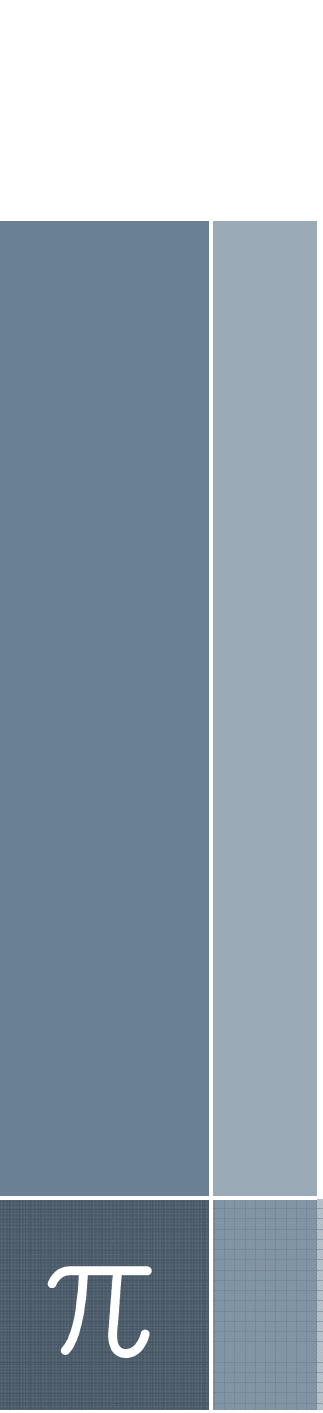
Embedded Thursday

Binary/Hex system and I2C

Ivan Quiroz



Today

- The project Goal and System
- Recap
- I2C

Goal Description

- Learn C as embedded language
- Use C to understand underlying processor
- Have a project so learning stays
  - We are making a self balancing robot

Todays Goal

- I2C bus hardware
- I2C connections
- I2C network

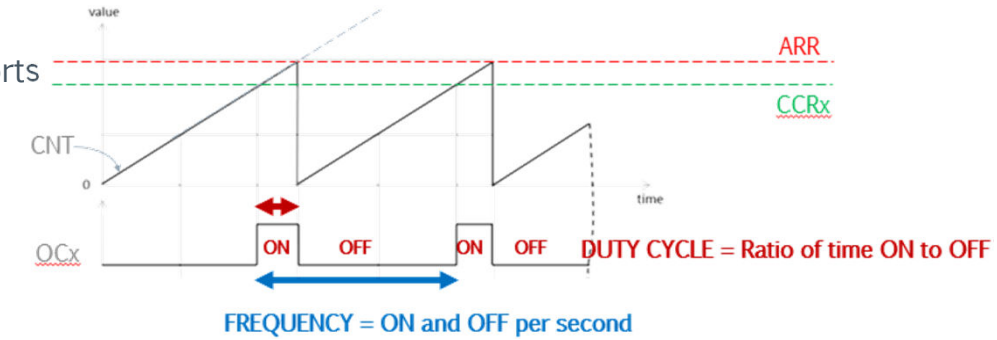
Recap on Timers and Variables

- **`Volatile:`** A variable that may change at any time without any action being taken by the code
- **`ARR`**: Sets the frequency
- **`CCRx`**: Sets the duty cycle
- Selection of proper system timer must be understood
- Enable output of Timer
- Link Timers to outputs
- Enable Alternate Function to ports

```
#include <stdint.h>
```

- `int8_t` • `uint8_t`
- `int16_t` • `uint16_t`
- `int32_t` • `uint32_t`

Data type	Precision	Range
unsigned char	8-bit unsigned	0 to +255
signed char	8-bit signed	-128 to +127
unsigned int	compiler-dependent	
int	compiler-dependent	
unsigned short	16-bit unsigned	0 to +65535
short	16-bit signed	-32768 to +32767
unsigned long	unsigned 32-bit	0 to 4294967295L
long	signed 32-bit	-2147483648L to 2147483647L
float	32-bit float	±10-38 to ±10+38
double	64-bit float	±10-308 to ±10+308

