DATE / / OBJECT

2005 → ARM cortex

cortex (A) → application
Ex → Lab & mobile

cortex (R) → Real Time
Ex → Airbag

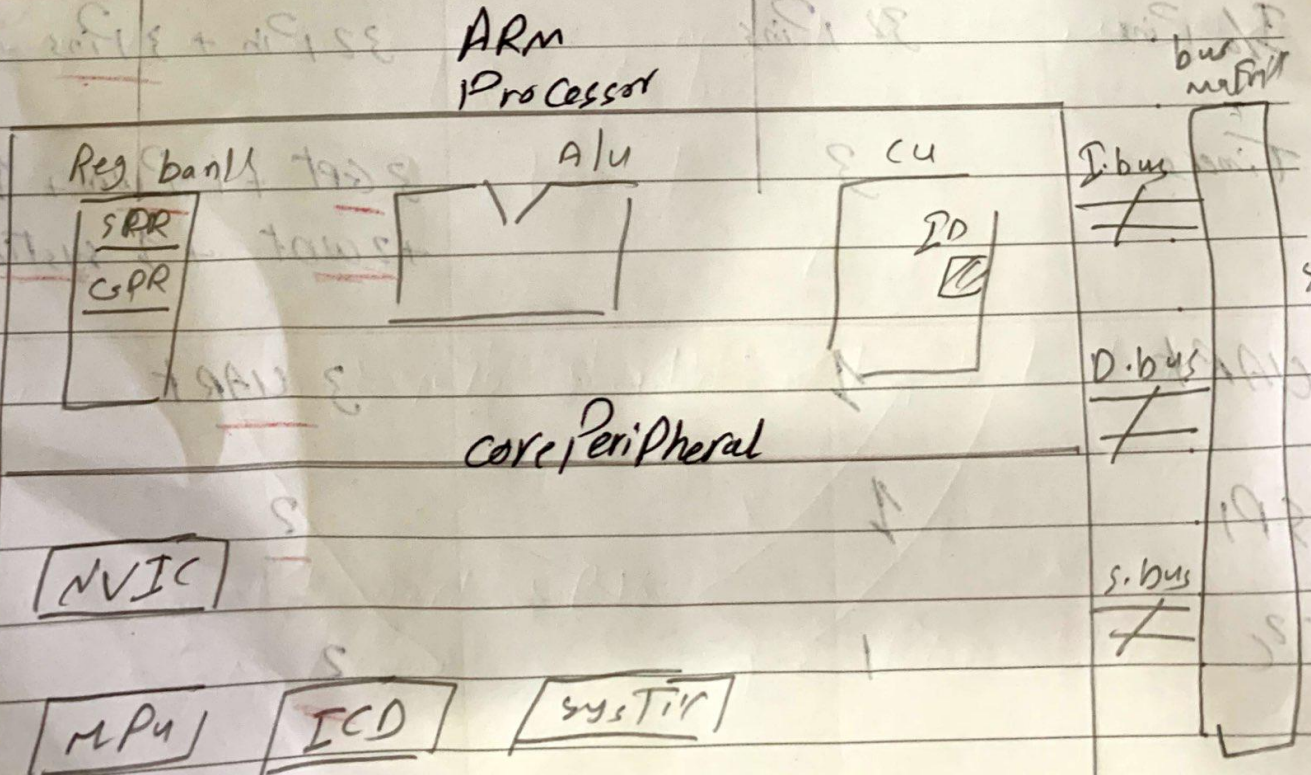
cortex (M) → micro controller → Embedded
ARM cortex M → STM
↳ STM32 F103C8

Compare between atmega32 & STM32 F103 C8

	atmega 32	stm32 f103c
instruction Set	8 bit "word"	32 bit "word"
freq. max	16 MHz	72 mHz
RAM	2 KB	20 KB
I/O Pins	32 Pins	32 Pin + 3 Pins
Timers	3	3 GPT + 1 Pwm + 1 RTC + 2 WDT + 1 systick "inside P
UART	1	3 UART
SPI	1	?
I^2C	1	2

DATE / / OBJECT

ADC	8 channel Resolution 10 bit	10 channel Resolution 12 bit
CAN	X	✓
DMA	X	✓
ICD = in circuit debuger	X	✓
MPU Memory Protection unit	X	✓
Power consumption	5 V	3,3 V
Cost	0.7 $ 3 $ → product and kit Ed	0.7 $ 1,7 $ → for industrial Me



DATE / / OBJECT

*NVIC → Nested vector interrupt control.
*MPU → Memory Protection unit.
*ICD → in circuit debuger.

*I.bus → instruction bus → flash

*D.bus → Data bus → flash

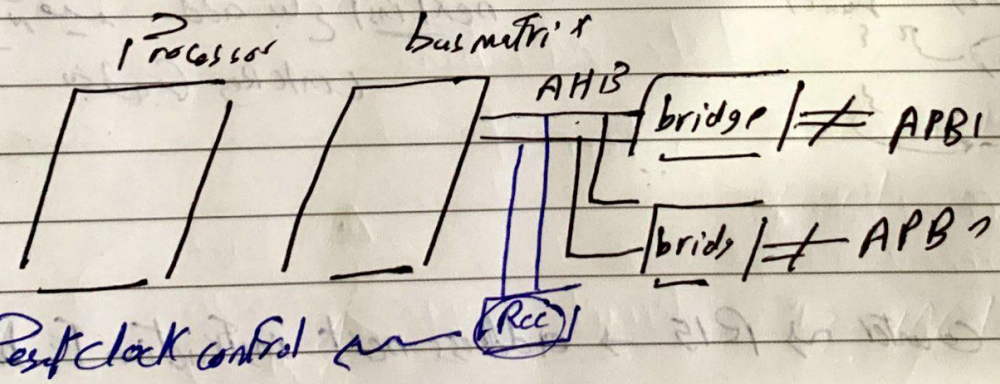
*S. bus → system bus ← SRAM I/O memory

* how To connect other Things outside Processor
→ Serial/Parallel and synchronous/Asynchronous

ARM → AMBA ← AHB → advanced higher Performance ← high Power & Performance Full duplex
APB → a Peripheral Bus → low Power & Performance half duplex

advanced microcontroller Bus Arch

EX



AHB → upto 72 MHZ → full duplex

APB1 → " 36 MHZ → half duplex

APB2 → " 72 MHZ → half duplex

Register Bank

1) GPR → General Purpose Register ⇒ (13) → R_{12}
32 bit

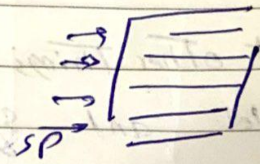
← بنخزن فيها متغيرات بنحتاجها في Run Time

Ex → Z = x + y → In R_0, x
" R_1, y
in R_2, R_0 + R_1
out Z, R_2

2) stack Pointer (SP) → R13

MSP — Main stack Pointer

PSP — Process stack Pointer



RTOS

scheduler | task → memory "TCB + stack"

بيشاور على آخر address في stack تاعي

3) Link Reg → R14

Ex → int x=2; fun(); func() { }

PC → x=1; Z = x+1

← الـ PC بتاعي بنخزن فيه add بتاع next inst بنحطها في LinkReg

4) PC → Programe Counter → R15 → address next instruction to be execute

5) PSW → Program Status Word → R16
(N, Z, C, OV) ← خاص بـ flags

DATE / / OBJECT

6) interrupt mask flag → R17

3 bit

1- PRI mask (1 bit) → 0

↓1 → disabled all interrupt except non maskable interrupt "NMI" ⇝ Reset OR hard fault

1/0 (logic)

2- fault mask (1 bit)

↳ disabled all interrupt + hard fault except (NMI)

3) - Base PRI (1 bit) → disabled all interrupt of lower Priority level.

