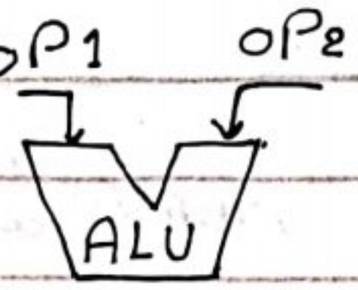
1

Advanced RISC Machine
ARM
Cortex _ M

1 مقارنة بين MicroController

AVR (Atmega32) 1.5$	STM32F103C8T6 0.7$ ↳ ARM Cortex M3
* يعني أكبر عملية بعملها تكون 8 bit	* Processor 32 bit
Processor 8 bit	Max Freq 72 MHZ (UP to)
F. Max 16 MHZ → يعني يقدر يعمل 16 مليون عملية في الثانية	
* (Flash Memory) 32k	* Flash M 64k
* RAM 2k → 10 أضعاف	* Ram 20 k
Pre * I/O 32 Pin	* I/O 35 Pin
* Timers 3	* Timers 9 (16bit) أصغر واحد
* 8 ADC Channels	* 10 ADC Channels
* 1 USART	* 3 USART
* 1 SPI	* 2 SPI
* 1 I2C	* 2 I2C
* 5v	* DMA , CAN , RTC
* بسط يستخدم للتعليم وبالتفاصيل في ال datasheet	* 3.3v وشوية Prefairals أثناء الكورس.



(سؤال ايه علاقة دقة التصنيع لسرعة الـ Processor و استهلاك الطاقة)

Embedded System Challenges
↳ Size
↳ Cost
↳ Performance
↳ Power consumption

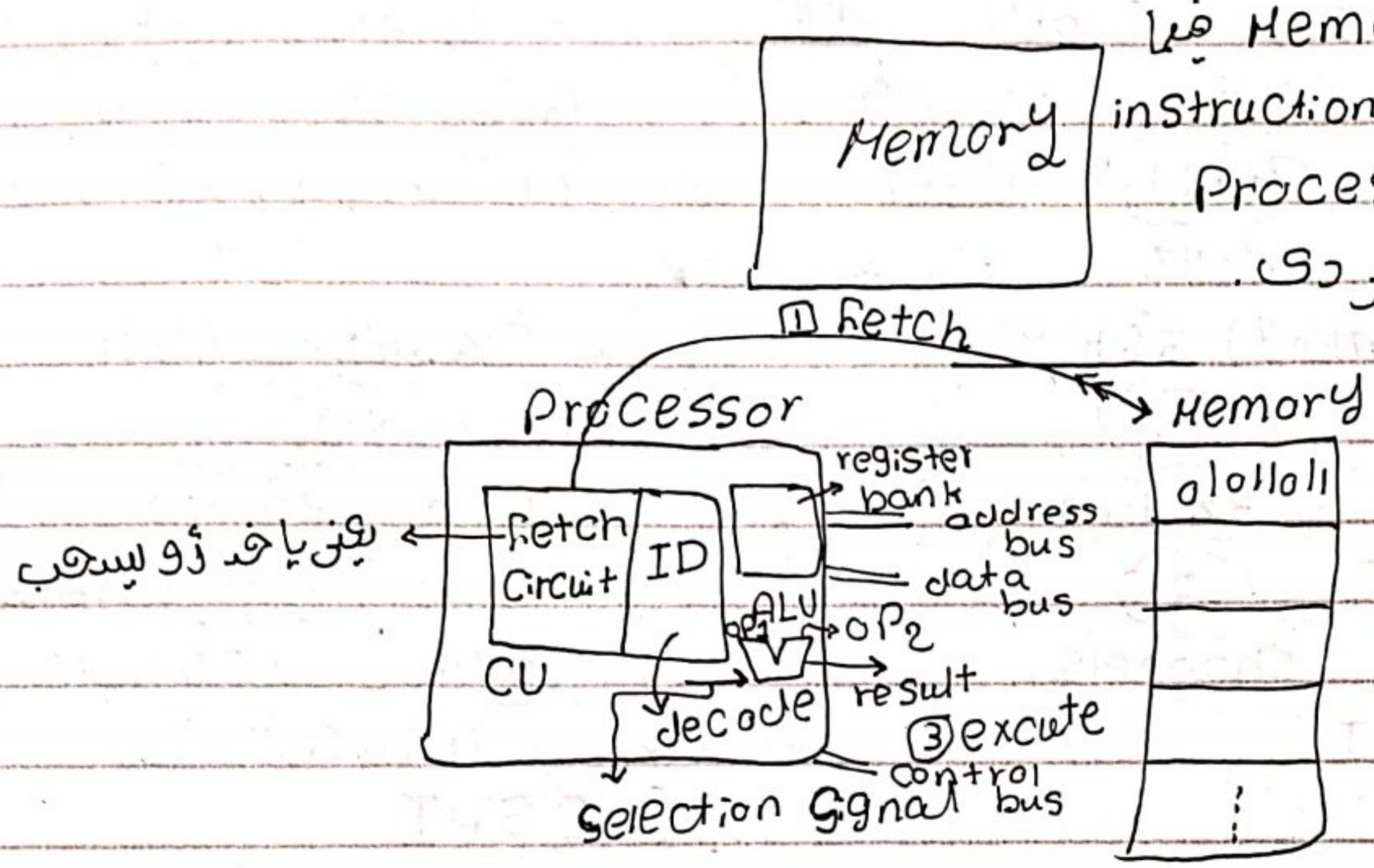
ARM Arch

الـ Processor بيشتغل ازاى ؟

Processor

الإجابة: الـ Processor هو المسئول عن عملية تنفيذ اللود (instruction excution)

مثلاً عندنا Memory فيها مجموعة من الـ instruction ووظيفة الـ Processor إنه ينفذ الأوامر دى.



كل الـ Processor دوره يمر على كل location وينفذ كل instruction ويدخل على التى بعده وهكذا.

* الـ Processor بيعمل كدا ازاى ؟

الإجابة: عن طريق دايرة جواه اسمها fetch circuit وهى جزء من CU (Control unit)، والـ fetch cir هى المسئولة عن سحب الـ instruction من الميموري إلى الـ instruction by struction وده هى عملية الـ fetch وده أول مرحلة الـ cycle بتاعة الـ Processor إنه يروح يعمل الـ fetch.

وليك جه لـ Processor البايزى الـ فى الـ Memory دا هيعمل ايه؟ فيه بقى الجزء التانى فى CU اسمه Instruction Decoder هى الـ Unit المسئولة عن عملية الـ decoding، يعنى ايه؟ يعنى هى الـ بتفهم الـ instruction دى معناها ايه؟

3

Instruction Decoder :

هى ال Circuit المسئولة عن فهم ال instruction وعلشان يفهمها بيحتاج معلومتين

① Instruction set

عبارة عن جدول

Instruction	OPcode
ADD	010
SUB	100

→ operational code

مثلاً

كل عملية ليها OPcode خاص بيها مبتكررش من instruction للتانية

* يبقى دى أول حاجة بيعرفها ال instruction decode

② Instruction format.

هى معلومة بتقول ال Instruction decoder إزاى هتقسم ال instruction ال جايالك ؟

Ex:

3 bit	3 bit	2 bit
OPcode	operand 1	operand 2
010	110	11

بعدين هياخد ال 010 ويروح على Instruction set

ويقول دا معناه ايه ؟

هيلاقى معناه ADD ، وال بعده 110 دا كدا رقم 5 وال operan 2 معناه 3 ، ويبقى ال instruction دى معناها إن عايزة أجمع 3 + 5 وبكده حصل هو هيفهم بس لكن مش هينفذ ، هيقوم بعتها على ال ALU (Arithmetic logic unit) وهى ال unit اللى فيها كل الدوايس الخاصة بال instruction اللى بيعملها ال Processor زى Add , Muti , Sub .....

علشان ال ALU تشتغل بتحتاج ٣ حاجات :

① operand 1 ، operand 2 ، selection signal

وبكدا ال ALU هي هتطلع ال result

وده بقى عملية Excute .

selection signal ↓ مسئولة عن فتح وقفل دايرة معينة من الدوايرالموجودة داخل ALU

* طيب سؤال ؟

طيب ال بيعرف ازاى أن location علي الدوى وفين بتخزن عملية الجمع مثلا ال result ؟

ج: هنا بقى في ٥ جزء في CU اسمه Register bank or file هو مجموعة من ال register جوا ال processor مثلا

* معلومة : ال Memory و register :

نفس الوظيفة ، يعني ال Re هو عبارة عن حتة Ram ، ولكن يطلق كلمة register على ال memory location ال ليه وظيفة خاصة في حاجة معينة .

① أشهر هذه ال register هو PC (Program counter) وهو ال بيشاوي على ال instruction ال عليها الدور في ال memory (على مكانها) ، ولكدا عرفنا ال fetch بيعرف من المكان ال في PC

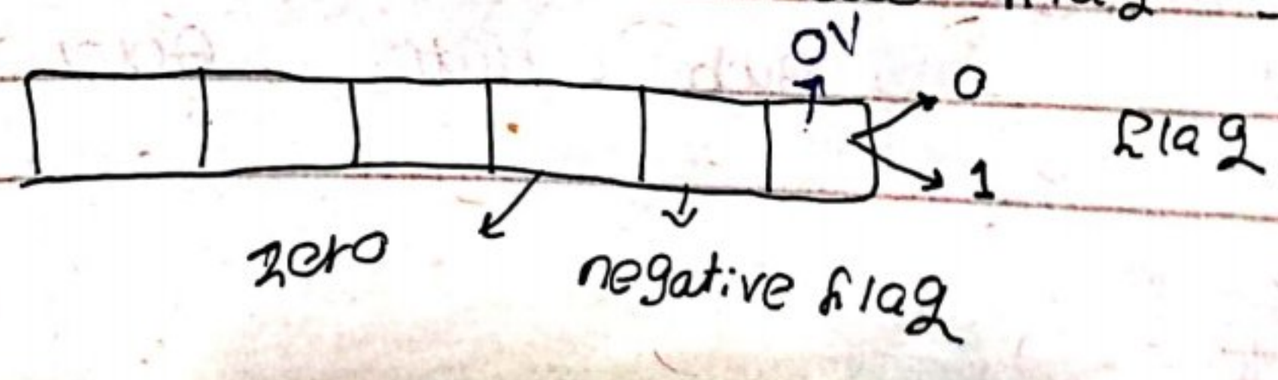
* في كمان (IR) (Instruction register) :

بعد ما عرفنا مكان ال instruction في ال Memory فعملية ال (fetch) ② أخدتها وحطيتها في ال IR وهو بيشيل ال instruction ال بتنفذ دلوقتي وبكدا ال Instruction decoder بيبدأ decode ال instruction الموجودة داخل IR

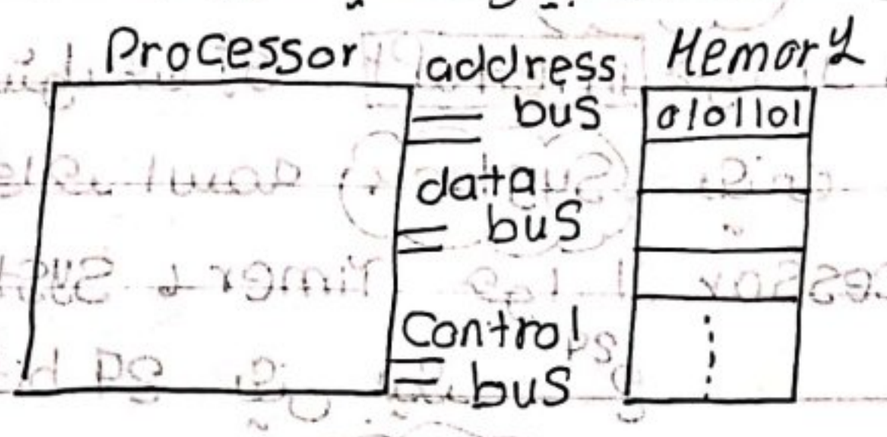
③ ، فيه كمان ACC (Accumulator) وال register دا فيه عملية ال operation ال عملناها 3+5 = 8 ، يعني دايما في النتيجة ال هيطلع بيها بيها ال Processor

④ كمان PSW وهو من أهم ال registers الموجودة في R bank (Processor status word) وفيه مجموعة من ال (flags) ال flag معناه 1 bit

وال flags دي بتعبر عن حالة ال Processor



بين الـ Processor والـ Memory والـ Prepherals بيكون فيه buses



① الـ address bus :
بنحط عليه address الـ location اللي عايز أكلمه سواء location في الميموري أو Address of PrePhiral

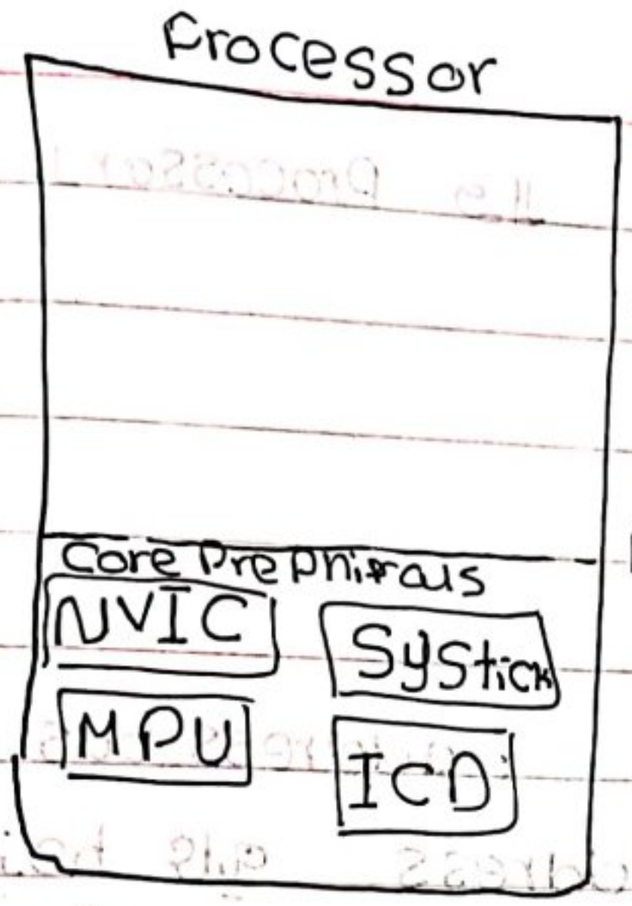
② الـ data bus :
الـ CPU هتطلع عليها الداتا الموجودة في الـ location اللي address بتاعه على الـ address bus وسواء هتقرأ الداتا أو أكتب فيها أو أعمل operation معينة

③ الـ Control bus :
هي مجموعة من الـ signals بتتحكم في العملية اللي عايز أعملها
قد تكون read or write أو مجموعة أخرى
وبعد كدا تطلع الـ data بتاعة العملية على الـ data bus

ملاحظة :
لما ذكرنا إن الـ Register داخل الـ Processor بتاعة الميموري.

