Portafolio

Embedded Computational Systems

10TH QUARTER

STM32 Course Portfolio

Adrian Felipe Gongora Suarez

03/01/2023 – 03/02/2023

Index

Index 2

C Language review topics. 3

Data Types: 6

Storage Classes 6

Functions 7

Data Type Manipulation 7

Pointers 7

Decision Making 7

Looping 7

Arrays 7

Strings 7

Macros ..... **¡Error! Marcador no definido.**

STM32F446ZE Information 12

General information 12

Specifications 12

Processor 13

I/O 13

Block Diagram 14

Memory Map 17

References ..... **¡Error! Marcador no definido.**

Glossary

ARM

ARM or Advance RISC machine is an RISC architecture (Reduced Instruction Set Computer) of 32 Bits, the main design principle is that ARM processors require the least amount of transistors possible.

SWD

SWD (Serial Wire Debug) is a protocol designed for ARM for programing / debugging.

SWDIO: Input / Output Data (Bidirectional)

SWCLK: Clock signal.

ITM

ITM Instrumentation Trace Macrocell, is used to obtain the full instruction trace over a high-speed parallel bus.

https://community.silabs.com/s/article/etm-and-itm-swo-trace-in-cortex-m3-and-cortex-m4-efm32-and-efr32-x?language=en_US

JTAG

Joint Test Action Group (JTAG) is a legacy protocol used for testing chips; this uses 20 pins.

TDI: Input data

TDO: Output data

TCK: Clock signal

TMS: Test selection mode.

TRST: Test reset, reset pin.

FIFO

Fist In, First Out, is a concept used in data structure and accounting, and queue theory. This can be implemented with vectors, using pointers and dynamic memory assignment. In the area of electronics, FIFO is used in a group of pointers for write/read, storage or control logic. Like an SRAM.

More info: https://es.wikipedia.org/wiki/First_in,_first_out#:~:text=FIFO%20(Primero%20En%20Entrar%2C%20Primero,caducidad%20o%20pueda%20quedar%20obsoleta.

FPU

Floating Point Unit is mathematical coprocessor, that is part of a CPU and it's specialized in the calculation of Floating Point units, it can do sum and multiplication.

Floting Point

Floating Point (coma flotante), is a form of mathematical notation, used in computers, that can represent extremely big real numbers and small ones, in a more efficient and compact way.

$$r = c \cdot b^e$$

c = Coeficiente a real number with a single Digit follow by a coma with fractionary digits.

b = Base in which the number needs to be represented.

e = the exponent in which we elevate the base in power.

https://es.wikipedia.org/wiki/Coma_flotante

ART Accelerator

Adaptive Real-Time memory Accelerator (ART Accelerator) is a memory accelator optimized for STM32 ARM Cortex M4 with FPU processors, it uses the inherent performance advantage of the ARM processors over the Flash memory, it implements an instruction prefetch queue and branch cache, which increases program execution speed, the performance achieved means that the wait state program execution from Flash memory at a CPU frequency up to 180MHz is 0.

SRAM

SRAM (Static Random Access Memory) is a type of RAM that uses Flip-Flops to store each bit, the SRAM is a volatile memory, the characteristic of SRAM is that it will hold data permanently in the presence of power, contrary to DRAM which decay in seconds. SRAM is faster than DRAM but its more expensive, SRAM is usually used for the CACHE of a CPU of its internal registers.

https://es.wikipedia.org/wiki/SRAM

FLASH

Flash memory is electronic non-volatile computer memory storage medium, it can be electrically erased and reprogrammed, there are 2 types of FLASH memory NOR flash and NAND FLASH.

RISC

Reduced Instruction Set Computer is a computer architecture designed to simplify the individual computer and get in to more contact with the direct computer, this means that it can result in more code, in order to accomplish a task, but this result in a more efficient, direct and cheaper operation.

RTC

RTC (Real time Clock) is an independent BCD timer/counter, it has dedicated registers that contain the second, minute, hour, etc. in BCD (Binary Coded Decimal) format with correction for 28,29 (leap year), 30 and 31 day of the month, which performs calculation to correctly follow the pass of time, the RTC provides a programable alarm and programable periodic interrupts with wakeup from stop and standby by modes. It uses a 32.768 kMHz external crystal.