Priority

7) Control Reg ⇝

Control [1] ⇝ stack status

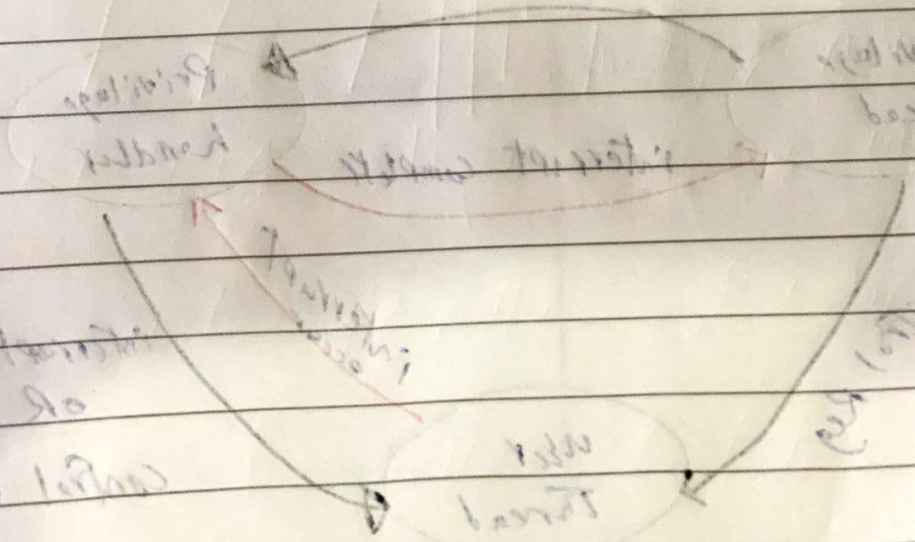
PSP → 1

MSP → 0

Control [0] ⇝ Access level

Privilege → 1

User → 0



DATE / / OBJECT

Operation Mode

1) Access mode

Privilage User

2) Processor Mode

→ Thread (Task), handler (ISR)