Register أسرع بكتير لأن الـ Connection مباشر من غير الـ buses بالرغم إنها متصلة بالـ Processor

ARM بتقدم قالب هتعمله جاهز جوا الـ Processor اسمها Core PrePhirals يعني هتعملك PrePhirals بتتعامل معاها سريع جداً وقوية جداً وفوق دا كله arm العملاقة



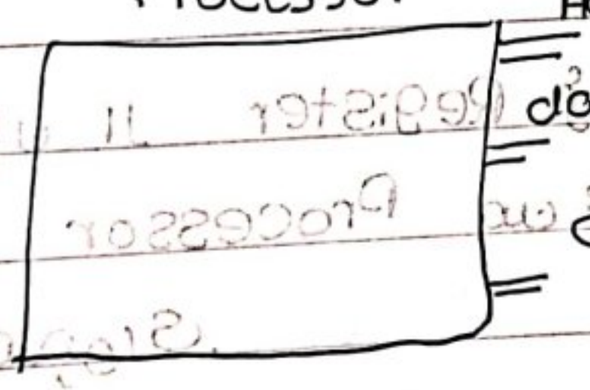
① أول Core Prephiral اسمه NVIC وهو مسئول عن أي interrupt جوا الـ Sys

② و تاني واحد اسمه Systick يعني System Timer ، Timer جوا الـ Processor وهو 29 bit يعني بيعد 2^{29}

③ تالت واحد اسمه MPU (Memory Protiction Unit) وده عبارة عن Prephiral بقدر يحمي الـ Memory بتاعتك يعني يخلي حته Memory معينة محدش يقدر يقرأ منها أو يكتب عليها (الحاجة دي مهمة)

④ الرابع اسمه ICD اختصار In Circuit Debuger هو عبارة عن tool جوا الـ Processor تقدر تخليها debuge وديباج معناه إننا نشوف الـ Processor أثناء الـ runtime وده بيخلي الـ Processor يمشي بسرعة إن شاء الله ولو ساعة علشان أشوف الـ output في كل سطر علشان لو حصل بجاية.

ARM Bus Architecture



Address bus # of location

data bus # of size of data one loca

مثال address bus size

8 bit = data bus ، 8 bit

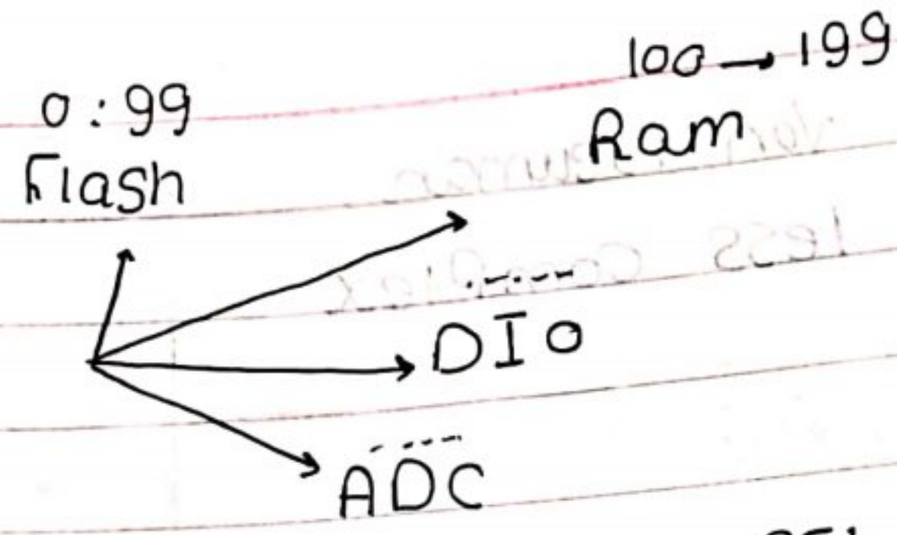
* الـ address bus بيعرفني عدد الـ locations الأقدر أكلمها

يعني أقدر أكلم 2^8 location يعني 256 location ← 0 : 255 from

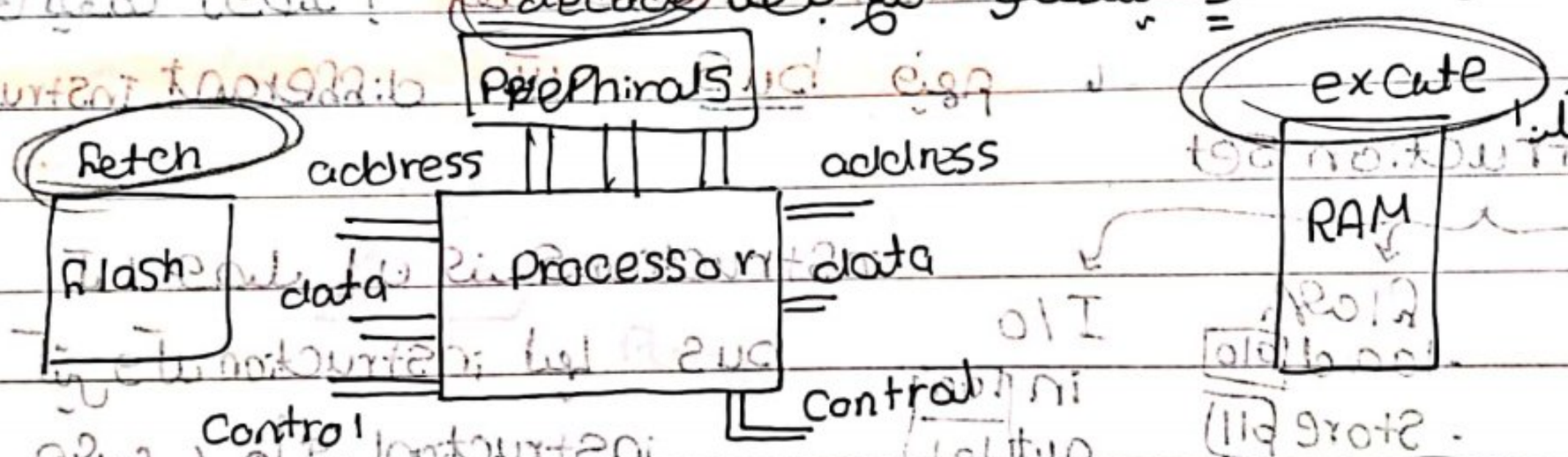
الـ data bus بتعبر عن Size الداتا في الـ location الواحد.

الـ Word Size هو Size الكلمة ، ولو أنا هعمل داتا فلازم يكون حجم الـ databus مساوي لـ databus . ~~لازم نفس~~ بتاع الـ Processor بتاع الـ Prephiral ، لازم يكونوا متساويين

مثال



وإنا عندى location 256 وكل واحد فيهم 8 عنده شوية location
عايز يـ Connect مع الـ Processor ، فهيوزع (P) الـ location
عليهم مثلاً Flash 100 ، Ram 100 ، DIO 32 Address
، ADC الباقى . وكل واحد ليه address مختلف ، والتفاصيل هنا
خاصة بالـ digital design . واسم العالم Von Neumann
، حتى بقى Harverd قالت لما تكون دايرة الـ fetch بتعمل
fetch دايرة الـ decode بتعمل ايه ؟ ع : خاصية مستنية الـ fetch تخلص
وهكذا الـ excute أثناء الـ decode
فكروا إنهم الـ ٣ يشتغلوا مع بعض decode ؟



Bus Set يبقى address + data + Control
الثلاثة مع بعض .

فكده عملنا Processor له busset للـ RAM والـ Flash
والطريقة دى اسمها Pipelining الفكرة عندى أكتر من
bus set وبالتالى دى اسمها Harvard Arch

كده كبداية تعوسلا تخيه RAM وعاكه
!!!

بعدين + ٤ ! قبلها

بسيطة
ومش
بتنسيها

(8) PortC → Pin13 → led on Microcontroller

Von Neuman	Harvard
* less complex	* faster * complex Software ↓ int *Ptr=0; نكتب كدا *Ptr = 100;

أيهما أفضل Harvard

كدا بكتب على الـ address (0) قيمة 100، فالعادي إن في Van Num الـ address الـ zero موجود في الـ flash، لكن في Harvard عشان الـ address: للـ bus set للـ bus تبدأ بـ zero

فأيهما أقصد؟ هنا بقى الـ Processor قال هعمل different instruction للـ bus فيهم، instruction set

RAM
- Read 000
- Write 001

flash
- load 010
- Store 011

I/O
in 100
out 101

+ OP code

الـ instructions عشان كل instruction ليها bus معين، فالـ instruction كانتها تروح للـ bus الخاص بيها

* الـ compiler بتفترض إنها بتكتب على الـ RAM وعملية الـ Write بتفترض إنها بتكتب على الـ flash

Harvard عشان الـ assembly تكتب الـ Software complexity أقل، الأبسط هو Von Neuman

هنا بقى ARM هيختار السرعة والبساطة !!!

نهاية L1 + البرنامج

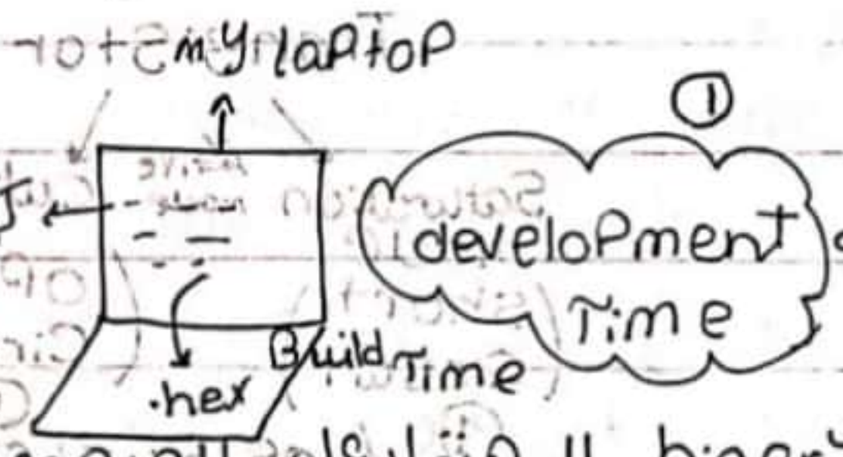
التسطيب باقي السيشن

Start L2

ازاى دا؟ لاحقا

② تانى element هوا Memory

وهو الجزء الهيشيل فيه اللود بتاعنا.



① * الوقت ال أنا بكتب فيه الكود اسمه development Time

خلصت الكود بعمل built

هى عملية تحويل الكود بالسى لـ binary format ال هيتحول على الميموري وهيكون

فيه extention كتير أشهرهم (.hex) وده اسمه build Time ②

هو عملية تحويل من C إلى binary، لسه كدا ال Microcontroller

لسه مظهرش، دا لسه على الكمبيوتر، hex دا ال محتاج (حرقه)

على ال Microcontroller فيجيب جهاز اسمه Burner

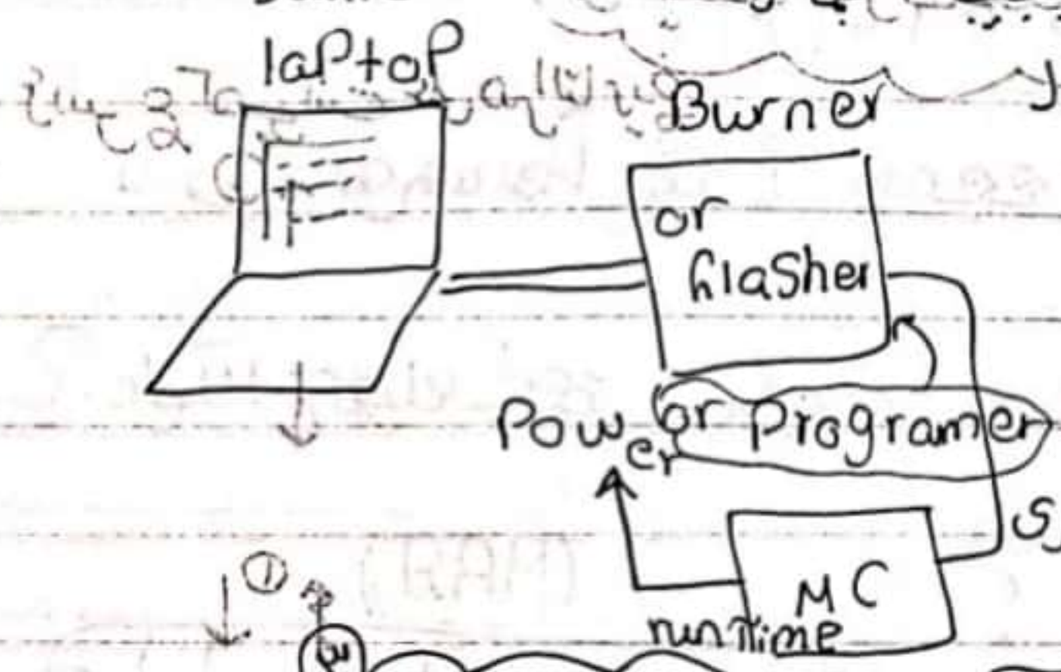
وده جهاز بيوصل مع اللاب بال USP وبيوصل مع ال kit

أو ال Microcontroller بتاعى

وبيبدأ يعمل download ①

للفايل hex. من اللاب إلى الميموري

ال داخل ال Microcontroller



وده اسمه Burning Time أو flashing Time

بعد كده هفصل ال burner بتاعى وأوصل Power على ال MC تانى وده اسمه

runTime ⑨ وهنا بقى يبدأ ال Processor

كدا أنا محتاجة Memory يكون عليها اللود.