RTOS

Real Time Operating System is a lightweight operating system used to facilitate multitasking and integration is resource management for time limited systems. The Real time term indicates the determinism in the extreme raw time execution.

https://www.digikey.com.mx/en/articles/real-time-operating-systems-and-their-applications

AHB

AHB (Advance High Performance Bus) is part of AMBA (Advance Microcontroller Bus Architecture) which is a set of interconnect specifications for ARM, that sets protocols for effective communication on-chip.

AHB is used to interface to any peripherals which are low bandwidth and not require high performance of a pipeline bus interface. Making it a NON-PIPILINED protocol, its used to read or write from a bridge to a master to a number of slaves though a shared bus, it does not support burst transfer.

Full-duplex parallel communication

https://www.linkedin.com/pulse/ahb-apb-amba-protocols-udit-gupta/

APB

Advance Peripheral Bus is used for High-Frequency Design, and it supports multiple bus master, burst transfer and pipelined operations, its common that AHB slaves are internal memory devices, external memory interfaces, and high bandwidth peripherals.

Massive memory I/O Accesses.

https://www.linkedin.com/pulse/ahb-apb-amba-protocols-udit-gupta/

C Language review topics.

Data Types:

- **Antennas:** Satellite antenna systems are used to receive and transmit signals to and from Earth.

Storage Classes

C storage classes are used to describe features of a variable/function, these are included in scope, visibility and lifetime which help to trace the existence of a particular variable during runtime of a program.

- **Auto:** This is the default storage class for all variables declared inside a function or a block. Hence the keyword is rarely used in C.
- **Extern:** Storage class that tells us that the variable is defined elsewhere (not in the same scope / block of code), useful to indicate that we are not initiating new variable but accessing a previous one.
- **Static:** This is a storage class used to declare static variables, which have the property of preserving their value, even when outside their scope, they are only initialized one time and preserved until the end of the program.
- **Register:** This storage class is used to declare register variables, it has the same functionality as that of the auto variable, but this makes the compiler try to store these variables in the register of the microprocessors if a free register is available. This makes the use of register variables to be much faster than that of the variables stored in memory during the runtime of the program.

Functions

Data Type Manipulation

Pointers

A pointer is a point in memory is used to store memory addresses, of variables, functions, and even other pointers, the use of pointers allows low-level memory access, dynamic memory allocation and other functionalities of C.

Syntax = datatype * ptr;

Where ptr is the name of the pointer, datatype is the type of data that is pointing to.

Decision Making

Looping

Arrays

Strings

Struct

A Structure by definition is a group of several variables into one place, each variable in a structure is known as a **member** of a structure, like an array, but unlike an array a structure can contain many different data types (float, ptr, char, etc.).

A structure can be created using the keyword `struct` and declare each of the members inside the braces of the struct.

Syntax:

```
{
    struct 01StructuresExercise1 // Structure declaration
    { // Start of structure
        int aNumber;       //Member (int)
        char aLetter;
        float aFloatingPointNum;
        double aDouble; // Member (double)
        int * aNumber; //Member (ptr)
    }; // End of structure
```

To access a member or in general the structure we need to create a variable of it and use the keyword struct inside the main() followed by the name of the declared structure and the member. Like:

```
struct myStruct01 s1;
```

Which acts as a unique variable related to the struct now with this and using the operator . we can initialize each member in the struct.

```
    s1.aDouble = 12.23;
    s1.aLetter = 'a';
    s1.aFloatingPointNum = 1.32343;
```

Now we can easily create multiple variables for the same structure, making as many variable as you like for one single structure.

```
    s2.aDouble = 5.32;
    s2.aLetter = 'u';
    s2.aFloatingPointNum = 723.455632;
```

Output:

```
                    .\01StructuresExercise1.exe
A double 1.323430
A letter a
A FP num 723.455627
A letter u
PS D:\Repos\STM32_BareMetalCourse_GongoraSuarez\Host\
StructsAndTypeDef>
```

Typedef

Typedef is a type of storage class where the declarator becomes a new type of variable, it can make a variable with a long and difficult to understand name a new short and easier to understand name. Syntaxis:

Typedef char, short, int, singed, etc.