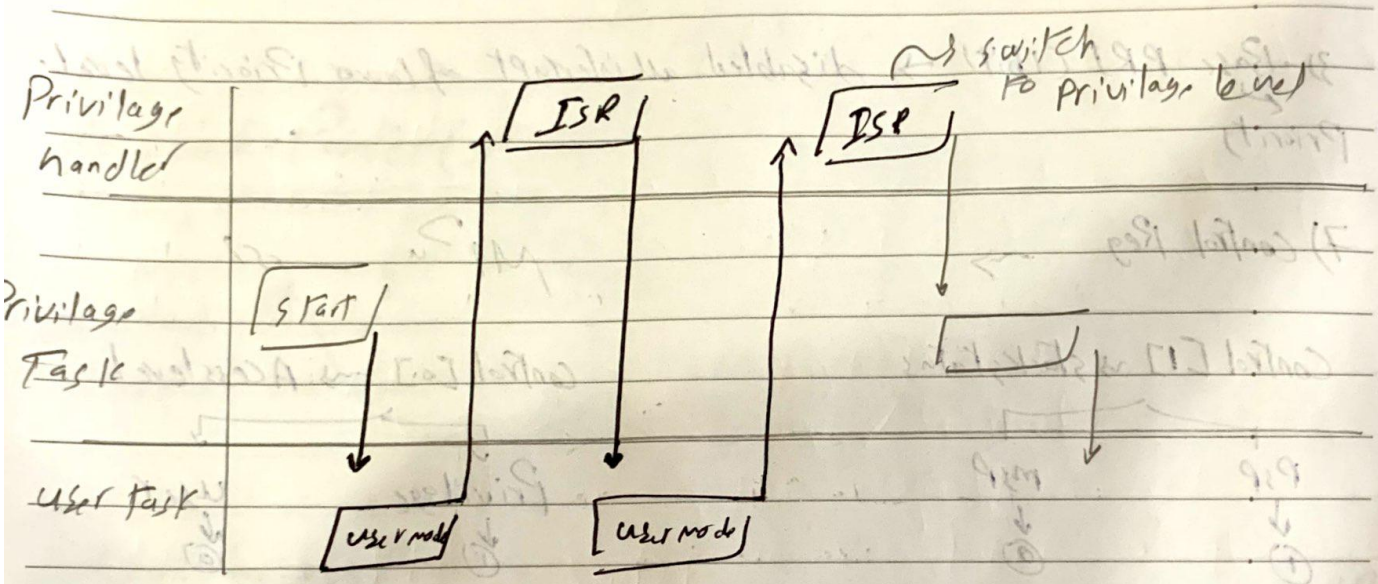
No interrupt interrupt

↝ handler → Privilage

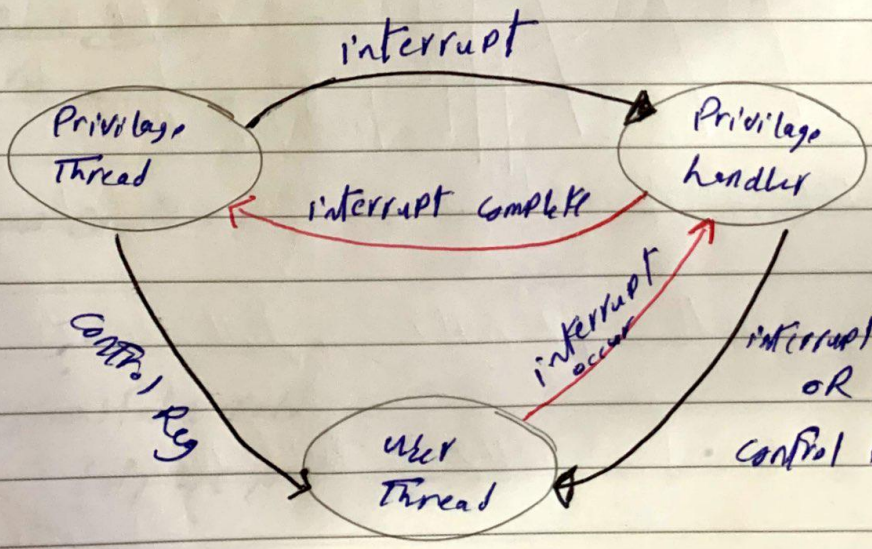
↝ Thread ⇒ privilage, user

mode

EX



State Machine



DATE / / OBJECT

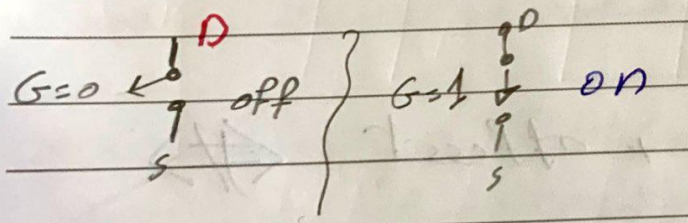
1) ⇒ GPIO at stm32f103c8 based on ARM cortex-M3.

* specification

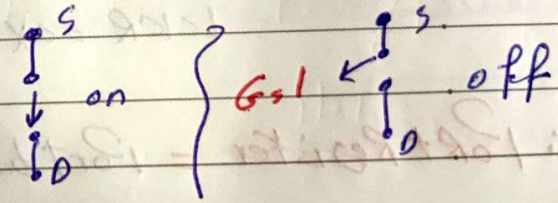
1) output state → Push Pull / open Drain
                → Pull up/down
2) input state → floating
               → Pull up/down
3) analog input (to ADC or out (from ADC)
4) high flexible Pin multiplexing using alternating fun.
5) Bit set & Reset register for bitwise atomic write Access
6) Fast Toggle every Two cycle
7) Speed Selection for output Pins
8) Locking mechanism 9) all Pins have external interrupts.

⇒ Transistor

N-MOS "open drain"



P-MOS "alternate fun"



Push Pull → N-MOS & P-MOS.

Port → 16 Pin ~> 0 → 15

DATE / / OBJECT

* GPIO Modes

* Input mode
  - Floating mode
  - analog mode
  - Pull up / Down mode

* output Mode
  - open drain
  - Push Pull
  - alternate fun Push-Pull
  - " " open drain

  → with different speed (2, 10, 50)

GPIO Register → Memory mapping

* Port base address
  - PortA = 0x 4001 0800
  - " B = 0x 4001 0C00
  - " C = 0x 4001 1000

* offset Reg ⇒ CRL → 0x00, CRH → 0x04, LCKR = 0x 18
  - IDR = 0x08
  - ODR = 0x0C
  - BSRR = 0x 10
  - BRR = 0x 14

∴ Port Register = Port base address + offset

* during & just after reset I/O ports configuration in input floating mode

DATE / / OBJECT

GPIO Res

1) GPIOx_CRL → Configuration Register low

CNF bits (2,3, 6,7, 10,11, 14,15, 18,19, 22,23, 26,27, 30,31) ⇝ Port Configuration bit
↓ input ↓ output

MODE (0,1, 4,5, 8,9, 12,13, 16,17, 20,21, 24,25, 28,29) → Port mode bits.
↓ reset ↓ speed 2 ↓ speed 10 ↓ speed 50

CRL → for Pin Num <= 7

CRH → for Pin Num <= 15

2) GPIOx_CRH → Configuration Register High.

3) GPIOx_IDR → input data register

bits → 16 → 31 → Reserved must be kept in reset value

bits → 0 → 15 → are read only & contain input value

4) GPIOx_ODR → output data register

bits → 16 → 31 → Reserved

bits → 0 → 15 → can be Read or write by S.W

Note → in atomic bit set/reset → can be set & clear by writing GPIOx_BSRR

5) GPIOx_BSRR → Set/reset register (Port bit)

bits (16 → 31) BR → bit reset → 0 → no action
→ 1 → Reset the corresponding bit ODRx

bit (0 → 15) BS → bit set → 0 → no action
→ 1 → set ODRx bit

6) GPIOx_BRR → bit reset register "clear bit"

bit → 16 → 31 → Reserved

bits → 0 → 15 → BRR → 0 → no action
→ 1 → Reset Pin.

DATE / / OBJECT

7) GPIOx_LCKR → Port configuration lock Register

sequence bits → 17 → 31 → Reserved

bit 16 [LCKK] → lock key ← 0 → not active
1 → active

lock key sequence

1- write 1 2- write → 0 3- write 1 4- Read 0 5- Read 1

* note → at writing seq The value of LCK[0→15] → must not change

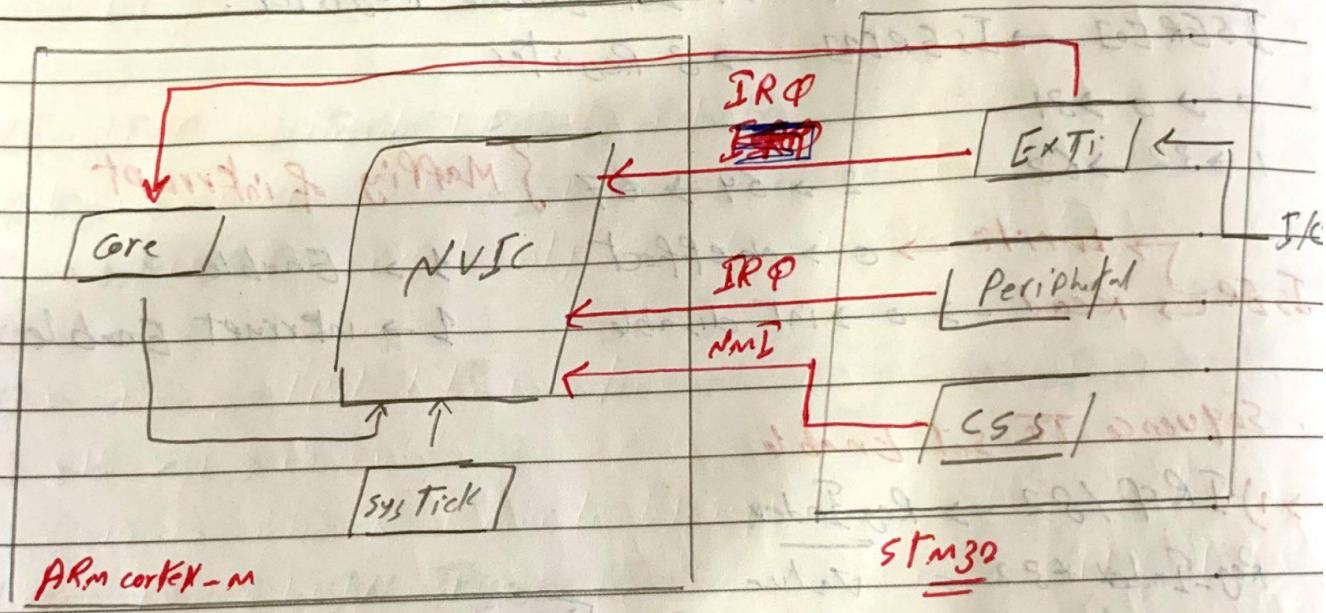
bits → 0 → 15 (LCK) → 0 → not locked
↘ 1 → locked

DATE / / OBJECT

2 ⇒ NVIC → Nested Vector interrupt controller.