* وهنا بدأت فكرة ايه ال electrical component ال

لو دايتها دايما تحتفظ بيها وهى تاب Memory

أ- أول فكرة جت في دماغهم ساعتها انهم يعملوا Memory based on ال Capacitor وهو عبارة عن خزان بيشحن ويفر

Memory

Based on Capacitor	Based on Transistor
وكل مكثف يمثل bit لو مشحون يبقى 1 لو مفرغ يبقى zero	Saturation mode (Short Circuit) ① / Active mode / Cutoff mode (open Circuit) ⓪ (as Switch)
charged / not charged	كل bit تبقى بـ Transistor لو في open يبقى (0) ولو short يبقى (1 one)

أيهما أفضل ؟!

* أسرع كمبيوتر على مر التاريخ ↑①

* Performance (سرعة)	يعني لو bit من 0 إلى 1 ↓
* Cost	↑①
* Size	↑ / ↑①
* Power consumption ↑↑	المقصود بيها ال Power المحتاجها علشان تحافظ على الداتا ↓↓①

لو حطيت كود على Saturation في ال Transistor لو بعد سنة هلاقيه زي ما هو ، إنما لو Capacitor وجيت بعد مثلا ٤ ساعات هلاقيه متفرغ فهو ليه leakage factor عالي معدل تسريب عالي ، فممكن بعد ما كان 1 بقى zero بالتفريغ ، وده محتاج اشحنه كل شوية ، فبحتاج طريقة اسمها refreshing Circuit

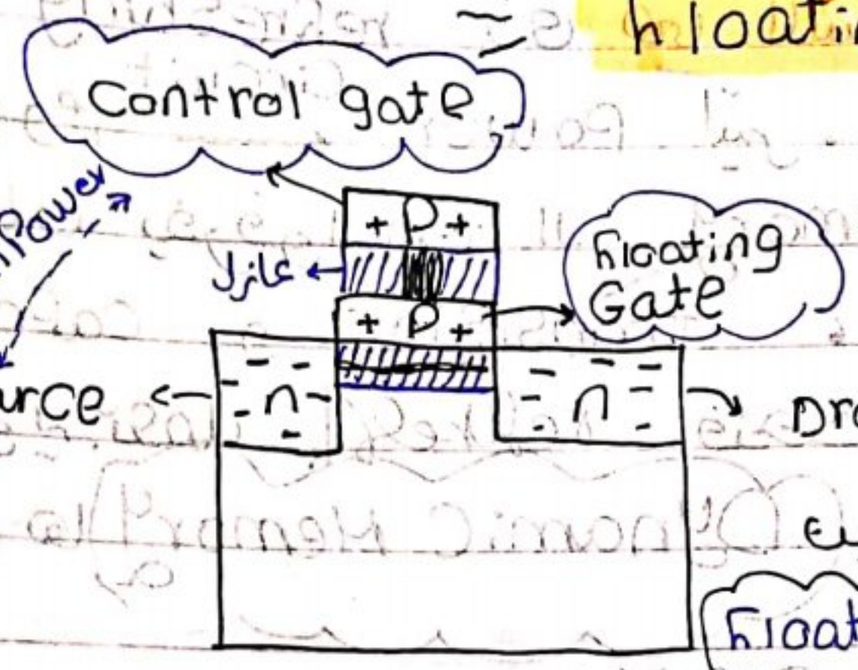
وهى عبارة عن دايرة مربوطة بـ Timer كل وقت معين حسب بها
ال design بتاع الميموري بتعمل كل شويه refresh
و refreshing دى هى السبب الأساسى إن ال capacitor flash
بيكنسيوم circuit power ليس.
* علشان يفرقوا بين ال Memory ال هى based on capacitor و Transistor ، فايما إن ال based على capa
كل شويه تعمل refresh فتحت إن فيه حاجه dynamic جوا محتاج
فسموها Dynamic Memory أو Dynamic Ram
(DRAM) (رخيصة)
أما ال هى based on ← Transistor فعلشان ثابته
سموها Static RAM ← SRAM
فكدا SRAM سريعة وحجمها أصغر لكن سعرها غالى

لكن الحقيقة هما الاثنين مينفعوش نحط عليهم اللوحة ← مشكلة ؟
(RAM) ليه ؟
ج: علشان هما الاثنين من نوع Volatile memory تب على ال
يعنى بيحتفظوا بالداتا طول ما فى كهربا متوصلة عليهم لو الكهربا غدر نعمل
اتشالت الداتا ال عليهم هتروح

طيب نعمل ايه ؟
غايزين electric Component لما أحطه فى وضع معين يفضل
محافظ على وضعه حتى لو شيلت من عليه الكهربا.

لحد ما اخترعوا electronic component عظيم جداً بتاع اسمه

Floating Gate MOSFET



فكرة بيشتغل ازاى

بعد الفكرة دى.

جابوا مادة عازلة وحطوها تحت

ال gate دى وكذلك فوق

ال gate وبقى محصور بين مادتين

عازلتين علشان كده اسمها floating gate

تحتها هواء وفوقها.

وبعدين عمل Gate تانية

* لو عملنا apply على ال Transistor دا High Power بنقدر نعمل Create

لـ Channel فى المادة العازلة تسمح بمرور الكهربا بشرط

apply ال high power بين ال Source و ال Control Gate

ولو شيلنا ال high P هتصبح عازلة تانى.

← هنوصل ال High P اهتموا electrons إلكترونات لدى

لحد ما توصل لـ floating gate وشيلنا الكهربا العالية فالالكترونات

اتحبست (Charged).

طيب عايزين نوع الشحنات توصل الكهربا صغيرة بين ال Drain و

ال control gate هتتقطع ولا تبقى not charge

فسواء كده أو كده ال MOSFET هيبقى على نفس الوضع الهملك

يوصل لـ 50 years ومنا بقى Non volatile Memory

بلغى بيعمل زى ما احنا حطينا component جنبها

ال charged ال not charged

يعنى فيه شحنات سالبة > 0 مفيهوش شحنات سالبة

0 1

مثال لو جيت وقولت هات ROM وامسح الداتا اللي فيها هيبقى فيها ones ولا zeros ؟

ج : هيبقى فيها ones يعني اعملها non charge زي الـ flash اللي على STM هلاقي FFF ---- يعني ones .

لكن هنا في عيبين !!!

← 1 بيحتاج high Power علشان اعمل create للـ channel

← 2 عنده high access time (بطيء) ماهو علشان أعمل كدا محتاج أعمل apply high power فترة معينة

ومن هنا قامت صناعة أنواع كتير من الميموري على نوع non volatile Memory

ROM (Read only Memory) based on وده

(MOSFET).

طيب ليه هي للقراءة فقط؟ مش أنا أقدر أشحنها وأفرغها؟

* المشكلة هنا وهو إنها محتاجة high Power طيب وايه المشكلة؟

هنا بقى الـ Processor مقصده provide not الـ high Power علشان يعني الـ Processor مش هيقدر يكتب على الـ ROM وفي كدا بالنسبة للـ Processor ← Read only لكن نقدر نعمل كدا من خلال الـ burner بواسطة الـ user .

أنواع الـ ROM based on MOSFET

1] Masked ROM

والنوع دا بييجي من المصنع متبرمجه مقدر أغير اللود اللي عليها

* أرخص سعر
* بتتحرق مرة واحدة بس وتزي مثلاً لود فنحوس رمضان

2] OTP ROM (one Time Programable)

يعني بيتبرمج مرة واحدة بييجي من المصنع مش متبرمج وأنا أبرمجه من خلال الـ burner مرة واحدة بس.

3] EPROM (erasable Programable)

يعني نقدر نمسح ونبرمج نمسح ونبرمج وهكذا

زي ما أنا عايزة بال Burner. (علشان بيعمل applying high power)

لكن بتتمسح بـ UV ultra vilit

أي ضوء أبيض ما ضوء الشمس زي مثلاً بتستخدم في الأجهزة المحتاجة protection زي آلات الحرب والأسلحة والحاجات الخطرة

4] EEPROM

Ellectricaly Erasable Programmable

يعني بتمسح بالكهربا وبتبرمج بالكهربا برضو.

5] Flash ROM

وهي كمان بالكهربا زي EEPROM ،

وهي زي الفلاشة

الفرق بين ال flash و EEPROM ؟

EEPROM | flash

R:

① Byte access (W/E) → write / erase

flash: block access (W, E) → erase — 100 byte or ... على حسب ال size

⇐ كدا أيهما أسرع؟

ex: flash access Time 10 Usec (Micro)

EEPROM 1 Usec

Case 1

↳ write 1 Byte

↳ on EEPROM → 10 Usec

↳ on flash → 10 Usec

فكدا ال EEPROM أسرع.

Case 2

↳ write 100 byte

↳ 100 byte بتاعها block ال flash وكانت ال

↳ on EEPROM → 100 Usec

↳ on flash → 1 Usec

كدا ال flash أسرع

فالإجابة: على حسب إنت عايز تكتب ايه، لو عايز تكتب حاجات صغيرة جدا byte بـ byte، فالأسرع EEPROM، لكن لو حاجة كبيرة بالجيجا زي 1000 بالميجا يبقى ال flash أسرع

باقي المقارنة :

	EEPROM	Flash
Cost	↑ لأن عدد الترانزيستور أكبر بكتير من Flash	↓
Endurance: (life Time)	100000 : 1000000 W/E write / Erase cycle	10000 W/E cycle على نفس الـ block

ملاحظة:
الاتنين مصنوعين بالـ floating MOSFET لكن ترتيبهم في كل واحد مختلف ، علشان كده تلاقي الترتيب دا بيسمح في الـ EEPROM إنه يكتب byte ← byte.

Memory
- Volatile (RAM) 2 type
  - DRAM
  - SRAM
- Non Volatile (ROM) 5 type
  - Masked ROM
  - OTP
  - EPROM
  - EEPROM
  - Flash

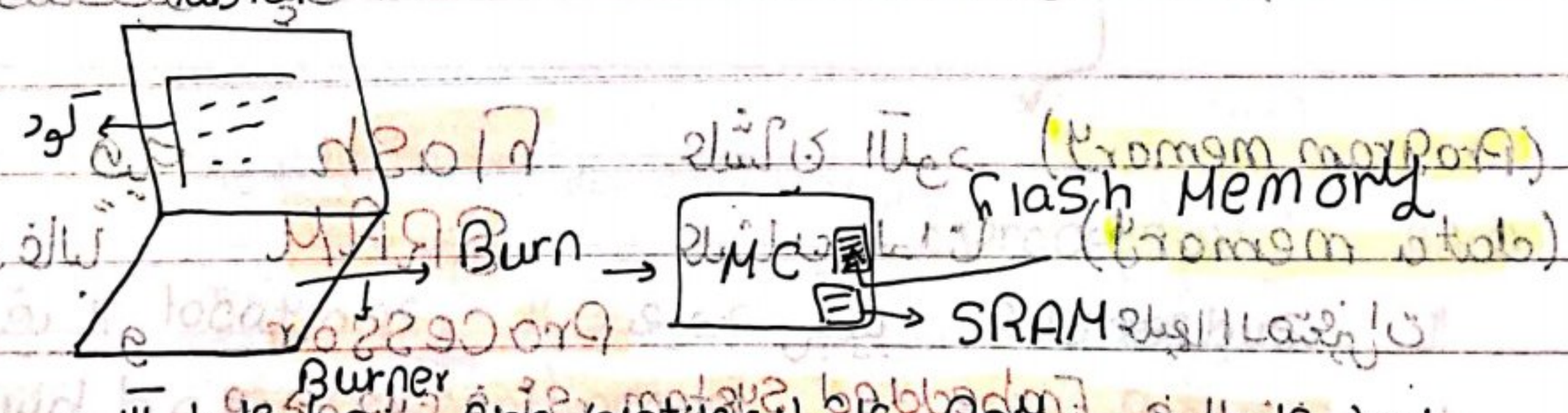
Embedded System. أيهما أفضل في عالم الـ

تعالوا نشوف الأنواع المميزات

عند كل نوع

	Volatile Memory	Non volatile Memory
Retention Time		50 Years
	- Fast - Addressable write يعني الـ Proc يقدر يكتب عليها من خلال عنوانها.	