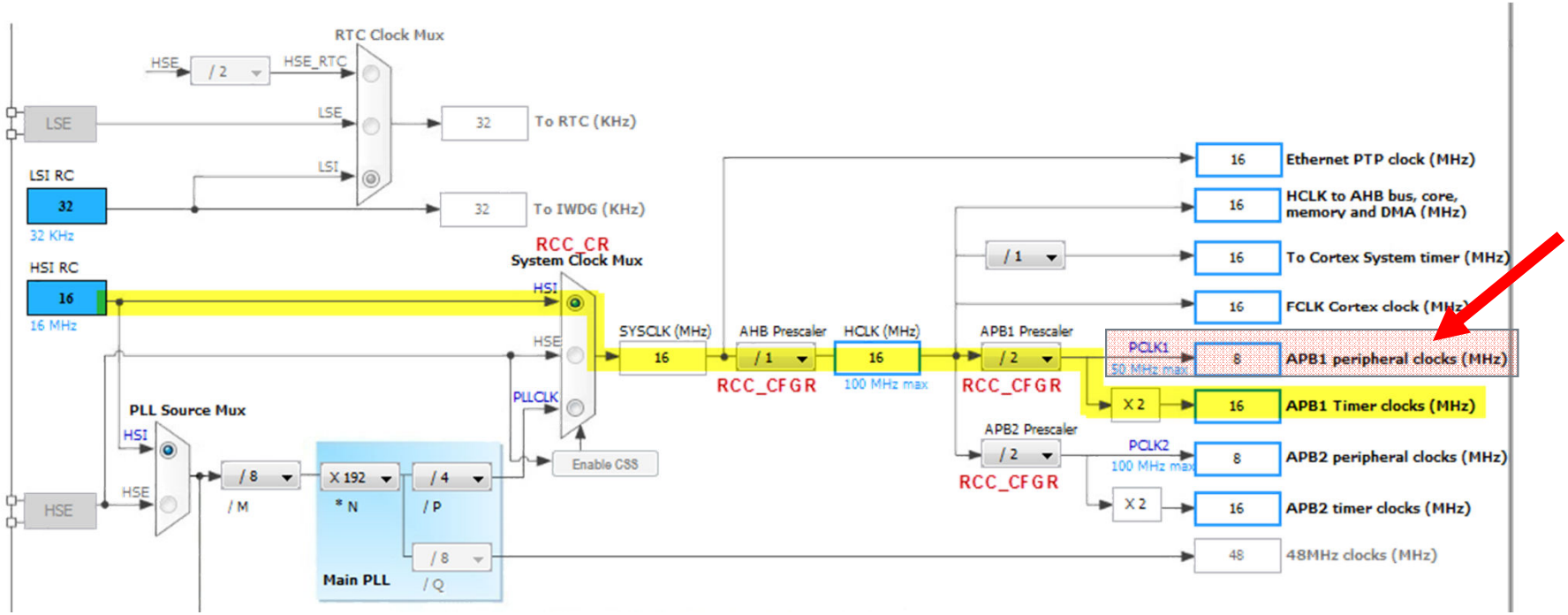


Operators

XOR	`^=`	Use: switch bit to opposite value
OR	`|=`	Use: Impact a bit, don't disturb others by OR'ing desired bit with 0x001
AND	`&=`	Use: Impact a bit, don't disturb others by OR'ing desired bit with 0x110
AND	`&`	Use: Mask a bit with using 0x001

Recap Clock Source

- STM32F411E-DISCO has a 8Mhz input clock
  - SystemInit() sets Clock source as HIS (Internal 16MHz clock)
  - SystemInit() sets APB1 clock for timers = 16MHz



Binary and Hex system

- Binary used at register level to enable functionality based on definition.
  - Each binary position is a bit
- Hex used for short binary representation broken down by 4 bits
  - Each HEX position is 4 bits