Exception
- event that cause change to program flow
- exception handler → excuted when exception happen

Interrupt
→ one type of exception generated from (peripheral, external pins, Triggered by s.w)
→ ISR → interrupt service routine excueted when interrupt fire
→ interrupt disable
→ interrupt mask → Pending



IRQ → interrupt request
NMI → non maskable interrupt → clock security system

~~* Vector Table~~
1- Array of data
2- Relocatable Controlled by Program register [VTOR] Vector Table offset reg
↳ SCB
system cont block

DATE / / OBJECT

* Vector Table for STM32

Position → 0 → 59 (+13 Position → reserved

Priority start from → 7 → 60

Base address → 0x E000E100

Registers

1) NVIC_ISER → interrupt set Enable Register.

ISER[0] → ISER[2] → 3 Register

0 → 0 → 31

1 → 32 → 68 2 → 64 → 67 } Mapping of interrupt

ISER → write → 0 → no effect 1 → Enable

Read → 0 → int disable 1 → interrupt Enable

Sequence To set Enable

1) IRQ/32 → Reg_Index

Reg_Index * 32 = Value

IRQ_Value → bit الملي

in code

Loc_Reg_Index IRQ/32

NVIC → ISER[Reg_Index] = (1 << (IRQ % 31));

2) NVIC_ICER → interrupt clear Enable Reg

3 → Register

Sequence to clear → عكس الي فوق

DATE / / OBJECT

3) NVIC_ISPR → interrupt Set Pending register — write: 0 → No effect, 1 → change to Pending. Read: 0 → Not Pending, 1 → Pending

→ 3 Register

sequence → نفس فكرة Set

4) NVIC_ICPR → interrupt clear Pending register.

→ 3 Register

sequence → نفس فكرة Set pending

5) NVIC_IABR → interrupt Active Bit register "read only" → return → Get Active

→ 3 → register (0 → 2)

0 → Not active

1 → Active

6) NVIC_IPR → interrupt Priority Register

→ 16 Reg → 0 → 15 → IP[0] → IP[67]

[0:7] → byte offset 0

[8:15] → " " 1

[16:23] → " " 2

[24:31] → " " 3

* Note [0:3] → read Zero & ignore Wr

so need [<< 4] to set priority.

7) NVIC_STIR → SW Trigger interrupt Register "write only"

Bits [9:31] → reserved

Bit 0:8 → SW generated interrupt ID

* To initialize NVIC → by going to SCB system control block

→ SCB_AIRCR → Application interrupt and reset control register

1- Bit [16-31] → vector key → Read → 0xFA05 0000

↳ Register key — write → ~~0xFA~~ 0x5FA0 0000.

2- Bits [8:10] → interrupt group priority

→ split group Priority from Sub Priority.

DATE / / OBJECT

~~2) EXTI~~

3) DMA ⇒ Direct Memory Access

→ Introduction

1- Provide high speed data Transfer between Peripheral & memory
2- data can moved without CPU action.

⇒ features.

1- Two DMA with 12 channel (Stream request (7 → DMA1, 5 → DMA2))
2- Priorities are SW Programmable (low, medium, high, very high.
3- Source & destination Transfer Size (byte, half word, word)
4- Circular buffer
5- 3 event flag (half Tran, Complete Transfer, Transfer error)
6 → Mem To mem
7 → Per To mem, Per To Per, Mem To Per
8 → Access To (Flash, SRAM, APB1, APB2, AHP Peripherals
9 → No of data → 0 → 65536 (16 bit)

DMA functional description

1- DMA Transaction
1- Peripheral send request signal To DMA Controller
2- Controller serves request depend on channel Priority.
3 → acknowledge send
4 → Peripheral realease request after Acknowledge from controller

3 → operation

- loading → data from Pre/mem Through Pre/mem register (CPAR, CMA
- storage → data loaded into data register
- Post decrement → Contain no of Transaction That have still To Perfor

DATE / / OBJECT

3) Arbiter → Manage channel request depend on Pariority.
Two option → sw or Hw for manage.
1- Sw → at CCR register "Low, medium, High, very High"
2→ Hw → in Case of (Two) request have same Sw pirority
so channel with lowest Number get high pirority.

3) Programmable Size → are Control Through Psize & msize at CCR register.

4) Pointer increment → depend on PINC & mINC → CCR reg
1, 2, 4 depend on Choosen Size To get data Transfer right.

5) Circular Mode → handle Circular buffer & Continous data flow (ADC) → Enable or disable at CCR register

6) Memory to memory mode → Can work with Triggered by request and dot need To Select channel
Just Enable by sw at CCR reg

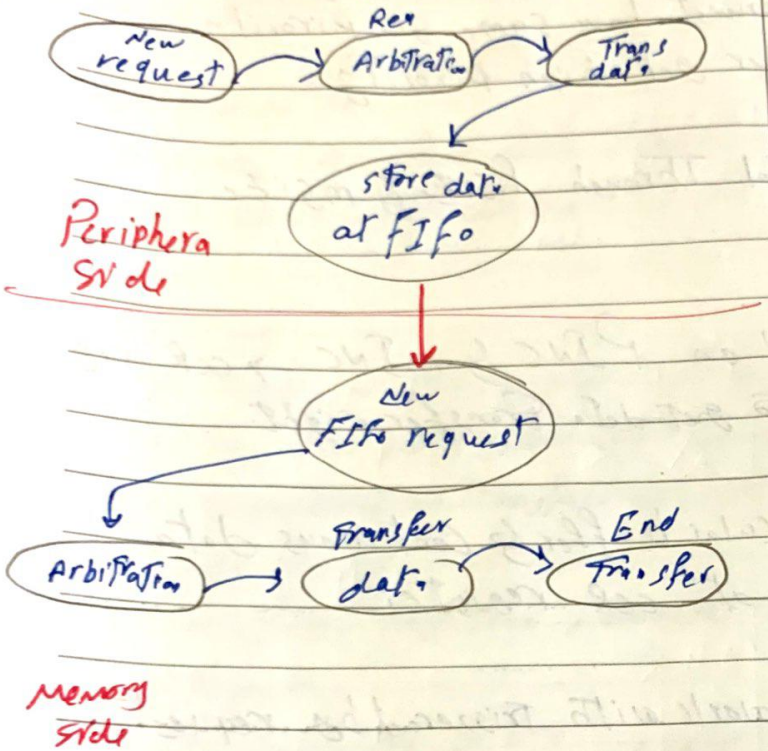
* Programmable data width & alignement → depend on
1- Source Port width (byte, half word / word).
2- destination " " "
3- NDT → No of data items to Transfer " " " "

