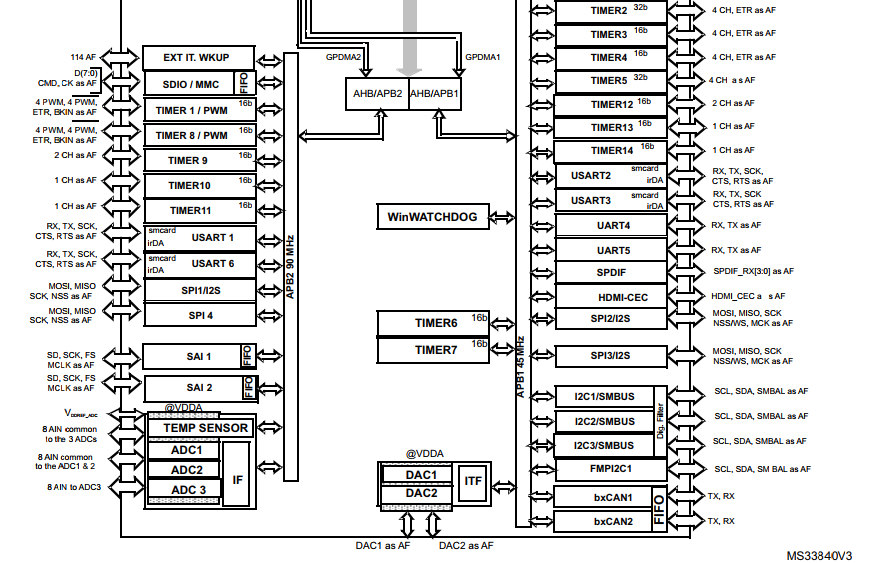
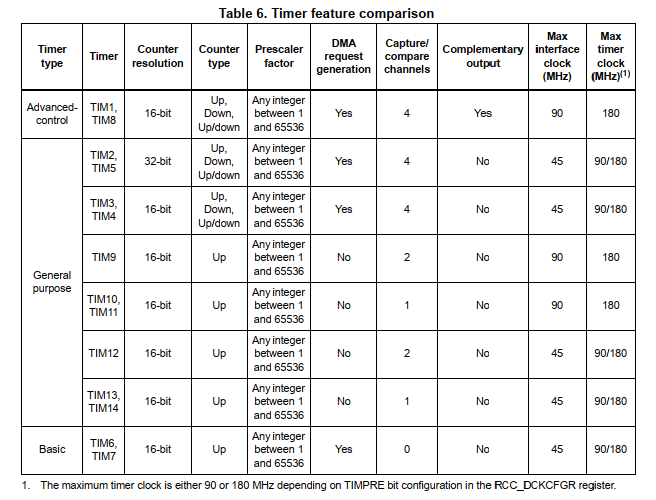
Interface with the termomenter

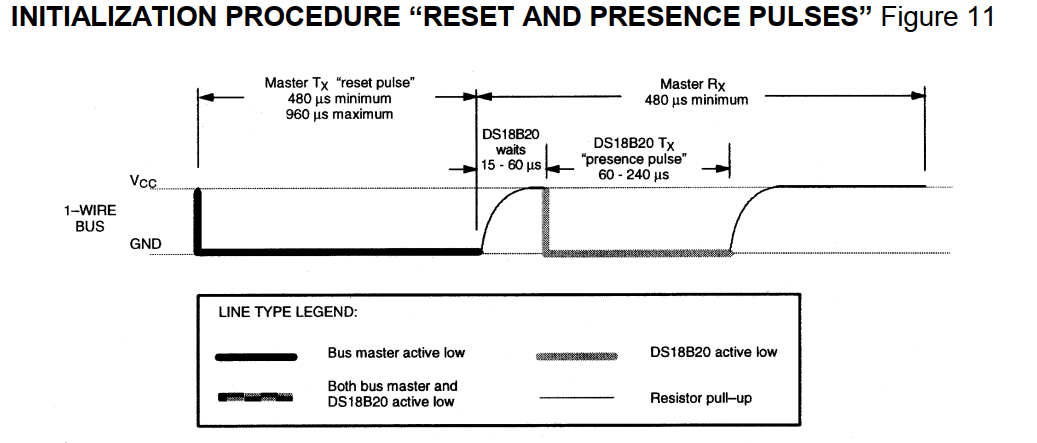
## Peripherals

General peripheral block diagram from pp. 16 in the general overview reference.