A declaration of a typedef it does not create new variables but it creates a synonym of an existing variable, when you use typedef as a variable type it can be combined but not used with others.

```
typedef existing_data_type new_name;
```

Example 1: A Simple Type Alias.

```
typedef int myInt;  // Now 'myInt' is an alias for 'int'

myInt x = 5;
```

Here the typedef is used a simple alias for the **int** datatype. Now you can use typedef as another data type.

This can be further implemented with an array.

```
typedef int IntArray[5];  // Now 'IntArray' is an alias

IntArray numbers = {1, 2, 3, 4, 5};
```

Now when you type IntArray it refers to a integer array with 5 spaces.

The same can be applied to a function:

```c
typedef int (*Operation)(int, int);  // Now '

int add(int a, int b) {
    return a + b;
}

int main() {
    Operation operation = add;

    printf("Sum: %d\n", operation(5, 3));

    return 0;
}
```

In this case the Operation is now the alias for a function that takes 2 integers and returns an int, this simplifies syntax when declaring function pointers.

Structure Typedef

A typedef applied to an Structure:

```c
typedef struct {
    int x;
    int y;
} Point;

Point p1 = {10, 20};
```

Here the “Point” is an alias for the whole structure that contains int x and int y.

We can see that “typedef” is useful when making the code more readable, specifically in scenarios where complex data types or functions pointers are involved. It enhances the code clarity and reduces the redundancy when using structures, pointers.