7) error Managment
8) interrupt

DATE / / OBJECT

⇒ State Machine

1) Peripheral to memory



2) memory to Periphera

Side

⇒ DMA registers → Base address 0x 4002 0000

1- DMA_ISR → interrupt status register } → control DMA Peripheral

2- DMA_IFCR → " flag clear register }

3- CCR → Channel Configuration reg

4- CNDTR ⇒ Channel No of data reg

5- CPAR → Channel Peripheral address reg } → control channel registers

6- CMAR ⇒ " memory " "

[redacted]

⇒ for clearing Global interrupt flag → IFCR reg

32 bit → 28, 29, 30, 31 → reserved → 1111 → 15

DMA → IFCR = (15 << INT Number)

for Ex INT0 → 15 << 0 → 15 → 32 bit 0000 0000 0000 0000 0000 0000 0000 1111

INT0

DATE / / OBJECT

* flow work

1- Enable DMA clock in RCC "AHB ENR" register
2- disable DMA stream "channel" → CCR
3- select DMA channel "set priority"
4- selet Transfer Mode
5- set memory element size → MSize (6, 16, 32) → CCR
6- " Peripheral " " → PSize (6, 16, 32) → CCR
7- " Memory address increment → MINC (~~CRR~~ CCR → en/dis
8- " Peripheral " " → PINC (CCR → en/dis.
9- Set circular mode → CIRC (CCR → en/dis
10- Set interrupt handler for event → CCR // TCIE, HTIE, TEIE

DMA Configuration Summary

DMA Transfer Mode	Source	Destination	Flow controller	Circular Mode
Peripheral to Memory	AHB Per Port	AHB mem Port	DMA	Possible
			Peripheral	forbidden
Memory to Peripheral	AHB mem Port	AHB Per Port	DMA	Possible
			Peripheral	forbidden
Memory to Memory	AHB Peripheral Port	AHB mem port	DMA only	forbidden

DATE / / OBJECT

4) RCC → Reset clock Control.

1- Processor clock 2- Peripheral clock 3- Reset management.

buses → AHB OR APB

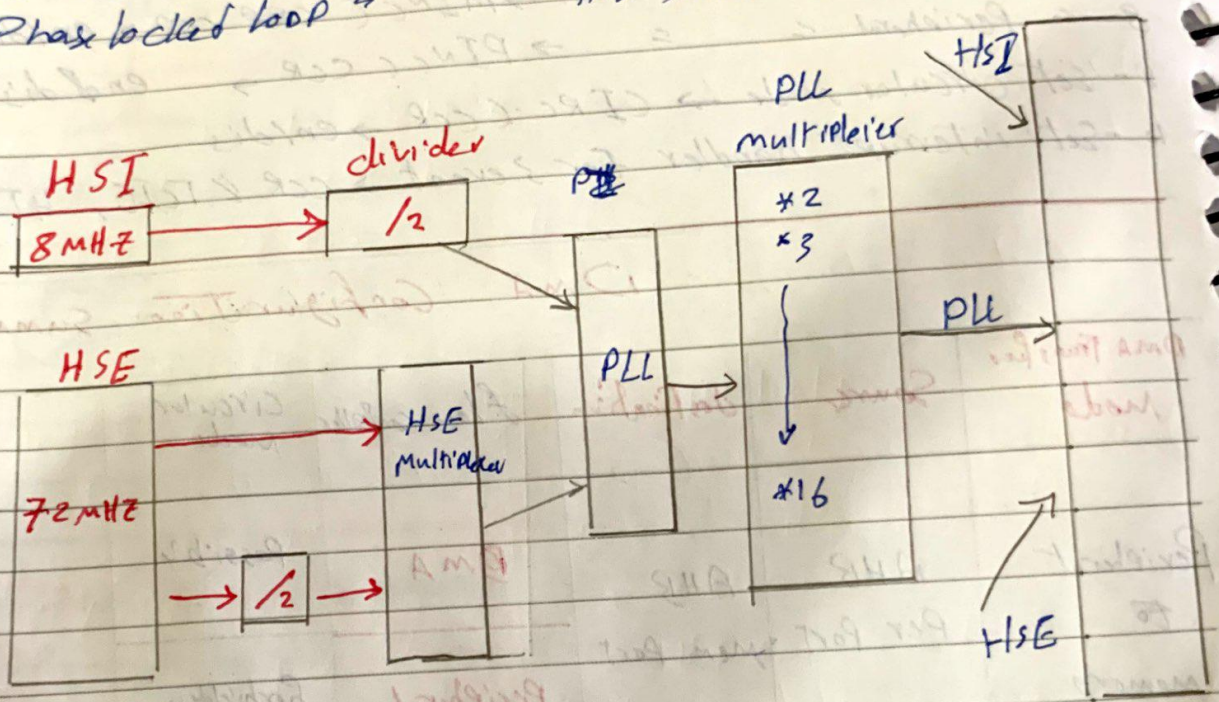
→ Three clock source

1- HSE → high speed external clock

2- HSI → high speed internal clock

3- PLL → Phase locked loop

HSI و HSE بيأخذ



→ basic Sequence

1- Enable bit every clock

2- Each clock has ready Flag

3- if PLL is on:

: a- PLL source

b→ PLL multiplier

4- To make current clock off → switch to another clock & wait for ready flag

DATE / / OBJECT

→ Three Types of Reset

1- System reset 2- Power reset 3- backup domain's Reset

1) System Reset → reset all registers value (except reset flags & register in Backup domain)

- generated when

1- Low level NRst Pin

2- window watchdog (WWDG).

3- independant " " (IWDG)

4- S.w reset

5- low Power mangement reset.

→ S.w reset → SysResetREQ bit → must be set to force Software reset.