هنا سؤال نستخدم أنهي واحد فيهم في الـ Micro controller ؟



فهنا عايزين ROM علشان Non volatile نحط عليها الكود.

طيب أنهي نوع؟ فيها؟

* الأكثر منطقية هو نهاجه أكثر براحتي وأحط كذا مرة و هكذا

فكذا يا إما EEPROM يا إما Flash لكن الكود هنا كبير جداً

فالأفضل هنا الـ Flash أسرع وأرخص لكن الـ WEM الـ 10000

∴ الـ Flash هي الاختيار الأمثل لأي Micro Controller

لكن أنا بحتاج ميموري سريعة وأقدر أكتب عليها أثناء الـ runtime
يعني باستخدام الـ Processor وده طبعاً هيخليني محتاج ميموري
من نوع (volatile) طيب أنهي نوع ؟؟

SRAM — DRAM

الـ SRAM أسرع وأصغر و Power consumption
قليل لكن غالية وده عادي. وهي الأفضل

* لكن الـ DRAM موجودة في عالم الـ PCs والموبايلز
* DDR 1,2,... يعني نوع الـ RAM (Data
اختصارها double data buffer dynamic RAM)

يعني في اللاب توب بقى محتاج RAM كبيرة زي 4G أو 8G
لو هاجبت الحجم ده في الـ SRAM هلاقيه سعره عالي جدوداً

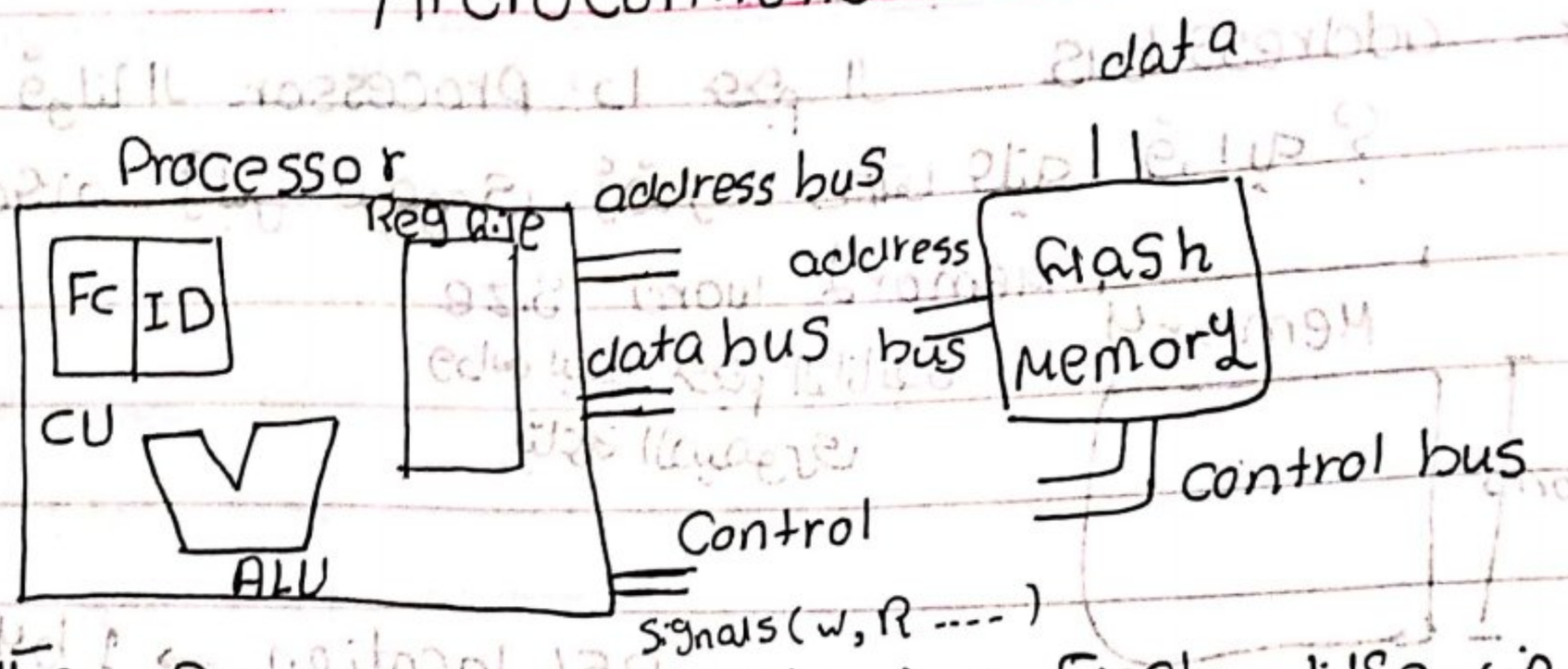
كده متفقين إن أي Micro controller

فيه Flash علشان الكود (Program memory)
وغالباً SRAM علشان الداتا (data memory)
و Processor عليها المتغيرات
موجودين في أي Embedded system في الدنيا

هل عندي
① why we may need EEPROM?
نعم ليه كلمة شويه
* الـ ARM مفيهوش EEPROM

② How to write on flash @ RunTime?
آه ، ازاي لسه هنعرف قدام في آخر ٣ محاضرات
فيه وسيط

MiCroController Architecture



هنفترض مكانا Flash بس ما ازاى ال Processor هيكلم ال Flash ؟

علشان يعمل Connect مع ال Flash بيحتاج 3 buses وال Flash عندها نفس ال buses والمفروض بيركبوا على بتوع ال Processor

*Signals of Control bus

1* Read/write
2* Clock
3* Enable

هل ال Processor يقدر يعمل connect مع أى Flash ولا فيه limit معين بقى؟

هنا ال Processor :

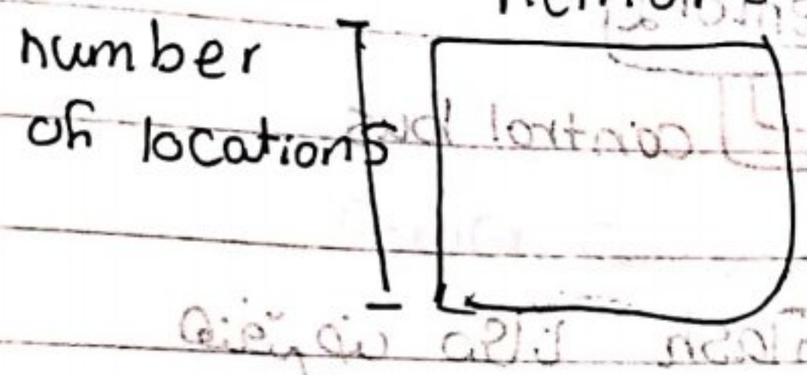
① ال Address bus يعبر عن عدد ال locations ال يقدر ال Processor يقدر يـ access (Could be accessed)

② ال data Bus أو (Processor word size) يعنى حجم الكلمة قد ايه ؟ (number of bit at Access Time)

③ Control bus ---

طيب لو قولنا ال Processor دا حجم ال address bus ← 8bit
و دا معناه أكبر ميموري أقدر احطها عليه قد ايه ؟

Memory word size
وده بيمثل حجم الداتا بص
بتاعة الميموري



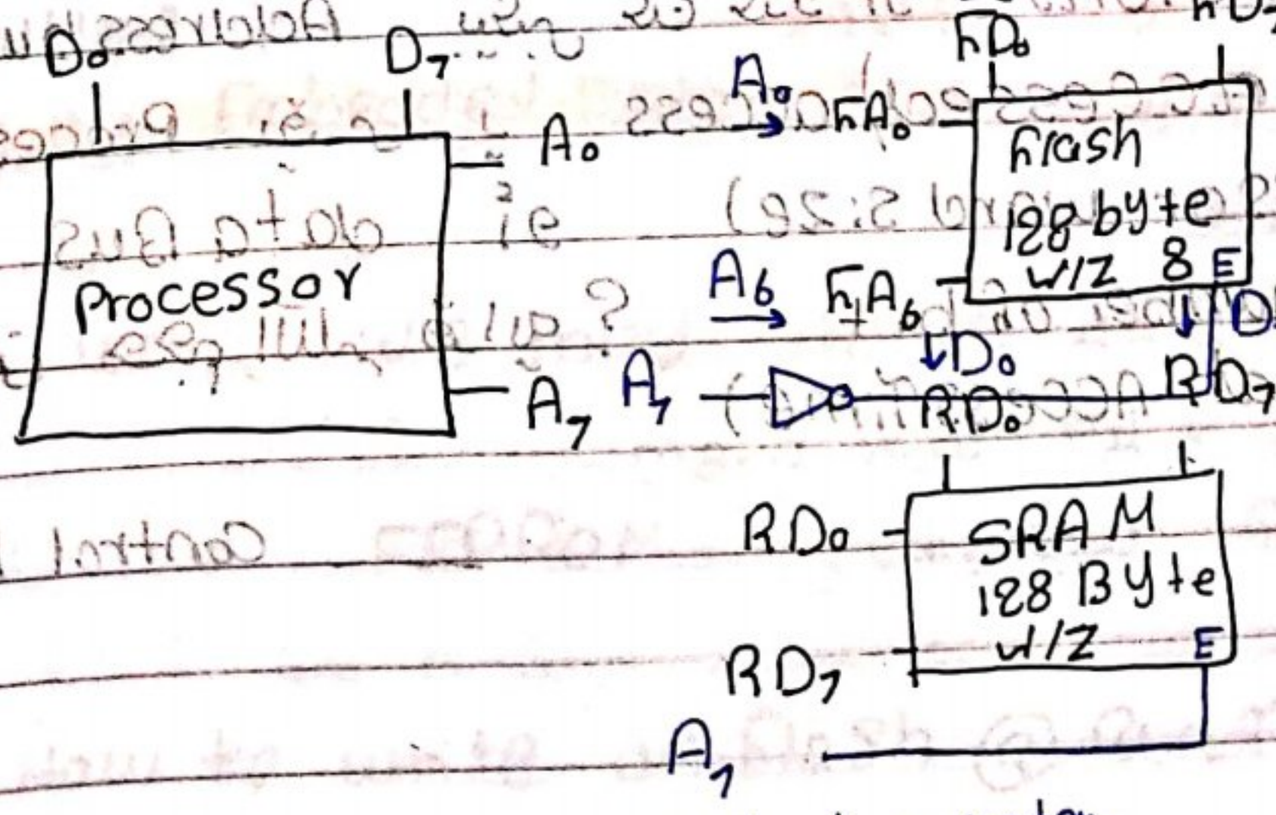
256 location Byte
- → 256 location → 1 byte / Locati
- → 128 L → 2 Byte / L
- → 64 L → 4 Byte

فلو قولنا زي الأول ال Processor ← 8bit
إنه بقدر يعمل access على 256 ← locations
طيب كل location بقى قد ايه ؟ ؟ : حا يبقى حجم ال data bus
فلو 8bit هو كمان يبقى محتاج 256 ← locations
وال data bus = 8bit .

Memory
يعني : address bus = 8bit

طيب لو

Processor
- → 8 bit Address bus ⇒ 256 location
- → 8 bit data bus ⇒ 1 byte



يعني لو حطيت على أصفار على A_7 هيجيب أول location في ال flash ولو 6 أصفار وواحد هيبقى تاني location.

	A_7	A_6	A_5	A_4	A_3	A_2			
أول loca... في ال flash	0	0	0	0	0	0	0	0	flash
أول آخر loca... في ال flash	0	1	1	1	1	1	1	1	
أول loca... في ال SRAM	1	0	0	0	0	0	0	0	SRAM
آخر loca... في ال SRAM	1	1	1	1					

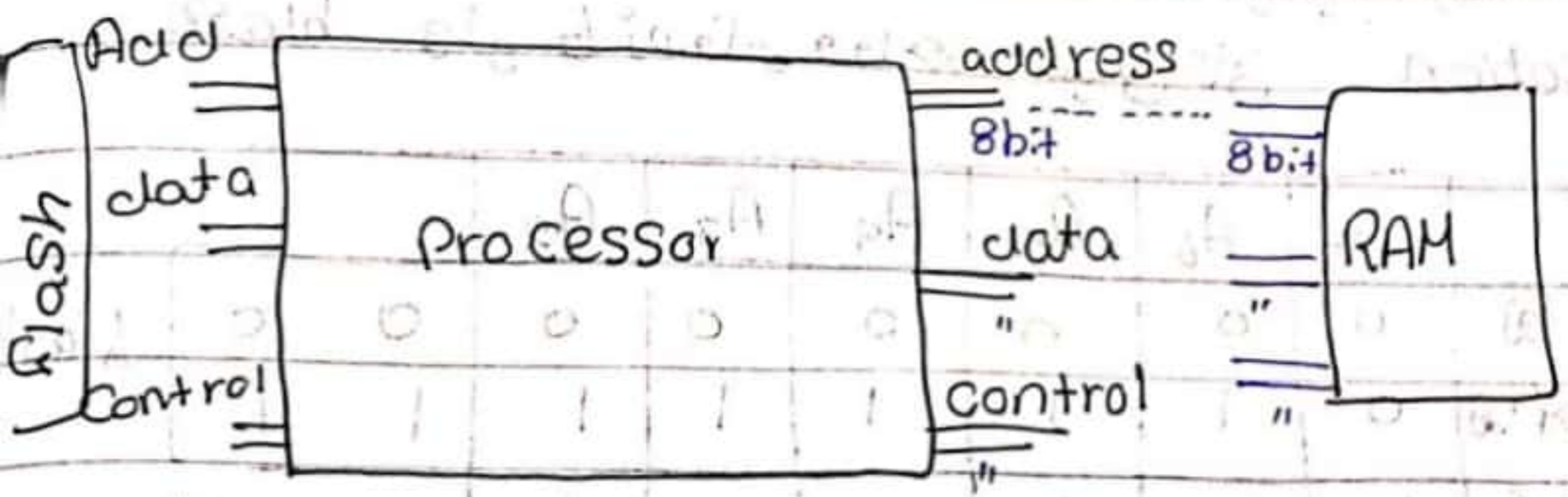
لو أنا بقى ك Processor طلعت على A_7 ← zero هنا كدا A_7 هتبقى 1 بعد ال inverter فكدا flash هتبقى enable و SRAM disable وباقي الأصفار هتدخل على ال flash وتأكسز location في ال flash.