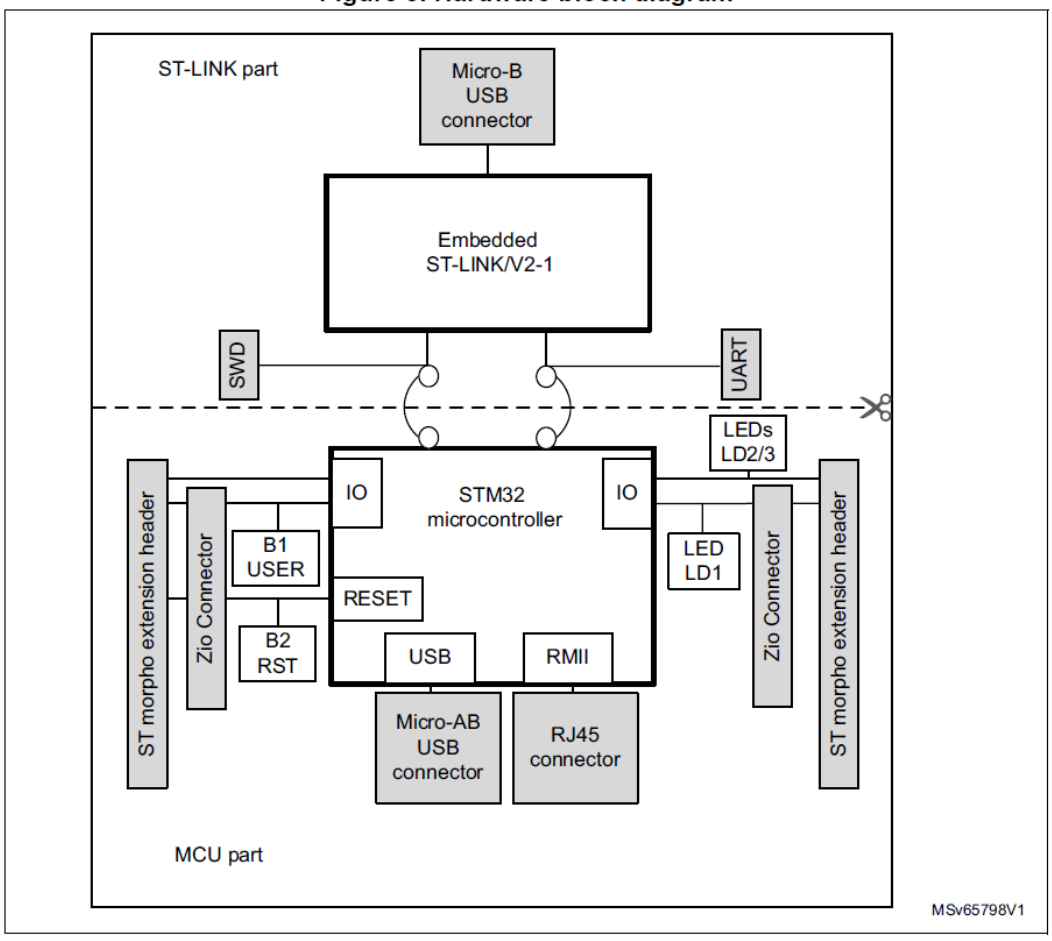
Macros

NUCLEO-STM32F446ZE BOARD Information

The main features of the STM32 Nucleo Board STM32F446ZE is that is has an ARM Cortex core-based microcontroller in an LQFP144 package with 3 user LEDs, 2 user and reset buttons, a 32.768 kHz crystal oscillator, the board has a USB with Micro-AB standard, SWD, ST Zio expansion connector including compatibility with ARDUINO UNO V3

A ST morpho expansion connector, the option for flexible power supply options, VBUS, and On-board ST-LINK/V2-1 debugger / programmer, with USB re-enumeration capability and USB OTG FS on the Micro AB connector, this board has the up to 114 I/O ports with interrupts, 111 Fast I/O up to 90 MHz and up to 112 V-tolerant I/Os.

Figure 3. Hardware block diagram



STM32F446 Microcontroller Information

General information

The STM32F446 is an ARM Cortex M4 32-bit MCU+FPU microcontroller with 225 DMIPS, with up to 512KB of flash/128+4KB of RAM with USB OTG HS/FG, 3 ADCs and 20 communication interfaces, has up to 114 I/O Ports, the processor has a ART accelerator, DSP instructions, Flexible external memory controller with an Up-to 16-bit data bus, and dual memory QuadSPI interface it operates in the -40 to +105 C range temperature from a 1.7 to 3.6V power supply.

Specifications

- 1.7-3.6 Voltage application supply and I/Os, a 4 to 26 MHz crystal oscillator, internal 16 factory trimmed RC, Low Power functions: Sleep, Stop, Standby Modes.
- 3 – 12bit 2.4 MSPS ADC: up to 24 channels and 7.2 MSPS in triple interleaved mode.
- 2 – 12bit D/A converter.
- General Purpose DMA.
- 17 timers: 2x watchdog, 1 SysTickTimer and up to 12 – 16BIT and two 32Bit timers, reaching up to 180 MHz each.
- Debug Mode (SWD, JTAG interfaces, Trace MacroCell).
- 114 I/Os
- SPDIF-RX
- 4x I2C interfaces
- 4x USART and 2x UART (LIN)
- 4x SPI (45 Mbits/s)
- 2x SAI (Serial Audio Interface)
- 2x CAN (2.0B Active)
- SDIO interface
- CEC
- RTC
- CRC

Processor

The processor is an ARM Cortex M4 with FPU and embedded FLASH and SRAM.

Is the latest generation of ARM processors for embedded systems, is a 32-bit RISC processors with a set of DSP instructions for signal processing and complex algorithm execution, a single precision FPU (Floating Point Unit) that speeds up software development using metalanguage development tools.