2	B	9	F
0010	1011	1001	1111

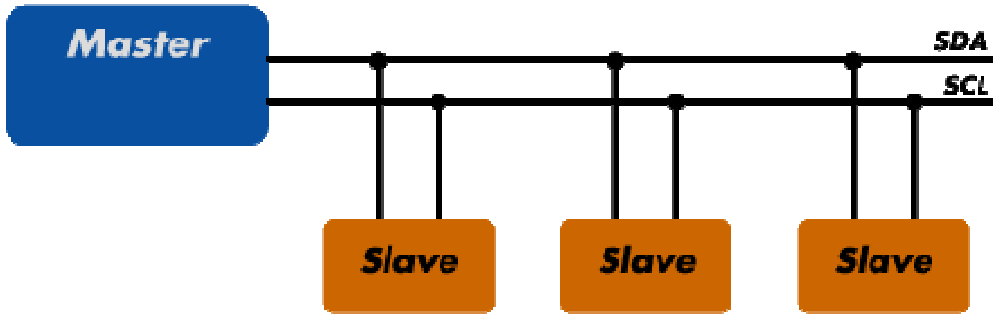
- Quick conversion tip

1	0	1	0		
(8)	(4)	(2)	(1)	→	8+2 = 10 = A

bin		dec		hex
0000	=	0	=	0
0001	=	1	=	1
0010	=	2	=	2
0011	=	3	=	3
0100	=	4	=	4
0101	=	5	=	5
.				
.				
.				
1010	=	10	=	A
1011	=	11	=	B
1100	=	12	=	C
1101	=	13	=	D
1110	=	14	=	E
1111	=	15	=	F

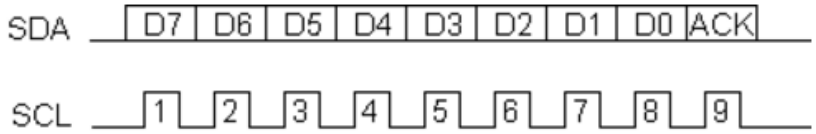
I2C Introduction

- I2C : Inter-Integrated Circuits
  - Not great for out of board communications
  - Normally used between controllers and sensors or EEPROMs
- I2C Environment
  - Most common there is a single master
  - There are multiple slaves
  - It only requires “2 wires” (+Ground)
    - SDA: Serial Data
    - SCL: Serial Clock
  - Signal levels driven by pull ups