ودي بقى طريقة Von Neumann

المهم تكون عدد ال location مناسب لل Processor حتى لو ال Processor ليه addresses زيادة بنسميها resarved address

Von Neumann
- addresses are incremental يعني ورا بعض
- only one Peripheral enable at Time يعني مش Pipelining

Harvard



Address map table :

ال addresses مش incremental

يعني

Flash 0000 ... 0000

:

Ram 0000 ... 1111

يعني على ال lines مختلفة

Harvard
↳ Pipelining
↳ Not incremental Addresses

العيب الأساسي في Harvard هو ال Software Complixity

هوايه بقى؟ ↳ different Instruction Set.

هنا زي ما قولنا int *Ptr = 0

هنا قصدك أن zero في ال addresses

لكن في ال C أي أدرس بالسنبالي هو على ال RAM.

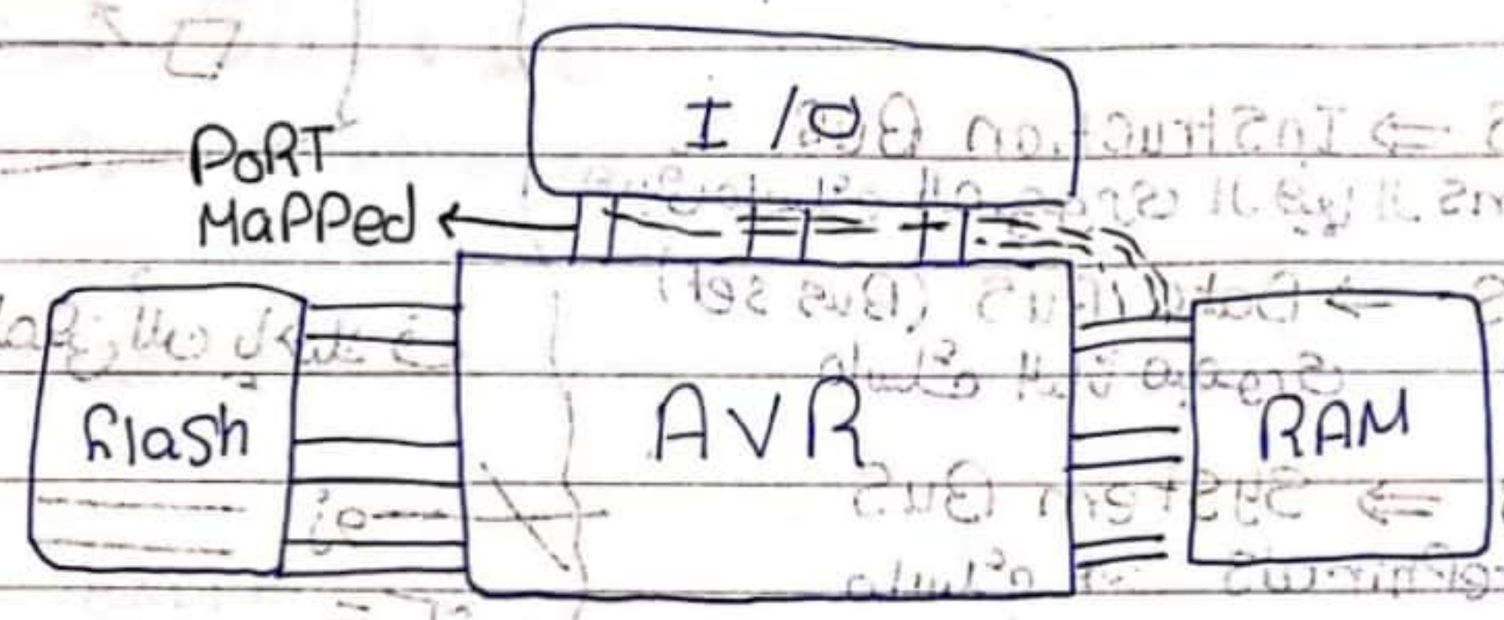
طيب لو عايز أكتب على ال Flash ، هنكتب Assemply

ماهي ملهاش حل.

لأن في Assembly instruction لكل واحد فيهم مختلف عن التاني زي

RAM	Flash
* load	* Read
* Store	* write

في ال AVR عملوا ايه علشان أقدر أكلم Registers زي PORTA وهكذا بالسي رغم إن ال addresses بالسي بتحولها لل RAM على طول؟



مثال PORT A = { 0x50 (0x20)

0x50 ← Port MAPPED

0x20 ← Memory MAPPED

يعني حط I/O لو مع ال RAM اسمها Memory map لو حطيتها PORT MAPP ، بالتالي كل register هيبقى ليه 2 addresses

* (0x20) = ✓ ، فلو عايز تكتب

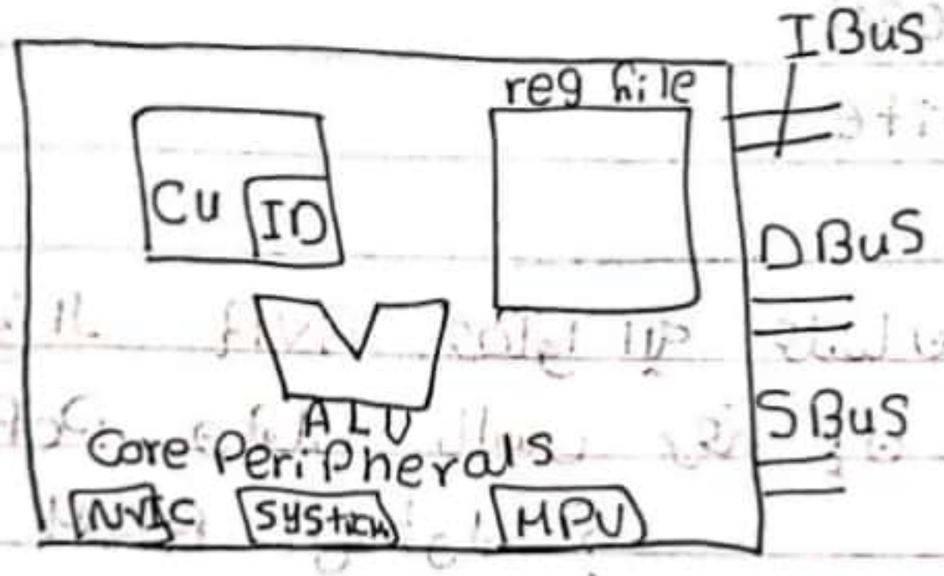
أو * (0x50) = ✓

Assembly

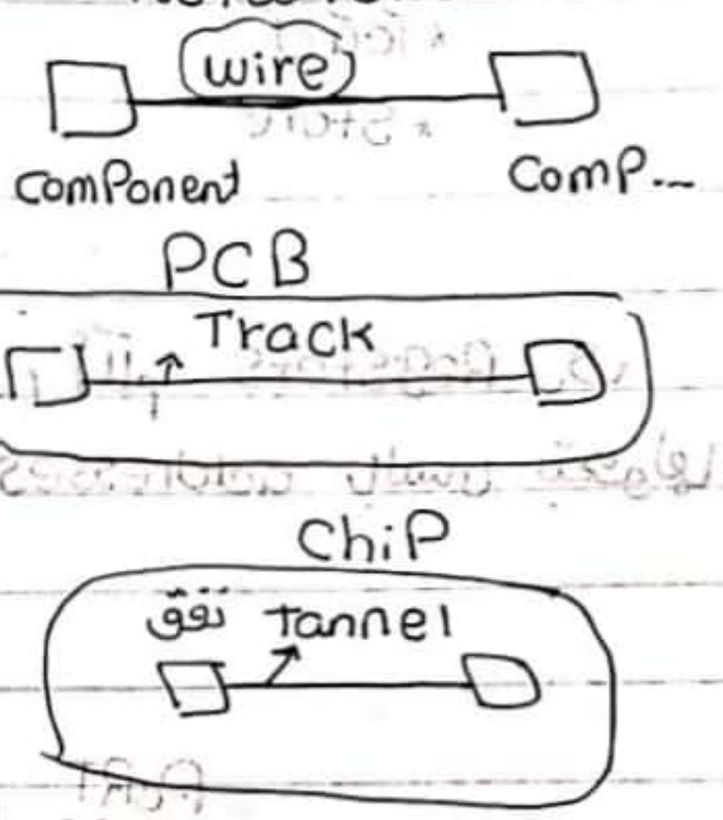
* (0x20) = ✓

لأن لو كتبت بالسي هتحصل فكرة ال Pipelining و هتكتب على ال RAM فعلشان اكتب على ال I/O لازم اكتب Assembly على (0x50) ~~فكرة إن الكتب من خلال (0x20) هتحقق ال Pipelining~~

ARM Cortex M3



Notation



IBus ⇒ Instruction Bus
instructions هيبقى ماسك الميموري ال فيها ال

DBus ⇒ Data Bus (Bus set)
ماسك الداتا ميموري

SBus ⇒ System Bus
ماسك ال Prephirals

هو harvard لحد دلوقتي

هما نفس المعنى لكن باختلاف المكان.

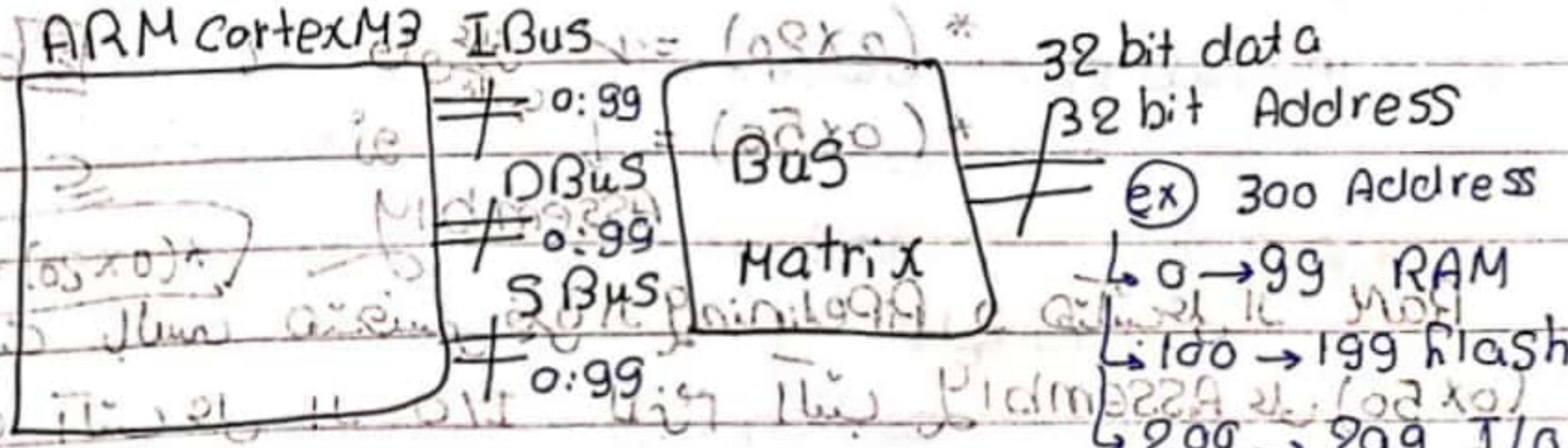
Bus ⇒ زو

معناه مجموعه من الكونكشنا ثلاثة من سلك

ولو # Bus set

Bus Matrix

لكن ARM عملت Chip عظيمه جداً اسمها باخد الثلاثة buses ودخلهم ع bus كبير



طيب وهنا هو كان في الأصل harvard وبعدين بقى في الجزء التاني
بتاع ال Bus Matrix وكأنه بقى von Numann

*

هلا عندي هنا 300 Address من بعد ال Bus Matrix

RAM ↳ 0:99
flash ↳ 100:199
I/O ↳ 200 : 299

هنا كتوصل معلومة بس وبعدين هتوضح الصورة

* تخيل كدا إن كل واحد من IB ، DB ، SB كل واحد ليه واحد
من RAM أو flash أو I/O وليه 100 address زي الرسم الفاتت

فهنا ال Bus Matrix

لأنها بتحول ال addresses من Range قبل ال Bus Matrix
إلى Range ما بعد ال bus Matrix

هي مش بتعمل fetch و decode و excute لا هي
بتعمل ال translation في وقت بيكافئ ال fetch أو
decode أو ال excute ، يعني بقدر دا لله في فترة واحدة و
Clock واحدة

يعني هلا ال IBus هتطلع address فال Bus Matrix تحولهولي و
تقوله روح اعمل fetch ، DBus تطلع address فال
Bus Matrix تحولهولي وتقوله روح اعمل fetch فيكون IBus
خلص fetch هيدخل في ال decode

فكدا إنا كي user للـ Processor استخدمت التاريخ الموضوع
من بعد ال Bus Matrix وكأنه von neumann توأقدر
أكتب C بمنتهى السهولة