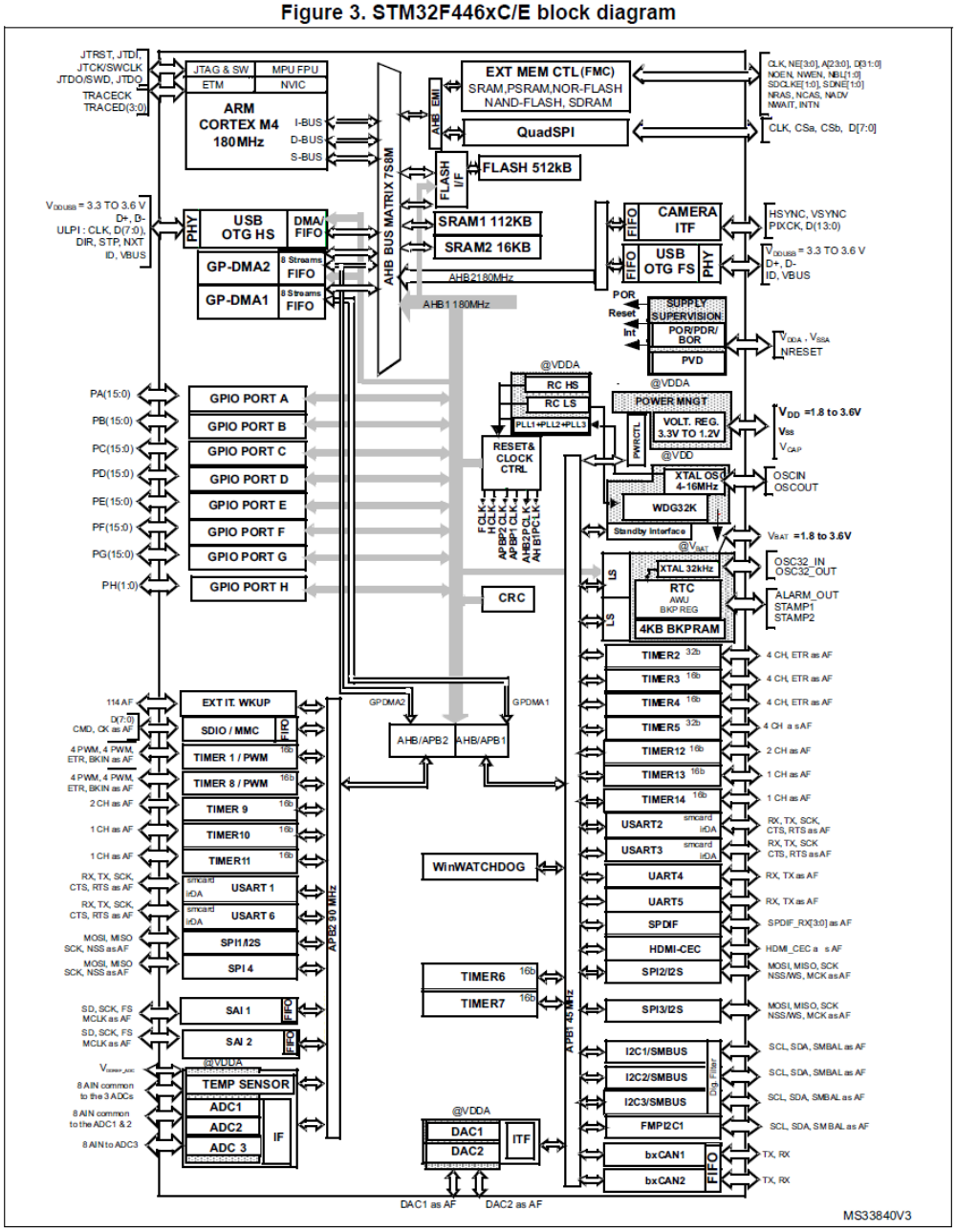
I/O

The I/Os are the GPIO or General-Purpose inputs/outputs these can be configured by software as an output with the configurations: push-pull or open-drain, with or without pull-up or pull-down.

Most of the GPIOs are shared with digital or analog alternate functions like TIMERS, ADCs, Communication interfaces (I2C, CAN, etc.) All the GPIO are high-current capable and have a speed selection. Fast I/O handling allowing maximum I/O toggling up to 90 Mhz.

Block Diagram

Figure 3. STM32F446xC/E block diagram



03/01/2024

The block starts from the top in the ARM CORTEX M4 180Mhz block this is the processor, then the diagram shows the buses that connect to the main bus the (AHB BUS MATRIX 7S8M, this bus is then ramify in to the various components and features that the STM32F466 has, showing how each block ends up with an Output Pin associated to it,