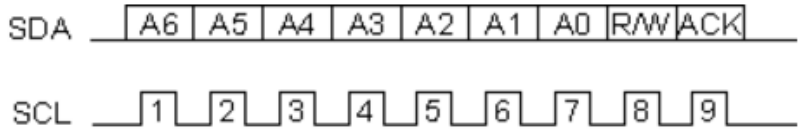


I2C Hardware Layer

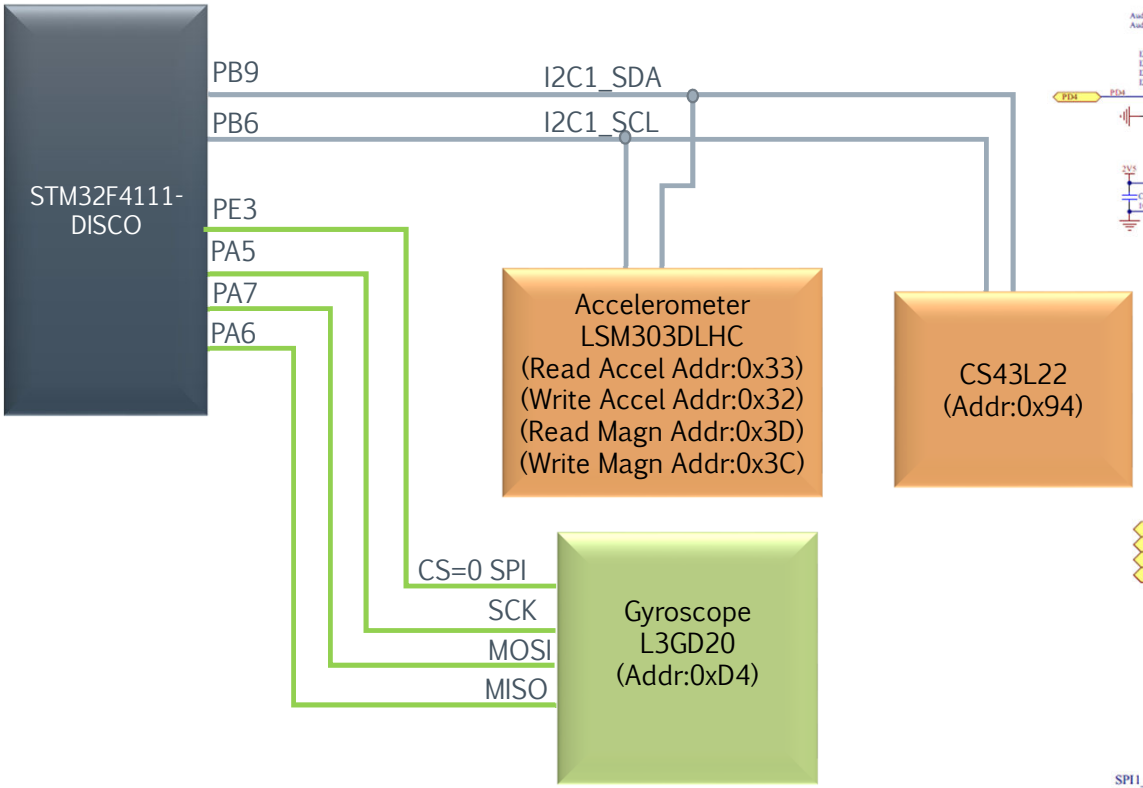
- I2C uses two bi-directional lines (Clock and Data)
  - Single Master controls CLK
  - Lines are open collector, so pull ups are needed
- Standard speeds are 10kHz, 100khz,400kHz (1MHz)
- Data transferred in 8-bit sequences
  - Data

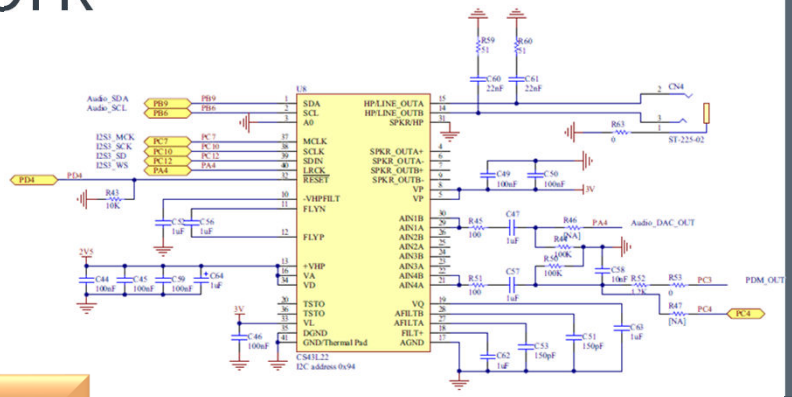


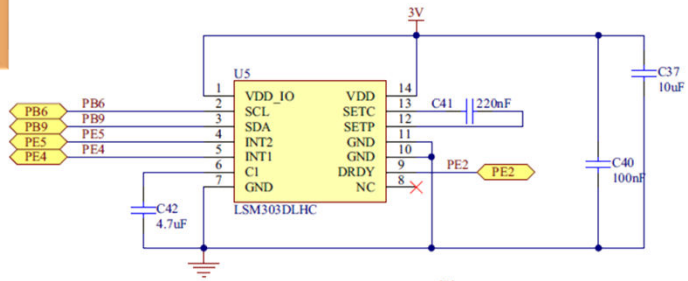
  - Addressing

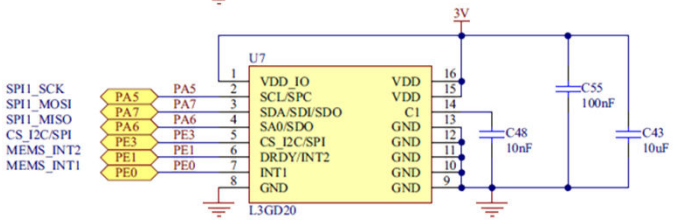


STM32F411-DISCO I2C Network









I2C WRITE SEQUENCE

1. Send START bit
2. Send slave address (7-bit) plus R/**W** bit low (WRITE)
3. Wait for ACK from slave
4. Send register Internal address you want to write to
5. Send the internal register number you want to write to
6. Send the data byte
7. [Optionally, send any further data bytes]
8. Send the STOP bit