ملاحظة: ال Buses (I, D, S) هما :
not Accessable for user

(ال أنا بشوفه بس ال Bus Matrix فبتخيله إنه von neumann لكن

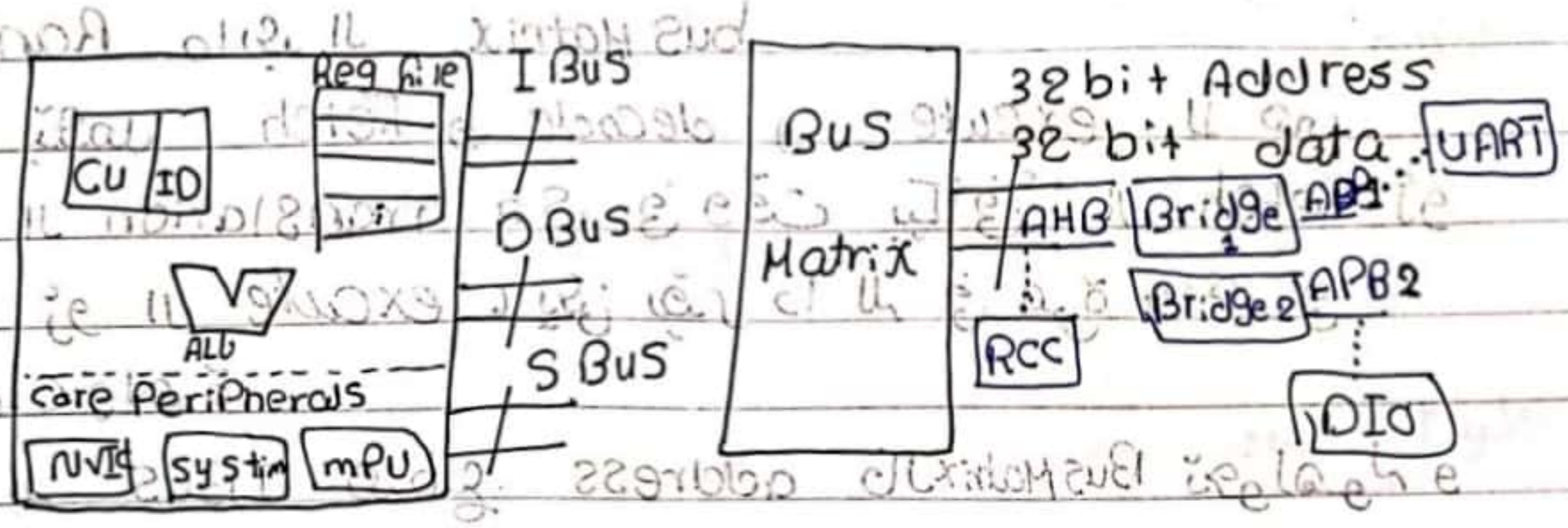
هو في الحقيقة من جوا Harvard وادا السحر الاعملة ARM
علشان يجمعوا بين الاثنين (سريع (Pipering)، سهل في ال Coding
لأن باين Von neumann .

فهو الأصل Harvard وال addresses بتكون incremental علشان ال Bus Matrix .

End ?

Lec 3

ARM Cortex M3



بعد هنا دور ARM بينتهي ويرجع بعدها ال Vector يبدأ
يحط ال PerPhirals بتاعته من RAM، Timer، Flash...
لكن في ARM مستنى قال هعمل قواعد يركب فيها ال PerPherals
بتاعته، هعمل Protocolls يركب بيها ال PerPherals
متوضح فيها شكل التوصيل وغيرها

AMBA

في ARM اخترع حاجة اسمها
(Advanced Microcontroller Bus Arctichture

(27)

وهو علم هنو ضعلك فيه Protocalls ال إنت مقلت توصل من
خلالها ال PrePhirals بتاعتك ، بيكلم ال Vector إنه يستخدم
واحد أو اتنين من ال Protocall علشان يوصل ال PrePhirals
* AMBA فيها تقريباً (5 Protocalls) يهمنا منهم
اتنين بس وهما:

AMBA

AHB — APB

وغيرها ....

دول الموجودين في ال MicroController بتاعنا

AHB: Advanced ~~High~~ High Performance Bus

APB: Advanced Peripheral Bus

① * AHB:
بيدي أعلى Performance في ال AMBA وهو AHB
* High Power
② * Complex design
* full duplex RW يقدر يتكلم ويسمع في نفس الوقت
* Multi Master يعني ممكن أكتر من 2 يقدروا يتكلموا على ال Bus دا

APB: PerePhirals. هدول هيستخدموا منه
* Low Power
* less complexity
* Lower Performance

CamScanner

الـ Bus الطالع من الـ Bus Matrix شغال على AHB
وراكب على الـ AHB دا أهم الـ PerPhirals اللي عايز Performance عالي
زي RCC والـ PeriPherals دي مش مينفع AHB لا عملاها
لأدى الـ عملها الـ Ventor لبيعة لية AHB
كدا الـ PeriPherals دي كدا هتكسيوم Performance عالي فكدا هيمكنها
الـ PeriPherals نفسه هيبقى محتاج لأن هو أساساً بطيء
فبيتعمل دايرة اسمها Bridge وهي دايرة بتحول من Protocall
إلى Protocall تاني ، وفي 2 Bridge (1, 2)
Bridge 1 واخد من AHB وبيطلع APB1
و 2 Br... " " " " APB2
سبب هما الاثنين مش بنفس السرعة APB1 أبطء من APB2
وفيه كمان PeriPheral اسمه UART متوصل على APB1
و DIO متوصل على APB2
ملاحظة:
① الـ APB2 سرعته بتصل لـ 72 Mica أما APB1 بتصل 36 Mica
، AHB بتصل لـ 72 في الـ Mica
② جزء الـ Bridge اللي في ARM هو العلاه
Documents → ST: Register Description
ARM: Programming Manual

* Register Description:
هي الـ PeriPherals اللي عملاها الـ Ventor ST هتلاقي فيها
كل المعلومات بتاعتها جواها وهكذا ولو عايز تشغل حاجة
الـ ARM عملاها ترجع على Programming Manual علشان
تعرف

مثال :

عندنا مثلا Perphiral اسمه SYS tic جوا ال ARM

فهنفتح ال Programming manual

* لو أنا قولت نشغل ال Periphera ال اسمه كذا ما لازم أول

خطوة نعرفها هو شغال على أي Bus ①

* هنعرف دا من document بتاع ال ST

أي شركة في الدنيا لما تيجي تعمل Software بتمر بمرحلة اسمها

Software development cycle (مهمة جداً)

أولاً في Customer وراح الشركة وعايز يعمل System معين

يبدأ ال Customer يتواصل مع الشركة وبعدين الشركة تقول

① محتاجين نعرف التفاصيل بتاعة ال System دا.

فبتبعتله شخص اسمه Analyst

بيروح يقعد مع ال Customer ويقوله عايز تعمل ايه؟ وبياخد كل

تفاصيل ال Project اللي عايز يعملها ويكتبها في document

اسمها RFQ ال اسمها مملكة يختلف من شركة للتانية

اختصاره (Request for Quotation)

عادة ال Analyst مش بيلاقي مهندس وال Customer

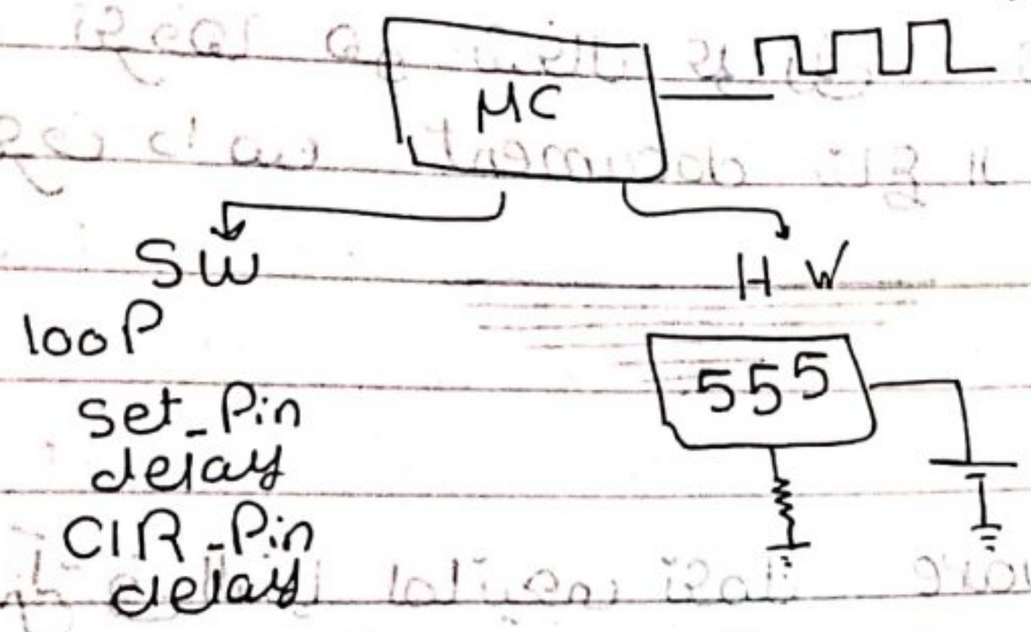
بيسألنا هنا حاجتين هنعمله؟ بكام؟ وفي كام وقت قدايه؟

بعدين يبدأ يتحول ال RFQ لحاجة اسمها CRS

(Customer Requirement Specification)

وهي document بتشرح ايه ال Specification

الـ customer عايزها ، لأن RFQ مكانش مكتوب بشكل Technical كفاية ، لكن CRS بتكون Technical أكتر
و بيعملها التيم الهينفذ الـ هو عرض على الـ customer السعر والوقت
و CRS بيبقى فيها requierment كتير جداً
و تقابلنا مشاكل كبيرة جداً إن الـ requirement دي هتتعمل بالهارد
ولا بالسوفت زي مثلا



ومن هنا يجي الـ System Engineer وهو شخص
أكتر حد خبرة وهو الـ هياخد القرار على مستوى الـ System
ولـ requirement هيتعمل بالـ Software ولا الـ Hardware
ومن هنا هينقسم الشغل بتاعنا لحاجتين واحدة اسمها SRS
والتانية HSI
و SRS (Software Requirement Specification)
و HSI (Hardware Software Interface)
والـ بيعمل التقسيم دي هو System Engineer والموجود
في الـ SRS كلمة Software محتاج كود يعمل كذا
وفي HSI دي هنا بقى الـ MicroController نوعه ايه والـ Chips
عاملة ازاي والـ Connections بين الـ MicroController وبين
Chips عاملة ازاي وكل مقاومة كام وكلام دا كتير .....
غالب الشركات في مصر (الكبيرة) بيبدأ دورها من عند SRS
وفي اللحظة دي بيبقى معانا document فيه المطلوب Software
requirements

ومن هنا أول خطوة في كتابة ال Software
نبدأ نقسم ال SW لمجموعة من الأجزاء الصغيرة وكل جزء من
الأجزاء دي هيعمل حتة من ال SW وبعدين هنركب الأجزاء دي
مع بعض ونعمل ال SW الكبير ال بيعمل القصة دي شخص
اسمه Architect هو المسئول عن تقسيم ال SW الكبير
إلى SW صغيرة ونحدد بقى كل SW من الصغير إيه ال function ال هيبدأ
بتاعها ويعمل إيه ويتاخد إيه من ال Parameters ويرجع إيه والوقت
والمنطوري وكل حاجة لأنها إطار عام للـ SW
والشخص دا بيقوم بهما حتين واحدة اسمها Static Architcture
والتاني بالـ Dynamic Architectur

نبدأ بال

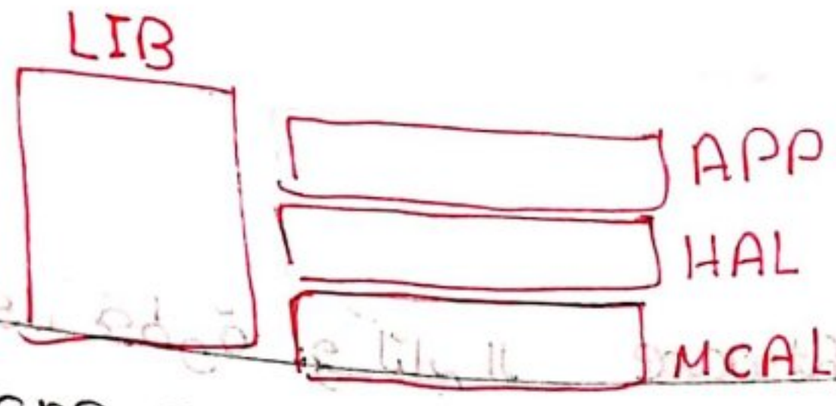
Dynamic Architecture :

(تنفيذ runtime)

بيعملوا ال Archtect لكل function عندك
متاخد excution Time وقت قدايه
و RAM consumption يعني فيها
variables قدايه في الرام
و Flash consumption وده كله بيقدمه
في document مهم جداً اسمه
HLD (High level design)
فال Architect واحد من أهم
ال Position المطلوبة في مصر
ومن هنا نبدأ ندي ال SW
الصغيرة دي يعني كل component
لكل مهندس علشان ينفذها
* قبل ما نبدأ تنفذ كلنا بنمر
بمرحلتين

Static Architcture

↳ (SW) أجزاء بيقول فيه إيه اللومبونتس
العندك في ال System وبيرسم
القصة دي في حاجة اسمها
layered Diagram
يعني بيقسم السوفت وير بتاعي
لمجموعة من ال layers
وهي وهمية لكن تنظيمية
أول layer من تحت اسمها
MCAL (Microcontroller Abstraction layer)
ودي هنحط فيها ال Software
الخاص بأي Peripheral جوا
ال Microcontroller زي ال DIO
وال ADC، فوقها layer اسمها
(Hardware Abstraction (HAL) layer)
وهيكون فيها drivers بتاعة أي هاردوير
برة ال Micro زي Motor مثلاً أو
أي سينسور أو ال wifi



ملاحظة:

Software Component ≡ Software Driver ≡ SW Module

يعني جزء من ال SW

Static Dynamic

مرحلة أساسية ومرحلة اختيارية وبعدين فوقهم layer اسمها

الأساسية اسمها: Component design (APP) (APPlication)

وال بيقوم بيها اسمه Designer هي الاهتمام بالناس دي كلها ببعضها

بيبدأ لما يمسك كل Function الفاهمة اللي كمبونتس ببعضها وال layers دي

ال Architect ويرسمها Diagram ثابتة لكن ممكن تختلف بزيادة وهكذا

ويقول هي هتشتغل ازاي وأشهر ال APP ال main

* مثلاً Function اسمها Set Pit في قاعدة بتقول إن أي كمبونتس

بتاخد رقم ال Channel value في layer مقدرش يكلم غير ال

هنا انتهى دور ال Architect layer ال تحته بس حتى لو APP ممكن

يجي ال designer ويقول هعمل condition هي تكون أكبر من يكلم MCAL. الكلام دا يعني call fun

num of channels لو yes في بقي layers اشهرها اسمها horizental layer وهي LIB library

هيعمل End لو لا يبقى بعد وامختلف عن ال vertecal لأن فيها

تمام و يعمل check على كمونتس من حق أي حد يكلمها من أي layer

value لو high يعمل ... مش من حقها تنادي على أي حد.