the AHB is connected to the AHB EMI that connects the external memories and QuadSPI, is directly connected to the FLASH I/F that connects the 512kB memory and the 2 SRAM (112 KB, 16KB), we also have the USB OTG HS connected directly with the 2 GP-DMA1, GP-DMA2, the main focus is the AHB BUS MATRIX, which is divided into 2 Buses the AHB1 and AHB2, the AHB2 as the only function of connecting the USB OTG FS and the Camera ITF, the other Bus AHB1 180MHz is one of the most important this bus connects various important components, the main one is that this BUS is the one that distributes the various GPIO Ports from PORT A to PORT H, this bus is also connected to the REST and CLOCK control block, that enables and connects to the XTAL OSC or external oscillator pins (4-16 MHZ), XTAL for 32kMHZ OSC. At this point the block diagram is divided into two different parts, the AHB/APB2 and the AHB/APB1, these operate at a lower frequency, the APB2 operates at 90 MHZ and connects the various communication interfaces like the USART, SPI, SAI, but it also connects the External wake-up the SDIO, the various TIMERS and the ADC and temperature sensor pins. Now the APB1 operates at 45 MHz bus connects the previously mentioned XTAL and RTC components, but also the Power Management block that regulates 3.3V to 1.2V, and enables the VDD, VSS and Vcap pins, this BUS also connects the TIMERS, USART and SPI, SPDIF communication interfaces as well as the most important I2C/ SMBUS and CAN1, CAN2 buses. Finally, this bus connects the DACs (DAC1 & DAC2).

System Bus can operate at the speed up to 180MHZ?

Yes, the STM32F466 can operate at a speed up to 180Mhz, this also applies to another buses via the BUS Matrix.

SRAMS are connected to System Bus T/F?

Yes, using the multi-AHB bus matrix, the SRAMS, FLASH and other buses are connected.

APB1 bus can operate at the speed up to 180MHz?

No, the APB1 operates at a 45MHz in the case of the STM32F446. This speed varies according to the speed of the ARM processor (180Mhz, 168Mhz, 100Mhz, 84Mhz).

Can I connect that peripheral via APB2?

No, if the peripheral operating frequency is above the 90Mhz of the APB2 BUS limit, the peripheral would not work or would work with information lost and errors.

What is the MAX. HCLK value of your MCU?

180Mhz

What is the MAX P1CLK value of your MCU?

90Mhz this is APB1 High-speed bus

What is the MAX P2CLK value of your MCU?

45Mhz on the low-speed APB2

GPIOs and processor communicate over 1AHB bus T/F?

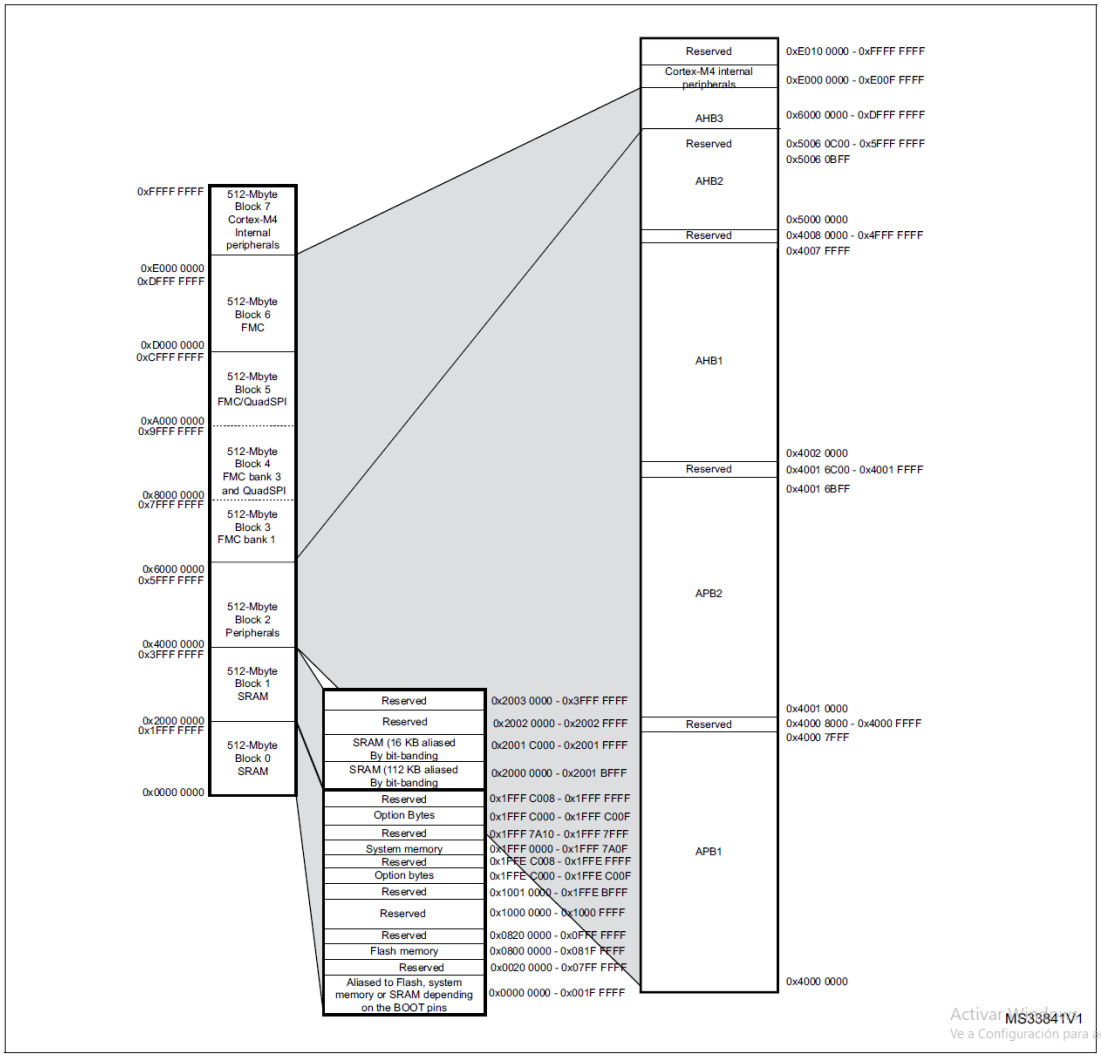
Yes, all of the 8 GPIOs ports and the ARM processors communicate over the AHB1 BUS.