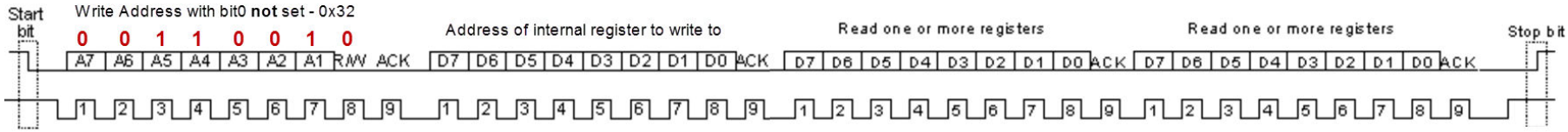
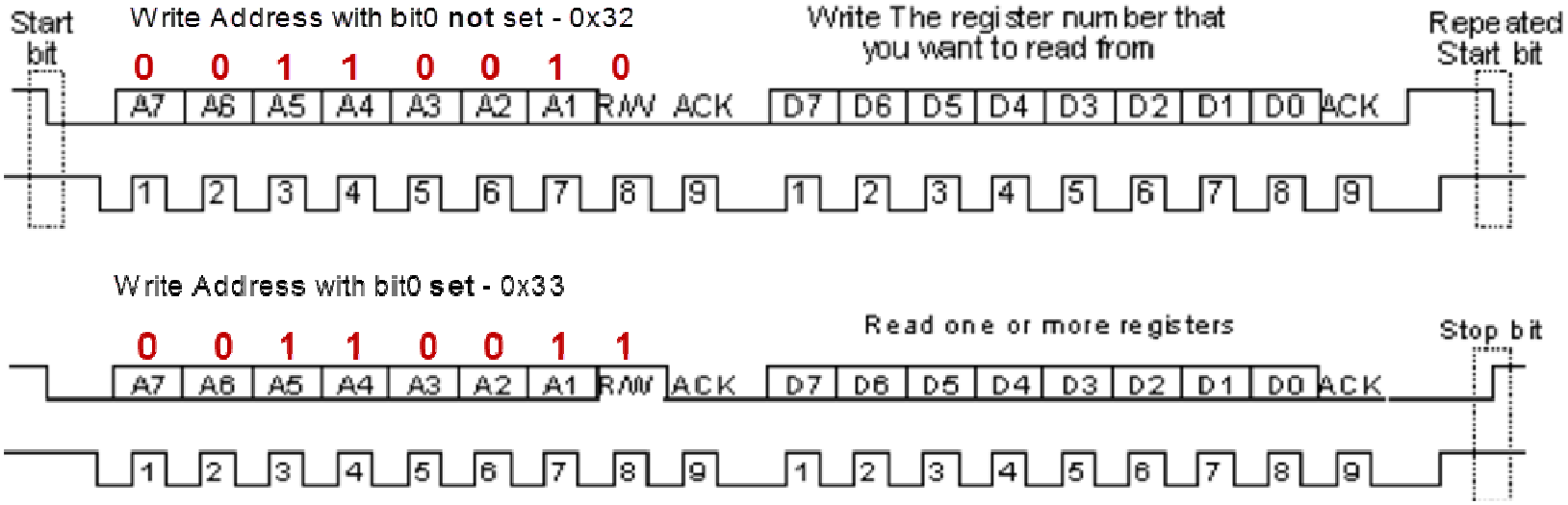


Table 12. Transfer when master is writing multiple bytes to slave:

