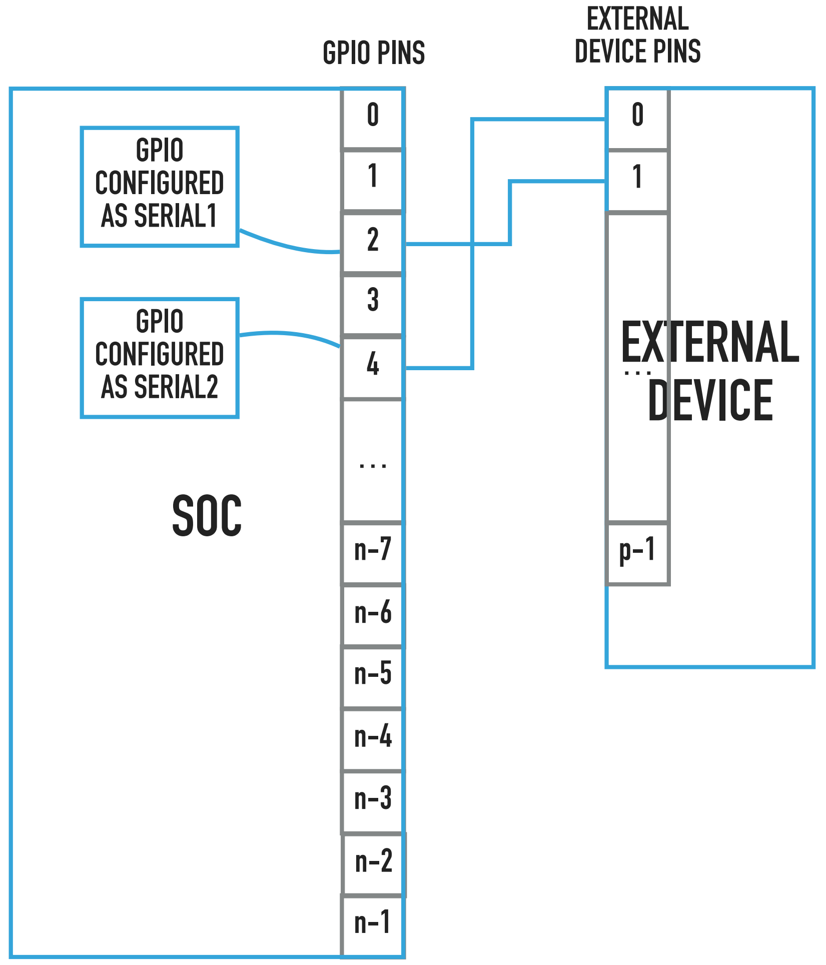




https://www.raspberrypi.org/documentation/usage/gpio/

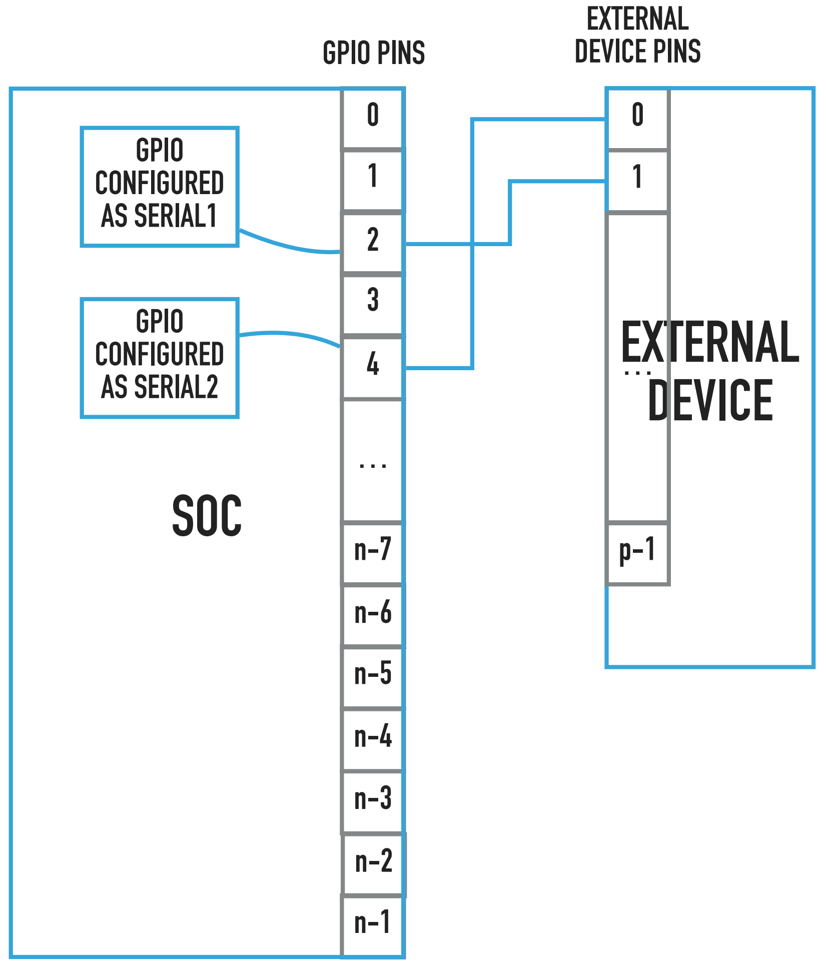
## SERIAL INTERCONNECTIONS

- Serial links permit to connect the CPU to other devices, either on-board or off-board, with a few signals, even only two
- Data is sent one bit at a time over the serial lines from a transmitter (TX) to a receiver (RX)
  - Serial links are cheaper than parallel buses but slower at the same clock rate
    - However, the speed reachable with serial link is enough for many applications that are constrained by external limiting factors (e.g. the maximum bit rate of a radio link or the reading speed of a memory card)



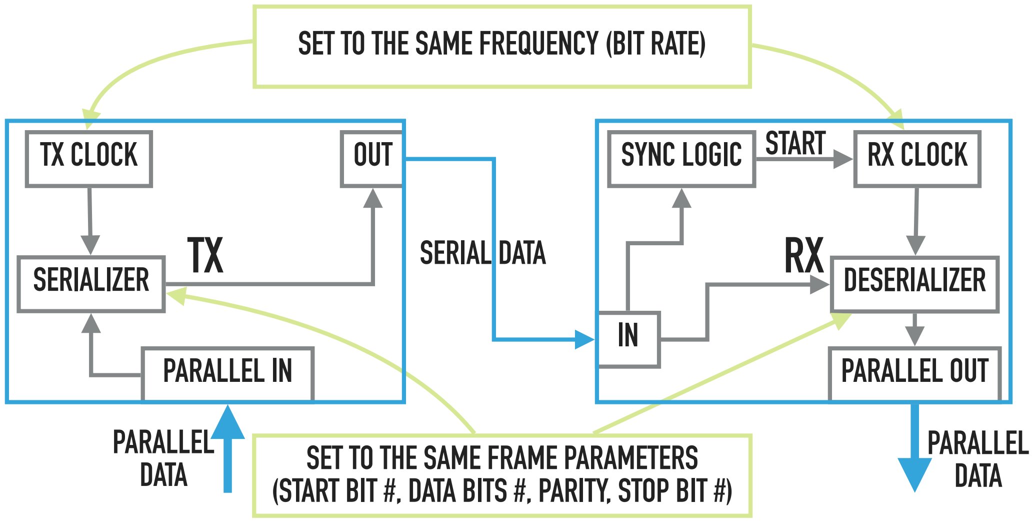
## SERIAL INTERCONNECTIONS

- Usually serialized bits are organized in sequences called **frames** (even a single byte)
  - Additional bits can be included in the frame for error detection and correction
  - Framing simplifies synchronization between sender and receiver, data reconstruction, and transmission error detection