أشهر حاجة اسمها Standard types مش int لو low يعمل set Port

وبعدين end داكلم على وزي Bit Math زي الابتعل set و clear

شكل flow chart علشان ... وداني أول مرحلة بيقوم بيها الـ Architecture

يوضح ازاي نكتب الكود. ويبدأ يرسم ايه الكمبونتس اللي عنده في

ولازمه وال Diagram ال System ويرسمها في layers زي الفوق كدا

دا شرط يحقق الشرط ولكل كمبونتس ايه ال functions ال فيه و

كمموري وكوهت وال اسمها و ال Arguments وايه return.

designe بيستخدم لغة

اسمها UML وهي معهم ذكرها مع نفسنا

(Unified Modeling language)

عبارة عن شوية رسومات

علشان مش depended على لغة برمجة معينة

نفهم الحاجات دى مهم جداً

- Flow Chart (Activity Diagram)
- State Machine
- State Chart
- Sequence Diagram

والـ designer بيحط عمله دا في document اسمها CDD
Component Design Document
CDD بنعمل لكل component ؟
جاهزين للكتابة للكود؟
لسه بقى فيه بعض الشركات بتحب تحط حاجة في النص كدا
اسمها DDD يعني سودوكود يعني comments وبعدين
يحطوا تحت كل comment سطر الكود بتاعها
DDD (Detailed Design Document)
* وآخر مرحلة هي الـ code والـ بيعمله هو والـ DDD
الـ Developer وبيكون فاهم كويس أوى C
* بعد هنا خلصنا الكود بتاعنا

* ملاحظة:
غالباً الناس الـ بتشتغل في MCAL و HAL اسمها
Firmware Developer
والـ بيكتب في APP اسمه Application Developer
وبعض الأحيان دا بالذات ممكن شرط يكون مهندس خريج هندسة
كويس فاهم

بعد كدا مراحل ال Testing :

1) unit testing بيعمله ال (Developer) مع آخر حاجة بنكتبها

unit → function

فبنمسك كل function ونعملها Test و نبعتلها ال argument

بتاعتها ونشوف هتطلع صح ولا لا

2) Component testing بيعمله (Tester)

وهنا ناخد كل ال functions بتاعة ال Component الواحد

ونعملها Test

3) Integration testing بيعمله (Tester)

وهنا بنعمل Test لكل ال Software مع بعضه

* كدا اللوجكلر مظبوط، لكن فاضل ال Hardware

4) Validation testing (Validator) موجود في مصر عادي

وهنا بنثبت ال System على ال Hardware

* لو ال System مشتغلش في الآخر مع ال Customer

هما أخطر اثنين ال Architect و Validator.

هنا بقى في الكورس هنركز على ال Developer و Architect

ملاحظة :

1) CotS : Component on The Shelf

2) Cash Memory

هي ميموري موجودة جوا ال processor نوعها SRAM دورها

بشيل فيها داتا مهمة شبه شوية GPR لكن هي أكبر بالكيلو بايت

أو جيجا بايت ، دورها أسرع من ال RAM لأنها أقرب من

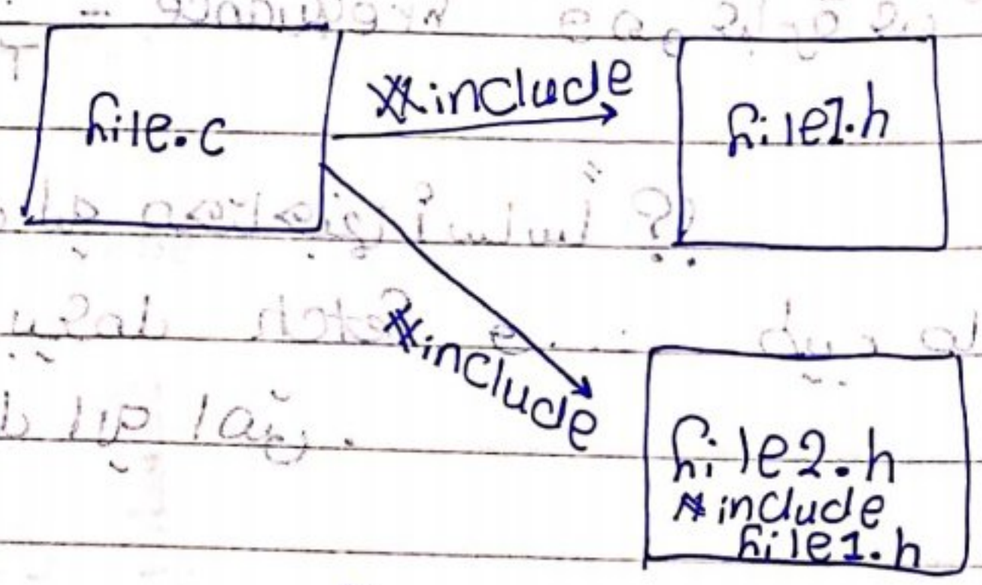
البروسيسور ، وأبطأ من ال registers هنعرف السبب قدام

الداتا سريعة ال access سواء read أو write.

Configration Management [٣]
أشهرهم في الشركات

↳ GIT
↳ SVN

Tools بيعرفونا ازاى تعمل Save للكود بتاعك بال versions بتاعته من غير ما تضيع



* وهنا file2 عامل include لـ file1.h
وكمان file.c موجود فيه file1 مرتين ودى كارثة
فبدأ الـ developer يعملوا طريقة تحميه من إن الخطأ دا يحصل
اسمها Header file guard هو تكنيك لو عملته الخطأ دا
فأول include بس هو اللى هينفذ والباقى لا، الطريقة ازاى
#ifndef Hamada (infile.h)
اسم →

معناه هو إنت متعرف حمادة؟ لو ميعرفهوش يعنى True هينفذ
لو يعرفوا هيخرج برة if ، وفى الآخر
#define Hamada لأن أول مرة آه ميعرفهوش

* كل حاجة بيكتبها
Code
#endif

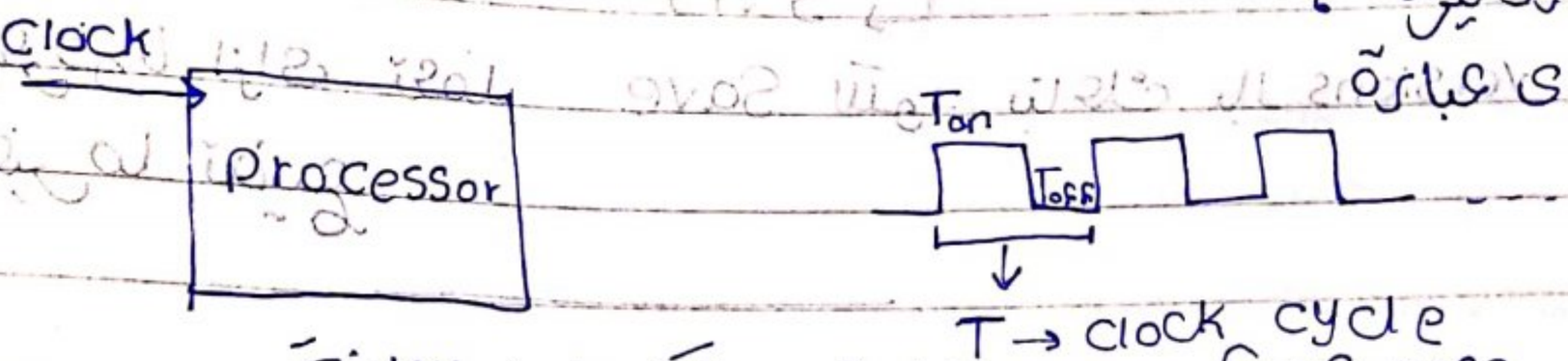
PreProcessor directory

* شغل Software

Clock systems

أي Processor في الدنيا عشان يشتغل بيحتاج Clock سواء ARM أو غيره ،

Clock دي عبارة

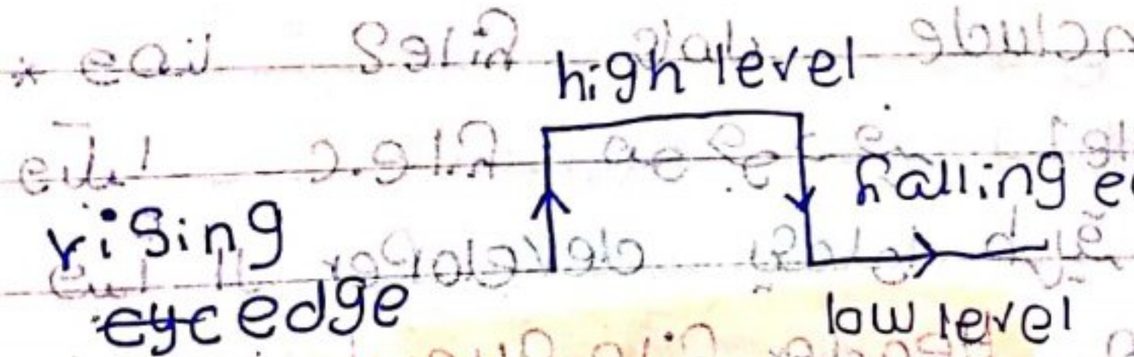


لو $frequance = \frac{1}{T}$ وهو عبارة عن تكرارها في الثانية .

* طيب ليه محتاجينها أساساً ؟؟

هو بيعمل fetch و ... طيب مالازم حاجة تعرف (Processor) يعمل ايه امتى .

cycle



كل واحدة فيهم اسمها ~~cyc~~ edge

Micro cycle وفيه Processor بيستخدم كل Micro cycle علشان ينفذ حاجة معينة ، فالـ cycle الواحدة بيحصل كذا حاجة وكذا اسمها clock used for synchronization

يعني ايه تزامن ؟

إزاي هنعمل الـ clock ؟

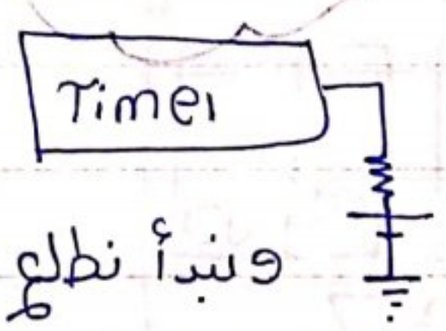
* الإجابة قادمة ...