Master	ST	SAD + W		SUB		DATA		DATA		SP
Slave			SAK		SAK		SAK		SAK	

I2C READ SEQUENCE

1. Send a START bit
2. Send slave address (7-bit) plus R/**W** bit low (WRITE)
3. Wait for ACK from slave
4. Send register Internal address for data needed from slave
5. Send a START bit
6. Send slave address (7-bit) plus **R**/W bit high (READ)
7. Read data byte from slave (3 bytes for accelerometer)
8. Send the STOP bit



STM32F411 I2C Bus Setup

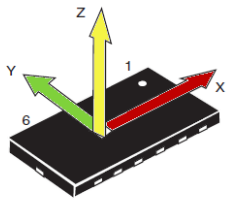
- Set I2C_CR2 register: set up timing
- CCR: Set Clock Control Registers: (generates SCL clock)
- TRISE: Set rise time register
- Set I2C_CR1 register: Enable I2C peripheral

Ready to start communications…

- Set START bit in I2C_CR1 register to generate a START bit

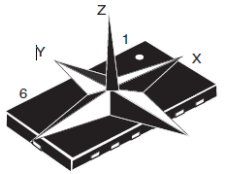
Accelerometer Details

› Per LSM303DLHC datasheet



TOP VIEW

DIRECTION OF DETECTABLE ACCELERATIONS



TOP VIEW

DIRECTION OF DETECTABLE MAGNETIC FIELDS

Table 11. Transfer when master is writing one byte to slave

Master	ST	SAD + W		SUB		DATA		SP
Slave			SAK		SAK		SAK	

Table 12. Transfer when master is writing multiple bytes to slave:

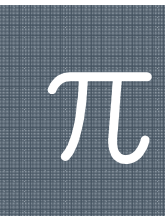
Master	ST	SAD + W		SUB		DATA		DATA		SP
Slave			SAK		SAK		SAK		SAK	

Table 13. Transfer when master is receiving (reading) one byte of data from slave:

Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

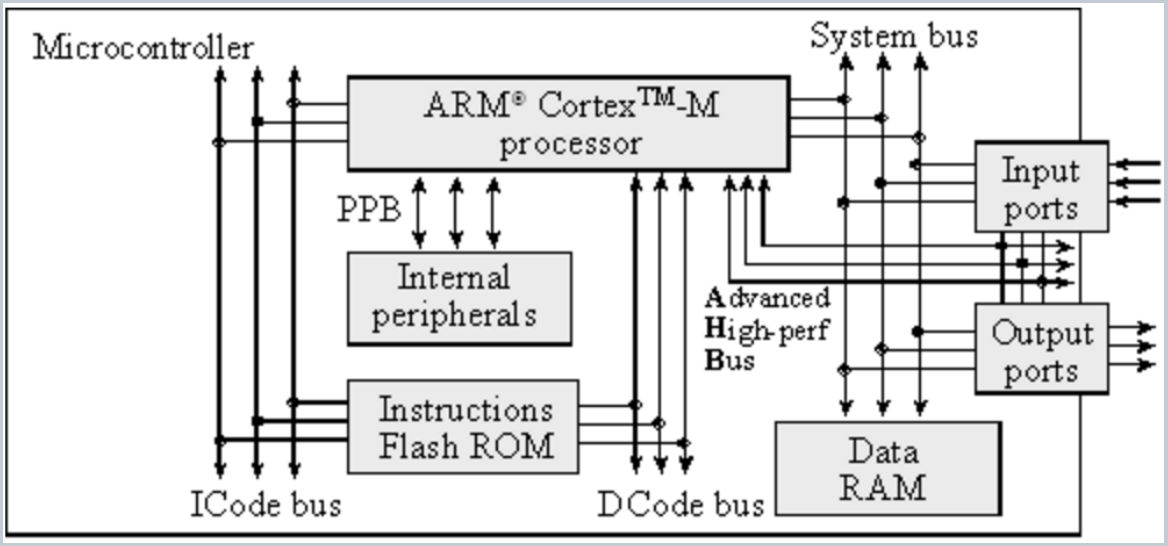
Back Up Slides

Hardware Reference Material



SIMPLIFIED STM34F411 ARCHITECTURE

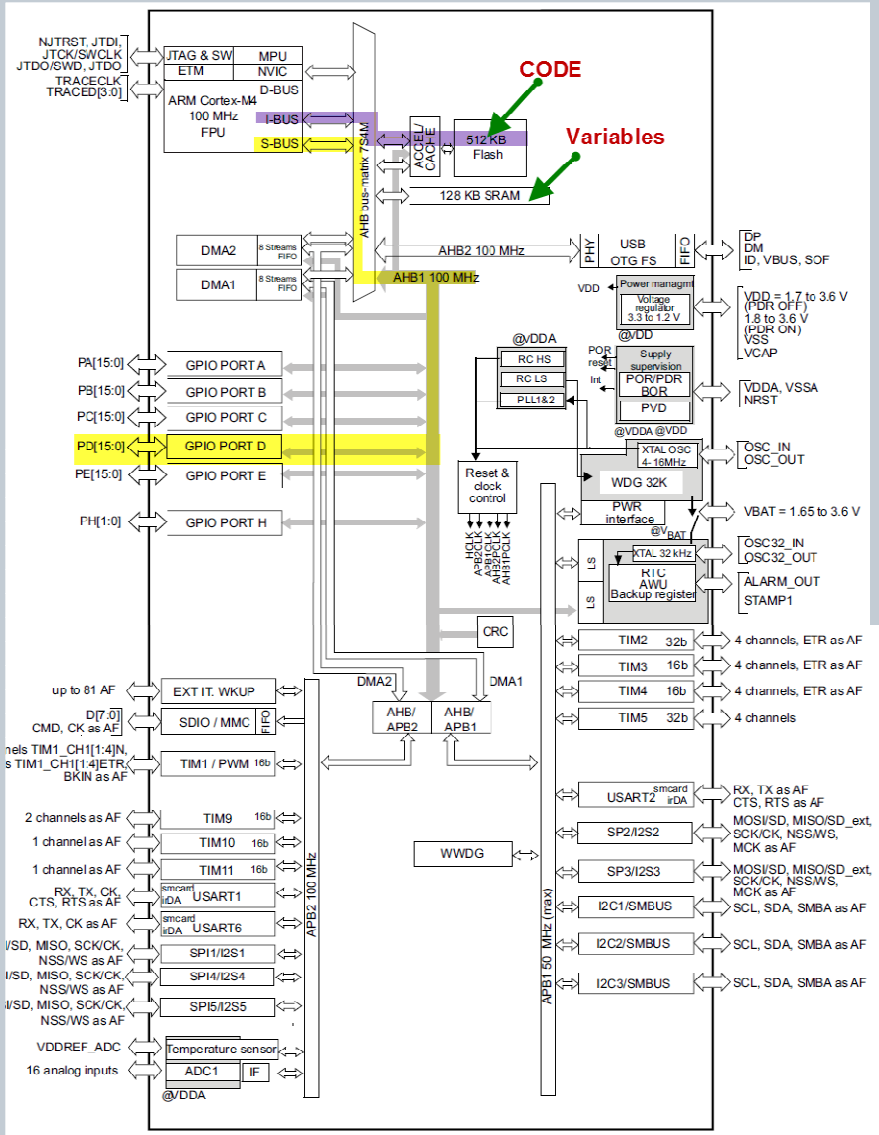
- I-Code Bus use to fetch instructions from Flash ROM
- System Bus: use to work with variables and IO Ports
- D-Code Bus: debug bus
- Adv Hi Bus: Connection to IO ports and dedicated USB ports



STM32F411 BLOCKDIAGRAM

Note the following buses:

- **RCC->AHB1ENR** needed as Port D uses AHB1 (yellow)



PWM DIAGRAM

CLOCK GENERATION

- APB1 clock depends on RCC_CR settings
- APB1 CLK is x2 for TIMx
- PSC can divide by 2
- ARR sets max count
- CCR set max ON time