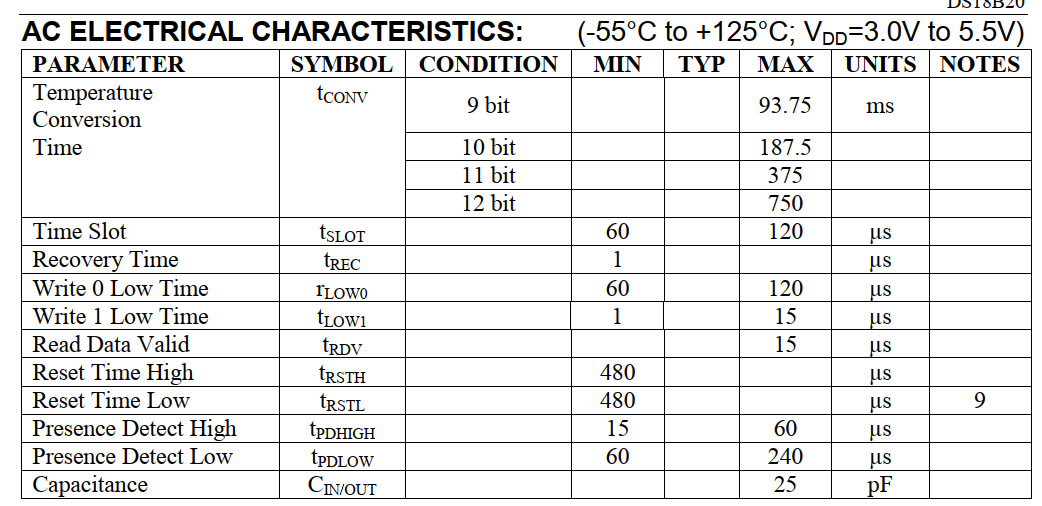




The following is the initialization procedure.

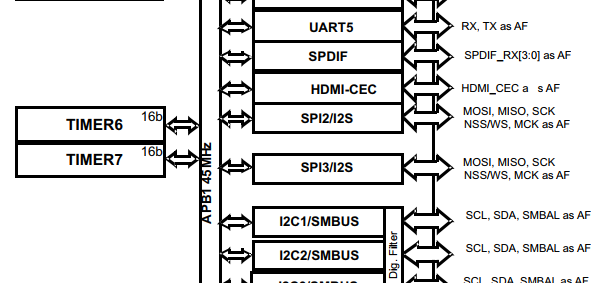


These are the times for the interface, p25.

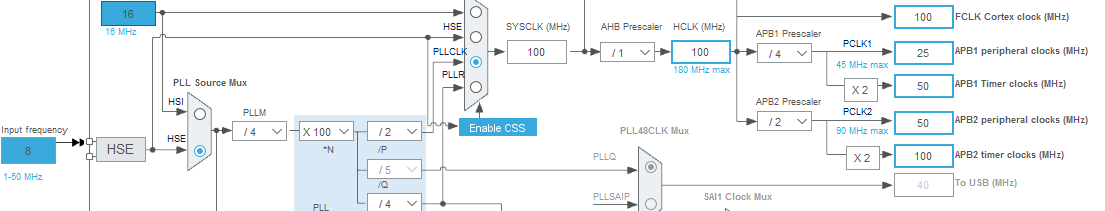


## Setting up the timers

There are two main buses with a base clock depending on which timer/counter or peripheral is to be used, namely APB1 and APB2 Buses. The next image is part of the whole diagram in p16 of the MCU Datasheet:



From it, it can be concluded that Timer 6 and Timer 7 are related to APB1, so they are implemented to be the first timers to work with during the implementation of the 1Wire communication interface. Since the time base is 1 us, then that lead us to a Timer/Counter of frequency 1MHz.



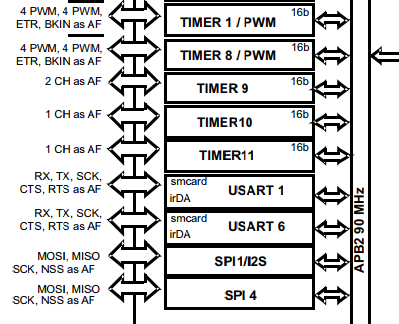
By disabling the USB feature the 100 MHz clock in the timer bus can be achieved for the APB2. That means that there will be a 10 ns base time achievable on the timers 10 and 11.

## The General Purpose IO Pins setup

When the I/O port is programmed as output:

The output buffer is enabled:

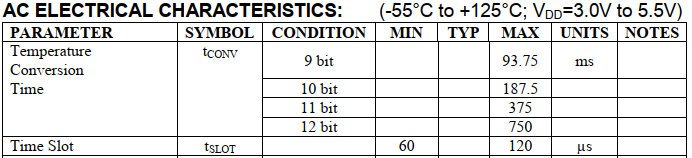
* Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
* Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS



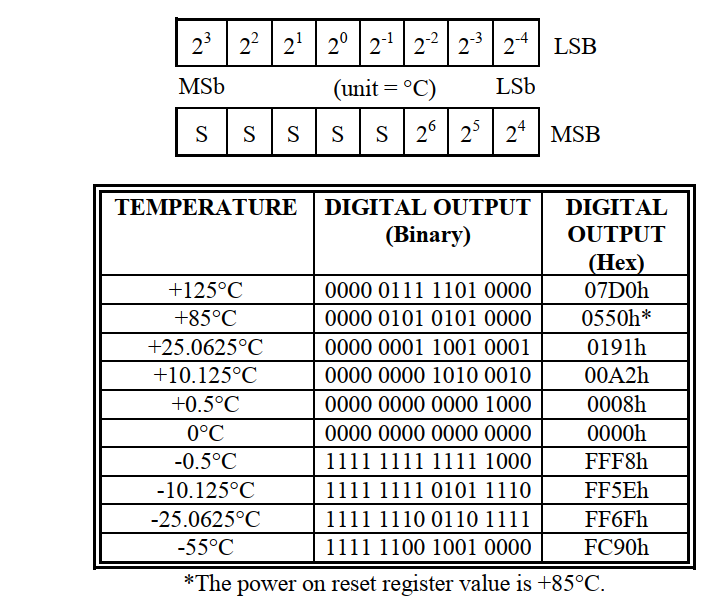
Memory map of DS18B20

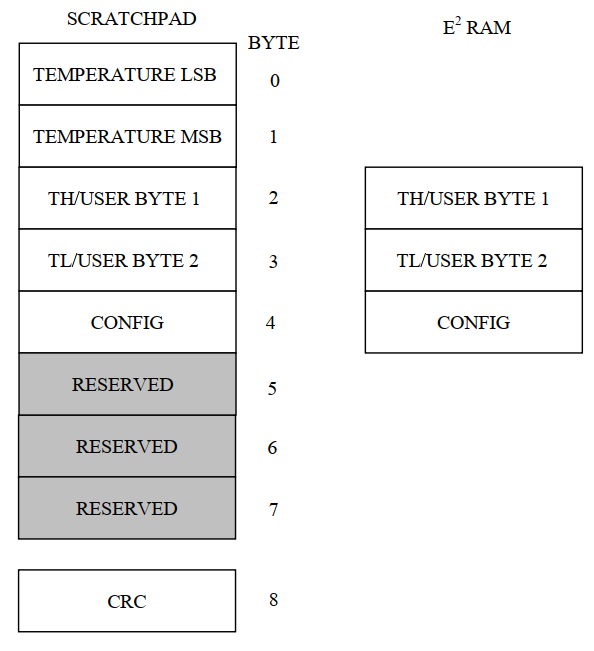
Due to its distribution the LSB is sent first followed by MSB.

The factory default sets the sensor to work with 12bit convertions of 750 ms.



The temperature is converted into centigrade degrees and a division factor of 16 must be used.





Main features that help the programming work:

The GPIO speed for the pin as an output is set to Low Frequency. The pull-up feature is enabled while the pin act as input. During the start cycle the Pin should be set first as an output. The same function should return a signed value to feedback the presence of the sensor. During the write routine, at the beginning, the pin should also be set as an output. The if condition was working until the comparison of results is shifted again to the right. During the read routin there was a delay missing just after release the bus and reading out the sensor.

**#define** GPIO\_SPEED\_FREQ\_LOW 0x00000000U /\*!< IO works at 2 MHz, please refer to the product datasheet \*/

**#define** GPIO\_SPEED\_FREQ\_MEDIUM 0x00000001U /\*!< range 12,5 MHz to 50 MHz, please refer to the product datasheet \*/

**#define** GPIO\_SPEED\_FREQ\_HIGH 0x00000002U /\*!< range 25 MHz to 100 MHz, please refer to the product datasheet \*/

**#define** GPIO\_SPEED\_FREQ\_VERY\_HIGH 0x00000003U /\*!< range 50 MHz to 200 MHz, please refer to the product datasheet \*/

## LED ring control

The main information about the hardware features within the MCU F446ZE are included in the DM00135183 Reference Manual, pp 203.

The control of the LED Ring relays mainly on the right implementation of TIMER with PWM output pulse, another Timer to adjust the Duty Cycle and the DMA controller.

The MCU has integrated 2 DMA controllers. Each of them can handle of to 8 streams which can handle in turn up to 8 requests from peripherals. Hence, there is 128 possible requests available.

\*Probably the direct mode will be implemented since it ensures one 32-length FIFO data will be transmitted. Does that mean that for each bit contained in the 24-bit-long data for each LED will need a whole 32-bit-long FIFO data?

\*DMA Flower controller is software configurable up to 2^16 data items

Four-word length 32 FIFO memory items. They can be selected to be ½ until ¾ of the length, therefore, a ¾ is selected (24bit-long).

Depending on the size of the destination location, only the FIFO mode can pack and unpack only the necessary information to optimize the bandwidth.