USB OTG and processor communicate over AHB2 bus T\F?

Yes, the USB OTG FS, communicate over AHB2 bus.

Memory Map

Figure 15. Memory map



The memory map is divided into 2 Columns the first column has 8 Blocks of memory, that each encompass a 512-Mbyte Memory Size buffer or block of memory addresses, on the other size we have a column divided into the different Buses (AHB3, AHB2, etc.) and uses of the memory, here a relation as seen with the Memory Blocks being represented in relation to the Buses and uses of those memory blocks, is a way to reference each memory that we use in the firmware.

So, the memory map is a Map for all the addresses in Memory as interpreted and set up in the STM32F466.

Memory Block 7:

The first block from top to bottom is the Block 7 which corresponds to the memory addresses for the internal peripherals of the ARM M4 processor.

Memory Block 6-3:

The next 4 blocks are reserved for external memory described as the FMC and the QuadSPI which are related to the AHB BUS Matrix as seen in the Block diagram.

Memory Block 2:

Memory block 2 has all the addresses for the different peripherals of the Microcontroller, and is linked to the AHB2, AHB1, APB2 and APB1 buses which in the block diagram connect all of the GPIO and all of the communication interfaces: SPI, USART, I2C, CAN, all of the TIMERS and clock related applications.

Memory Block 1:

The Memory block 1 contains all the addresses of the SRAM (112KB / 16KB) that the microcontroller has internally.

Memory Block 0:

Finally, the Memory Block 0, is segmented into the addresses of the system memory, option bytes, the FLASH memory, and several sub-blocks of reserved memory.

What's the base address of AHB1 BUS peripherals?

Is the Memory Block 2 which encompass the address from 0x4000 000 to 0x5FFF FFFF, in the case of AHB1 this is **0x4002 0000 to 0x4007 FFFF**.

What's the base address of RCC engine register of the MCU?

From the Block 2 and the AHB1 Bus the base address is **0x4002 3800 – 0x4002 33FF**

What's the base address of APB1 Peripherals?

0x4000 0000 - 0x4000 7FFF

What's the base address of FLASH memory?

From Block memory 0 = **0x0800 0000 – 0x081F FFFF**

What's the base address of SRAM2?

For the SRAM 16KB is **0x2001C000 – 0x2001 FFFF**

What's the base address of ADC registers?

For the ADC register we need to focus on the Block 2 for peripherals and the APB2 90Mhz Bus for ADC1,2,3 = **0x4001 1800 – 0x4001 23FF**