→ Low Power management

1- standby mode → nRst-STOBy bit

2- STOP mode → nRst_STOP bit

→ Power reset

1- Power (on/down) reset

2- exiting standby mode

→ Backup domain

1- S.w Reset → BDRst → Rcc-BDCR

2- V_{DD} or V_{BAT} Power on

Clocks

1- HSE

2- HSI

3- PLL

Five Secondy

1- LSI Rc → low speed internal RC → 40 KHZ → auto wakeup

2- HSI Rc → high speed external Rc → 32,768 KHZ → real time clock (RTC)

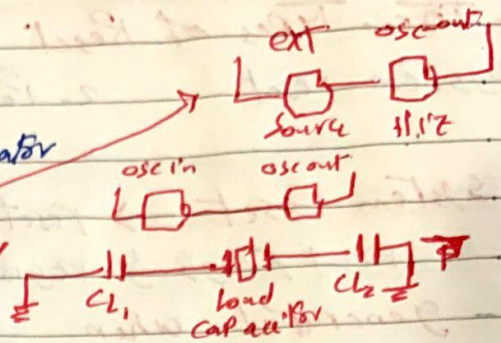
DATE / / OBJECT

1- HSE clock

The source

1- HSE external crystal/ceramic resonator

2- HSE user external clock



→ external source → freq up to 25 MHz → HSE bypass

→ " crystal/ceramic → 4 → 16 MHz

2- Hsi clock → internal 8 MHz RC

- Can be used as system clock

- or divided by 2 to use as PLL source

3- PLL → source 48 < PLL < 72 MHz

1- multiply HSI RC → at first HSI/2 Then multiply.

2- HSE crystal

* RTC clock

→ LSE RC

→ LSI RC } → used as RTC clock

→ HSE/128

* Clock out Put Capability MCO Pin [2:0]

- sys clk

- HSI

- HSE

- PLL clk divided by 2

DATE / / OBJECT

⇒ Control flow

1- choose system clock option → sys clk
→ HSI
→ HSE
→ PLL divided by 2
} CR, CFGR registers control

2- Set correction factor → CR Register

3- MCO PIN option → no clk → HSE clk
→ sys clk & → PLL clk divided / 2
→ HSI clk

4- clock security option → Enable / disable

5- Set AHB prescaler → CFGR Reg
6- Set APB1 prescaler → CFGR Reg
7- Set APB2 prescaler → CFGR Reg
8- Set ADC prescaler → CFGR Reg

Enable clk for buses → set bits
→ 1- AHB → AHBENR
2- APB1 → APB1ENR
3- APB2 → APB2ENR
for disable → clear these bits

DATE / / OBJECT

RCC Reg base add → 0x40021000

1- CR → clock control Register
2- CFGR → " configuration register
3- CIR → Clock interrupt reg
4- APB2Rst → APB2 Peripheral reset Reg
5- APB1Rst → APB1 " " "
6- AHBENR → AHB " clock enable Reg
7- APB1ENR → APB1 " " " "
8- APB2ENR → APB2 " " " "
9- BDCR → Backup domain Control reg
10- CSR → Control/Status Reg.

⟹ System clk

RCC_HSI

1- Enable bit
Set → CR → HSIon → bit 0
2- wait Ready Flag " booling "
CR → HSIRDY → bit 1
3- Sw → system clock switch "clear
Bit 1,0 → 00 → Hsi select

RCC_HSE

1- Enable bit
Set → CR → HSEon → bit 16
2- wait Ready Flag
CR → HSERDY → bit 17
3- Sw → sys clk switch
1 → Select HSE

RCC_HSE_RC

1- Enable bit
CR → HSEon
2- wait Ready Flag
CR → HSERDY
3- Set HSE bypass
CR → HSEBYP → bit 18
4- sys clk switch
1 → for HSE

DATE / / OBJECT

Rcc_PLL

1- clear multiplication factor at first → CFGR

2- Set " " → *2 → *16

3- check for PLL source

a- PLL - HSI/2

Enable → HSIon at CR

CLR → PLLSRC at CFGR → choose HSI

b- PLL - HSE

Enable → HSEon → CR

Set → PLLSRC → CFGR → HSE select

c- PLL - HSE/2

Enable → HSEon

Set → PLLSRC

Set → PLLXTPRE → Hse Clk divided by Two

4- Enable bit → PLLon → CR

5 → wait Ready Flag

6 → sys clk switch → 2 → PLL select

DATE / / OBJECT

5) AFIO → Alternate function I/O

→ optimize number of peripheral available
→ remap some AF function to another Pins.

AF Reg → Base → 0x4001 0000
1- EVCR → event control Register
2- MAPR → Remapping & debug I/O configuration Reg
3- EXTICR[4] → Ext interrupt Reg
4- MAPR2 → Remapping & debug I/O config Reg 2

* Workflow.

Line configuration

1- get Reg Index → = Line / 4 .
2- get offset → Line % 4
3- offset *= 4
4- clear 4 first bit at EXTICR by offset & Exti clear line (0x F)
5- set The mapping with Port << offset.

DATE / / OBJECT

6) EXTI → External interrupt event controller.

Features:
1- independant Trigger & mask on each Interrupt/event.
2- Status bit for Int line.
3- up to 20 int/event requests

* Trigger event → 1- rising 2- falling 3- both (rising & falling)

* wakeup management → handle internal/external event "wFE"
1- Enable interrupt in Peripheral control Register
2- Configure Ext/int EXTi line in event mode To resume wFE
"wFE" → wait for event.

* functional
- Enable interrupt line
- Enable interrupt bit in Mask Register } → for interrupt
→ for event
- Programming Two Register with edge select
→ S.w
- Set interrupt/event register

H.w interrupt	h.w event	S.w
1- Configuration mask bit IMR → Reg	1- Config mask bit EMR	1- Config maskbi IMR / EMR
2- Config Trigger Selection RTSR & FTSR → Reg	2- Config Trigger selection	2- Set required bit (SWIER
3- Config & Enable mask bit that control NVIC IRQ		

16 External Int/event — PA0, PB0, PC0 } EXTI0 ⇝ PA15, PB15, PC15 } → EXTI 15

EXTI 16 → PVD
= 17 → RTC

EXTI 18 → USB
= 19 → Ethernet

DATE / / OBJECT

* EXTI Register → Base → 0x 4001 0400
1- IMR → interrupt mask register
2- EMR → event mask register
3 → RTSR → Rising Trigger selection Reg
4 → FTSR → falling " " "
5 - SWIER → S.w interrupt/event Reg
6 → PR → Pending register

---

* Control flow instruct Es.
1- for initialize
1- Reset Registers value at first
2- un mask interrupt line → clear → IMR
3 - Select Trigger option
a - Rising edge
Set → RTSR & clR → FTSR
b - falling edge
Set → FTSR & clR → RTSR
c - both
Set → RTSR & Set FTSR

4 - Enable or disable line → IMR → Reg
5 - S.w Trigger → Set → SWIER → Reg
6 - clear Pending flag → PR Register → To know if Trigger accura

DATE / / OBJECT

7) SysTick Timer (STK)

→ 24 bit system Timer Count down from reload value to zero

→ STK source → AHB "Processor clock"
↳ AHB/8

→ Time unit → ms → load value = Time enter / clock source/1000
↳ us → load value = Time enter / clock source/1000 000

equation

→ STK Reg → Base → E000E010
1- CTRL → Control & status Register
2- load → Reload value Register
3- VAL → Current value Register
4- CALIB → Calibration value Register