- A few control signals can be also present
- Internal peripherals can ease the work of the CPU by translating parallel data reads and writes into serial ones,  assembling and dissembling frames, detecting and correcting errors and controlling data flow

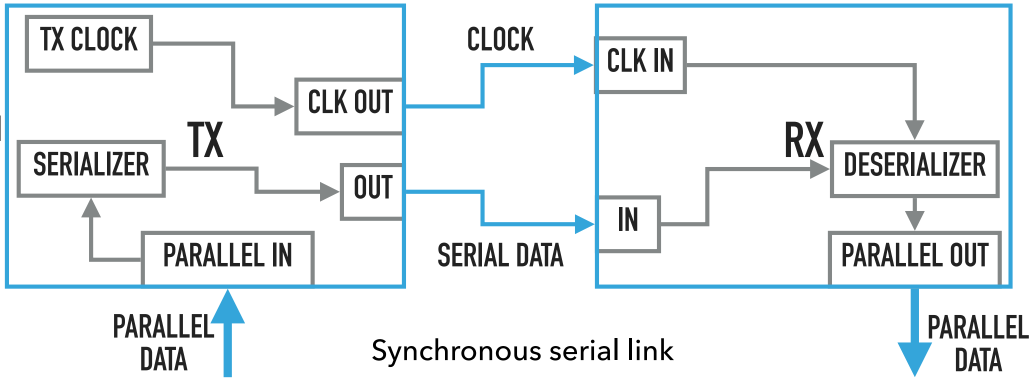


## SERIAL INTERCONNECTIONS

- Serial transmissions can be
  - Asynchronous
    - Data transmission can start anytime
    - TX and RX use each one its own clock, preconfigured to tick at the same rate, to transmit and receive data
    - The frame size is known to both TX and RX , and so only synchronization at the start of transmission is required and has to be provided by the transmitted bits
  - Synchronous
    - Data transmission is synchronized to a clock signal shared by TX and RX
    - There is no need for sync bits, but the clock signal requires another transmission line
  - Peripherals may support both transmission types



Asynchronous serial link



Synchronous serial link