→ Control flow
1) → initialize
1- Reset Register value
2- check for stk- source
a- AHB source
Set → CTRL → clk source → bit 2 / Global clk = AHB source.
b- AHB/8 source
clR → CTRL → clk source → bit ? / Global clk = AHB/? source.
2) → Set wait
1- switch Time unit.
local value = equation
→ Load Reg = local value
2- Timer initialize to count → Set → CTRL → Enable Counter →
3- check for counting flag → CTRL → bit 16
4- stop Timer

DATE / / OBJECT

3) Periodic interval
1- Switch for unit & load value
2- Enable Timer
3- Enable interrupt (CTRL → TickInt bit)
4- Call back fun

4) Single interval
1- disable Counter
2- clear value Register VAL = 0x0000 0000
3- switch for unit & load value
4- Enable Timer
5- Enable interrupt
6→ Call back

5) stop stk
1- stop Counter
2- disable interrupt

6) Resume stk
1- Enable Counter
2- " interrupt

7) start stk
1- clear "VAL" Reg
2- load → Reg with max 0xffff ffff
3- Enable Counter

8) get elapsed Time
1- Calculate local value
value = Load Reg - VAL Reg
2- switch for unit & Calcu
elapsed Time = value * $\frac{\text{clock}}{1000}$ ms
3- return elapsed Time

9) get remaining Time
1- Calculate local value
value = VAL Reg
2- switch for unit &
get remaining Time
3- return remaining Time

DATE / / OBJECT

8) Timer

→ Tim1 & Tim8 → advanced Timer
→ Tim2 → Tim5 → GPT
→ Tim9 → Tim14 → GPT
→ Tim6 & Tim7 → Basic Timer

Features

- advanced Timer (1 & 8) & GPT & Basic Timer
  - 16 bit reload (up, down, up/down)
  - measure Pulse length
  - Generate waveform (output compare, PWM)
  - independent & not share any resource
  - 4 Channels for → 1- input Capture 2- output compare
    3- Pwm generation 4- one pluse mode
  - interrupt / DMA generation on
    1- update counter (overflow / under flow)
    2- Trigger event 3- input Capture 4- output compare.
    5- break input.

* Functional

1- Time base unit → 1) Counter Reg (CNT) 2) Prescaler reg (PSC)
3- auto Reload Reg (ARR) 4- Repetition (RCR)

Steps:

1- writing or Reading on auto Reload Register
2- Preload register Transfer into shadow register
3- update event Sent after (overflow / down counting / under fl
4- enable CEN in CR → refer to slave mode controller.

DATE / / OBJECT

Timers Modes

1- input Capture Mode:

- Capture/compare register used to latch value of Counter.
- when Capture occur flag in SR Reg "CCXIF" is set
- if Capture occur while "CCXIF" is set → The over flag "CCXOF" is set

steps to detect capture

1- Select active input "CCR1" must be linked
2- write CC1S to config channel to be input
3- Program input filter "ICXF" in CCMRX Reg
4- Select edge of active "CCER" Reg
5- Enable Capture from counter "CCER" Reg
6- Enable interrupt request

as soon as Capture occurs

1- CCR1 → gets value of Counter
2- CC1IF flag is set & CC1OF flag is set at case capture occur when CC1IF is high
3- interrupt generated "CCIE" } EGR Reg
4- DMA request generated.

1-2- Pwm input mode → Case of input Capture

by select active Polarity as first rising Then Falling

Then read duty cycle

DATE / / OBJECT

2- output compare mode

→ used to control output waveform
→ when match occur
1- assign output Pin (CCMRx) Reg to → set active / set inactive / Toggle
2- Set flag in (SR Reg) → CCxIF
3- generate interrupt at DIER register
4- " DMA request at DIER register

steps to get into this mode
1- Select Counter clk (internal / external / Prescaler)
2- write data at APR & ccRx reg
3- Select out put mode at match (Set, disable / Toggle)
4- Set CCxIE in IR ϕ
5- Enable Counter → CR1 reg

3- Pwm "Pulse width modulation

→ freq → ARR
→ duty cycle → CCRx
- Can select independently on each channel → Pwm mode 1 → 110 / " " 2 → 111 → CCMRx
- Preload register transfer to shadow Register when update event before start Counter
- initialize all register → set UG at EGR Reg
- Set Polarity → CCxP → CCER Reg as active high or active low

* Pwm → aligned mode / center aligned mode → depend on CMS bit in CR1 Re

aligned mode → up Count / down Count → DIR → CR1

Centeraligned mode → active when CMS bits are different from 00

DATE / / OBJECT

Timer Reg → Base address → Memory MAP at datasheet

1) advanced Time
- CR1
- CR2
- SMCR
- DIER
- SR
- EGR
- CCMR1
- CCMR2
- CCER
- CNT
- PSC
- ARR
- RCR
- CCR1
- CCR2
- CCR3
- CCR4
- BDTR
- DCR
- DMAR

2) Basic Timer
- CR1
- CR2
- DIER
- SR
- EGR
- CNT
- PSC
- ARR

3) GPT
- CR1
- CR2
- SMCR
- DIER
- SR
- EGR
- CCMR1
- CCMR2
- CCER
- CNT
- PSC
- ARR
- CCR1
- CCR2
- CCR3
- CCR4
- DCR
- DMAR

DATE / / OBJECT

⇒ Control Flow

1- Timer Configuration 1- Reload value 2- Timer ID
3- Count direction 4- interrupt 5- Call back (IRQ)

2- Enable Timer
1- switch (Timer ID)
2- Set (Tim ID → CR1 → (CEN) → Counter Enable

3- disable Timer
1- switch (Timer ID)
2- Clear (Tim ID → CR1 → CEN → Counter disable.

4- Set reload value
1- switch (Timer ID)
2- TimerID → ARR Reg → entered value

5- initialize Timer
1- disable Timer at first
2- Set Reload value
3- Set Interrupt update → DIER → UIE "bit0"
4- Set ~~Call back~~ Counter direction DIR → CR1 → bit 4 → 0 → up, 1 → down
5- Set Callback
6- Enable interrupt
7- Enable Timer

DATE / / OBJECT

9) FPEC → flash Program erase Controller.
- handle the programing & erasing operation on flash
- 7 → registers
* key values → RDPRT key = 0x00A5
key1 = 0x45670123
key2 = 0xCDEF89AB
operation → 1- Program flash 2- Program Halfword
3- erase Page 4- erase Area 5- erase mass 6- onebyte Programming

FPEC Reg Base → 0x 4002 2000
1-ACR → Flash Access Control register
2-KEYR → key register
3-OPTKEYR → option byte key register
4-SR → status Reg
5-CR → Control Reg
6-AR → address Reg
7-OBR → option byte reg
8-WRPR → write protection Reg

1- unlock flash memory sequence
1- check for lock bit for CR Register bit 7
→ KEYR = key1
→ KEYR = key2
2- Flash Programming
1- Read Flash Lock Sequence
2- Set PG at CR
3- Perform halfword write
4- check for Bsy at SR → Flash Busy or not
5- Set Eof at SR
6- CR PG at CR

DATE / / OBJECT

3- flash memory erase

1- Page erase

1- Check for Bsy at SR
2- check for lock sequence
3- Set PER at CR Reg
4- Set start erase (CR Reg.
5- Check for Bsy
6- disable erase
7- Set Eof
8→ clr PG

2. mass erase

1- check for Bsy
2- lock sequence
3- Set MER at CR Reg
4- Set start erase → CR Reg

3. erase Area

1- Set first & last Page in Area
2- Loop for Page erase

4- one byte option programming

1- check on Bsy
2- check for lock sequence "optkeyR"
3- unlock oPTwRE → CR
4- Set oPTPG → CR
5- write data Halfword
6- check for Bsy
7- Set Eof , clr PG

erase

1- check for Bsy
2- Set oPTwRE
3- Set oPT ER
4- Set start erase
5- check Bsy
6- Set Eof , clr PG

DATE / / OBJECT

b) USART → universal synchronous Asynchrouns receiver Transimetter

- Serial - full duplex - synchronous & Asynch
- Peer To Peer - serial - wide range baud rate.

→ NRZ asynch serial data formal
→ data word length 8, 9 bits
→ support for a stop bit
→ Separate enable bit for TX & RX

→ 3 Transfer detection flag → Receive buffer full
→ Transimit buffer full
→ End of Transmission flag

→ Parity control → Transmit Parity bit
→ check Parity of Received data

→ 4 error flag → overrun error → Frame error
→ Noise error → Parity error

* functional of RX & TX

1- Idle Priority for TX OR RX
2- start bit
3- data word "LSB"
4- stop bits
5- baud rate
6- status Reg (SR) 7- Data Reg (DR) 8- baud rate Reg (BRR)
9)- guard time

DATE / / OBJECT

UART Reg → Base register mapping

1- SR → status Register
2- DR → Data Register
3- BRR → Baud rate Reg
4- CR1 → control Reg 1
5- CR2 → " " 2
6- CR3 → " " 3
7- GTPR → Guard Time & Prescaler register

* flow control

1) → initialize
1- reset Register value at first
2- disable CR3 Reg bits
3- disable CR2 Reg bits except (CPHA) bit to Get second clock fra
4- disable CR1 Reg bits
5- Set baud rate → BRR
6- Enable Peripheral → UE "13" at CR1

2) Transmit Synchronous
1- check for data not equal Null
2- Put data into DR Reg
3- wait to Transmit finish → SR → TC

3) Receive Synchronous
1- check for data Read Register is not empty → RXNE → SR
2- get data ~~from~~ into DR reg
3- check for data recieved.

DATE / / OBJECT

4- Receive Byte Synch
1- Check for recieve finish
2- loc data Loaded from DR Reg
3- return byte local data

5- Recieve data Asynch
1- Call back fun to handle DR Ø
2- Enable Recieve interrupt

DATE / / OBJECT

11) SPI ⇒ Serial Pheripheral interfac

* Specs

1- Full duplex 2- synchronous 3- 8 or 16 bit frame format
4- multi master capability 5- 8 master baud rate prescaler ($\frac{f_{PCLK}}{2_{max}}$)
6- Programmable clock polarity & phase
7- SPI busy flag 8- master mode fault / over run

* functional

PIns	Meaning	Master mode	slave mode
MOSI	Master out for slave in Put	out Put	in Put
MIso	Master in Put slave out Put	in Put	out Put
sck	Serial clock	out Put	in Put
NSS	slave Select	out Put	in Put

* slave Select (NSS) management → SSm bit at CR1 Reg

1- Hw or Sw

S.w → SSm = 1

H.w → SSm = 0

↳ if → SSoE at CR2 = 1 → NSS signal driven Low when Commun
and kept until SPI disable

↳ if → SSoE = 0 → classic in Put
slave selected when Low
↳ dis Elected ↳ High

* Data frame → shifted out (MSB or LSB)
de Pend on site on Dff in CR1 Register.

DATE / / OBJECT

*Configuration of Master

1- Select BR → CR1 → to define serial clock baud rate
2- Select CPOL & CPHA
3- Set DFF at CR1
4- Config Fram format
   LSB first → CR1
5- NSS Pin
   ↳ input mode / [redacted]
   i/p H.w mode → NSS → high level signal
   i/p sw mode → Set SSM & SSI
6- Set MSTR & SPE at CR1

Transmit Sequence

- written in TX buffer
- data loaded in to shift Register
- flag is set TXE
- interrupt generated TXEIE

Receive Sequence

- data in shift Reg Transferd to RX Buffer
- interrupt generated if RXEIE is Set

Configuration of Slave:

- Set DFF bit
- Select CPOL & CPHA as same master
- frame format (LSB / MSB) as master
- clear MSTR Pin
- Set SPE

Transmit Sequence

- loaded to TX buffer
- TXE flag is Set
- interrupt generated TXEIE

Receive Sequence

- data in shift register Transfer to RX Buffer
- interrupt generated if RXNEIE is Set
- copy data byte
- CR RXNEIE

SPI Reg

- CR1 → control Reg 1
- CR2 → " " 2
- SR → status Reg
- DR → Data Reg
- CRCPR → CRC Polynomial Reg
- RXCRCR → RX CRC Reg
- TXCRCR → TX CRC Reg
- I2SCFGR → I2S configuration Reg
- I2SPR → I2S prescaler Reg

Control flow

1) → initialize configuration of SPI

Configuration → Mode & CPol & CPHA & framesize & frame forma
Sw-slave manag ment, Transmission mode, Prescaler, interrupt stat

1- reset Register Valu
2- switch for mode config
1 → master 0 → slave
2- switch (clock polarity) CPol
1 → clk to 1 0 → clk to 0
3- switch (clock phase) CPHA
1 → second clock first data capture
0 → first " " " "
4- switch (frame format)
1- LSB first
0- MSB first
5- switch (frame size) DFF
1 → 16 bit 0 → 8 bit
6 → Sw. slave management
1 → enable 0 → disable

7- switch (Transmisson mo
1- Recieve only
0- full duplex (TX & R
8) Prescaler
- check for Prescaler < 8
set BRo of CR1.

DATE / / OBJECT

2- Send / Receive Synch
1- Send data To DR Reg
2- Check for SPI Busy flag Bsy at SR
3- Return data

3- Send/Rec Asynch
1- Call back fun
2- Send data To DR Reg