Clock System

هي في الدنيا نوعين والمقصود بيها ال System ال بيطلع Clock

Electrical

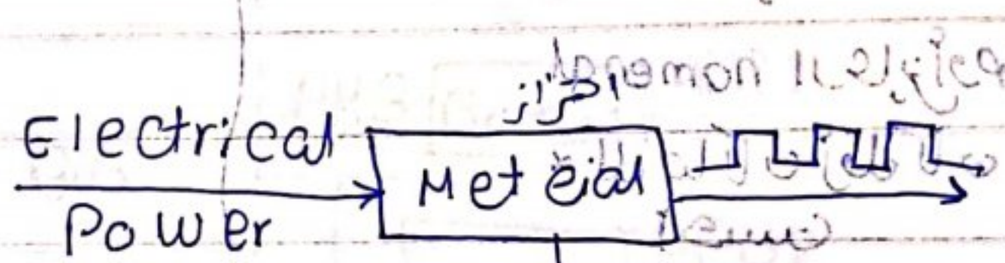
هي دايرة RC
R → resistor
C → capacitor
وهي عبارة دايرة فيها 555 زي Timer



وبدأ نطلع ال Signal
بتاعتنا اسمها electrical
علشان عبارة عن electrical component والمكثف بيشحن ويفرغ وهكذا

Mechanical

هي مادة ربنا خلقها في الطبيعة
لما نعمل apply عليها فولت
بتهتز والاهتزاز ده viprate
بنعمله او square wave



المادة دي

Ceramic Resonator — 401 — Crystal oscillator

والاتنين هزهم بردو

هما الاتنين ليهم طرفين واحد بنأبلاي
عليه كهربا وطرف بناخد منه ال clock
زي الكرستالة اللي عارفينها

اختيار واحد من ال 3 بيعتمد على ال System بتاعنا :
مقارنة

	electrical	Mechanical	
	RC	Ceramic Resonator	Crystal oSCillator
Cost	أقل سعر ↓	وسط ~	↑ أعلى سعر
ACCuracy	↓	~	↑
Settling Time	↑	~	↓
Noise Immunity	↓	↑	↑
↳ Temperature noise	↓	↑	↑
↳ EMI noise	↓	↑	↑
↳ Vibration noise	↑	↓	↓

يعني Settling Time الاستقرار وهو الوقت اللي بتاخده من $f=0$ لحد ال nomenal الحاجة أوصل

* كل ما كان أقل كان أحسن

الأفضل

هنا ↓ بتتأثر جداً

صناعة ضد النويز (الأهم في المقارنة كلها)

بسبب المقاومة والمكثف بتتأثر بالحرارة

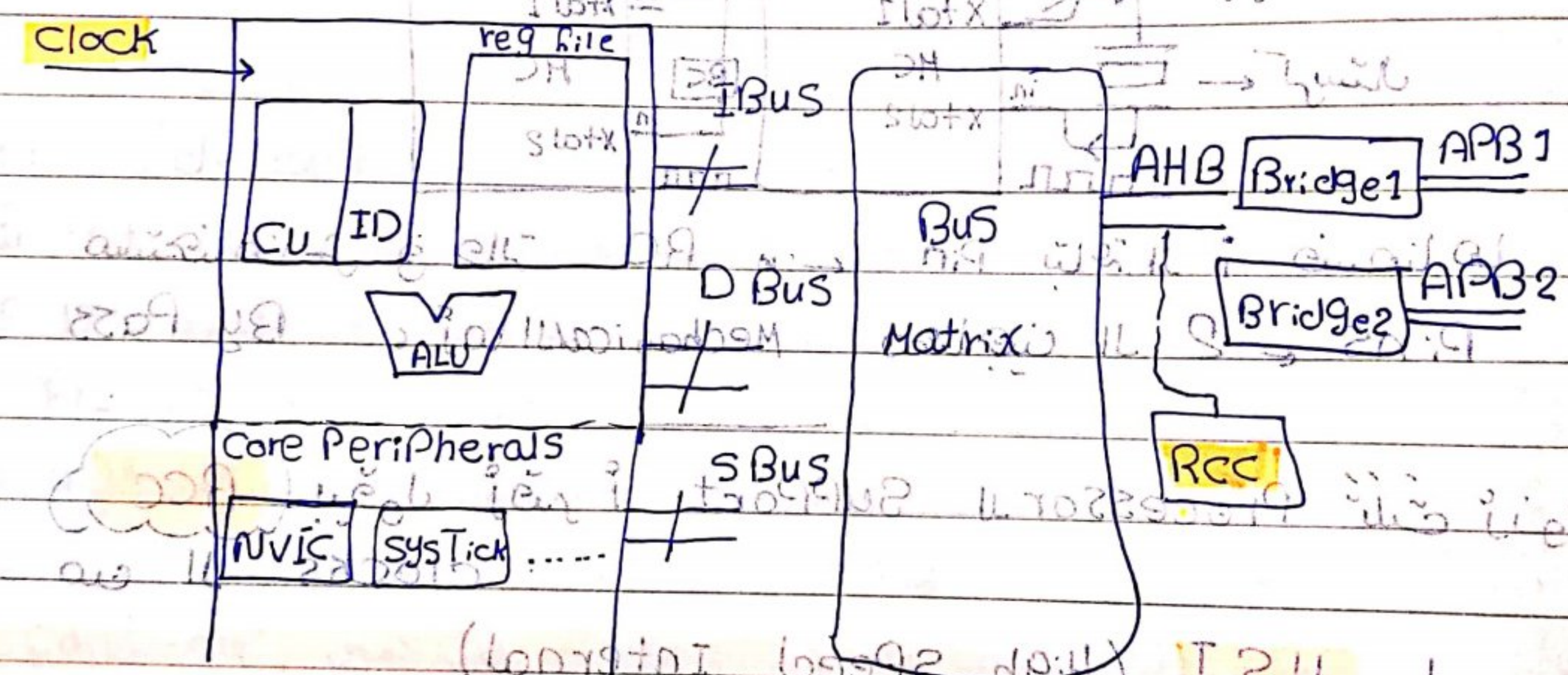
علشان كدا لازم أعرف الأول ايه ال noise هتتأثر به system ال

احنا عندنا في الـ Microcontroller بتاعنا الـ RC و Crystal
والـ PerPheral الـ اسمه RCC هو المسئول عن اختيار
إن Clock System الـ هيشتغل.

End 3

Start 4

ARM Cortex M3 Architecture



RCC هنستخدمه علشان نكنترول الـ Clock
داخلة على الـ Processor (Reset & Clock Control)

سؤال: Difference between Hard & Soft

1_ ↑Reset 2_ ↑Reset
3_ ↑Reset warm 4_ Power off & Power on

Reset & Clock Control

أي RCC هو الـ Peripherals

ST هي العملاق لأنه بره الـ Core Peripherals

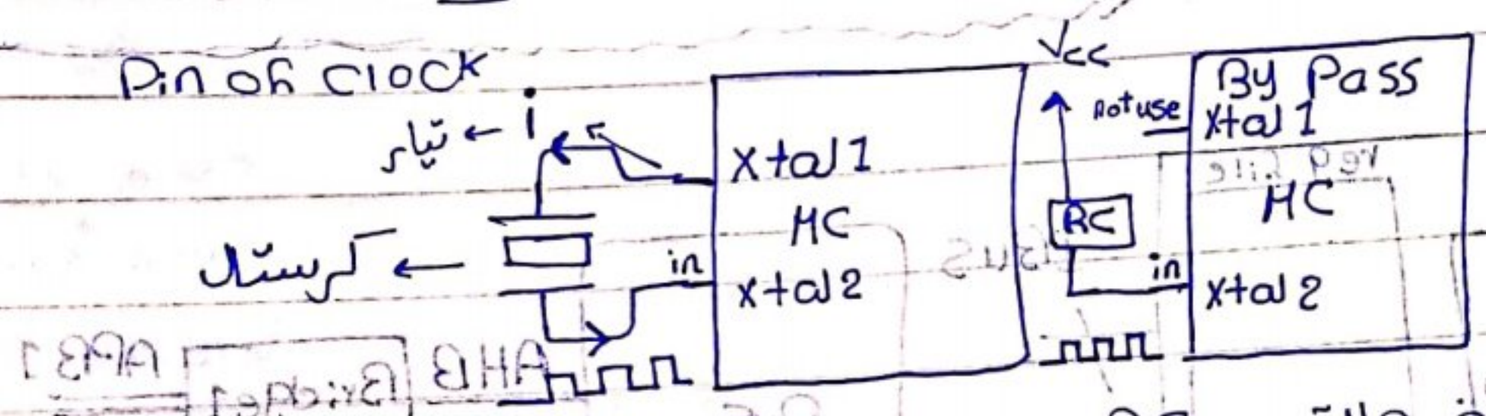
clk يعني بره ARM

* أول معلومة

لو Vendor Peripheral

1 Clock Connection

Pin of clock



مستخدمناش في حالة RC By Pass

Pin بتاعة الـ i خسرناها

محتاجين الـ 2 ← Pins

لأنها الـ Mechanical

RCC بيقول أقدر أ ..... Clocks من الـ

كل Processor لـ Support ثلث أنواع

- HSI (High Speed Internal)
  - RC 8 MHZ
  - هي جوا المايكروكنترولر يعني مش هتوصل، نوعها
- HSE (High Speed external)
  - Crystal → 4 to 16 MHZ
  - RC → up to 25 MHZ
  - هتوصل من بره وهي نوعين وتوصل في كل نوع زي ما بتوصل صح
  - يعني آخرك 25
- PLL (Phase locked loop)

clock → PLL → clock * X

up to 72 M

أومال ازاى الـ Processor ممكن يشتغل على 72M وآخره 25 ؟

* لأن أعلى clock هتوصل هى 25 ، لكن ممكن قبل ما أوصل الـ clock دى على البروسيسور أكبرها ، عن طريق دايرة اسمها PLL وهى دايرة بتاخد clock وبتطلع clock

* هنا بقى الـ RCC أول حاجة عملها لى هى اسمها

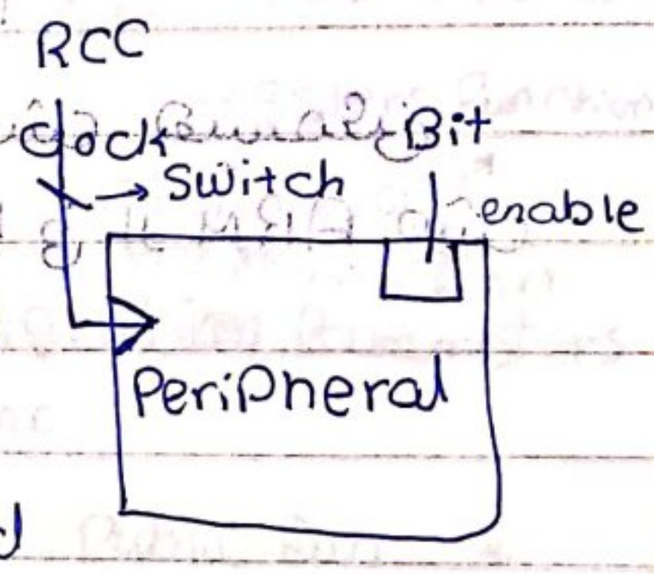
① Processor clock selection ميزة

نكمل Reset & Clock Control

① Processor clock selection

تانى ميزة

② Enable / disable Peripheral clock



أغلب الـ Peripheral بيكون ليها Bit بتكون للـ enable (0 or 1) ، أى Peripheral بيكون داخل عليه clock سواء هى الـ داخله على الـ Processor أو Pre Scaled هنا يعنى داخله على دايرة صغرتها شويه

طيب سؤال : هل لو هو disable هل هي consume باور ولا لأ ؟

- المنطقى لا .

لكن هو بيـ consume باور ازاى ؟

تعالى نشوف السؤال الأول

How The Peripheral is enabled ?

زى ما قولنا عنده Pin اسمها ① enable

وعنده دايره اسمها ② Enable check circuit

Periodically check The enable bit

بتبقى بتبص على الـ bit دى لو

① is written to enable bit

دا معناه إن السيركيت دى تروح تشغل الـ Peripheral circuit

if 0 is written

Disable Peripheral circuits

لكن السيركيت دى شغالة على طول علشان كدا الـ Peripheral
لو هو disable هيحصل إنه Consume Power صغير جداً
* طيب نعمل ايه علشان مبـ Consume ش Power ؟
هى الدايرة دى محتاجة Clock علشان تشتغل
*طيب ايه رأيك لو قطعنا عنها الـ Clock و هنحط الـ Clock
على Switch والـ هيتحكم فى الـ Switch دا هو الـ RCC
* كل Peripherals فى الـ ARM هيكون ليه enable
& disable للـ Clock .

ملاحظة : الـ DIO مفهوش Clock
أول خطوة ① بنفتح الـ Clock على الـ Peripheral مثلاً DIO
② بنروح نشغل الـ Peripheral دا
لو قافلين الـ Clock الـ Peripheral دا مش هيسمعنى
* ودى ميزة عالية فى الـ MC بتاعنا علشان كدا فى الـ ARM مش
بيحصل استهلاك باور كتير

بعت الـ RCC ليه وظيفتين :
① بيعمل enable & disable للـ Peripherals
② وإنى اختار الـ Clock الداخلى على الـ Processor

By default all Peripherals are disconnected
from the clock source
except RCC طبعاً
Core Peripherals

كدا أى Peripheral : هنعمل enable من حاجتين عن طريق
إن اعمل enable للـ Clock اللى عليه من الـ RCC ثم اعمل
enable للـ Per... نفسه .

معلومة :

open @ :

PLL & FLL → How it works?

Phase

* PLL داخل ال RCC

مصطلح جديد :

Naming convension

للتشركة لكتابة اسماء ال drivers

① Driver

→ XYZ_ interface.h → prototype of Public funct (fixed, owner / user)

→ XYZ_ Program.c → code

→ XYZ_ Private.h → Private information (registers functi, Private function)

→ XYZ_ Config.h → configration parameters

ممكن أغيره

* Public fun بتكون من حق ال drivers التانية يستخدموها بس

زي EnableClock & disable function

وفيه Private func مش عاوزين مبقدرش أي component يستخدمها

② Include

ترتيب ال include

↳ ① libraries أول واحدة

② interface file of lower layers

③ own Driver file → ① intehace file

② Private

③ config

زحياناً

ومش فارقة

في الداتا شيت 10

كل Register ليه اسم و offset وهو عنوان ال register

لكن بالنسبة لحاجة اسمها

كل Peripheral ليه

↳ Base address أول آدرس

وكل Register ليه

↳ offset

مقصود بيه بعد هذا ال register عن

ال Base address

وبالتالي ال Address

of Register = Peripheral Base address + offset

يعني مثلا

وكل register بيبعد عن اللي قبله بمقدار 4 addresses

لأن حجمه 4 bytes

يعني لو محتاج أعرف ال register ال اسمه CR دافين

لو هيبقى محتاج أعرف ال Based address بتاع ال RCC

Casting

((U32)10) = 50

اقصد address 10

dereference وأعمله U32 على إنه

وحط فيه 50

احنا عارفين إن RCC بيعمل حاجتين
Reset & Clock Control
من 10 ← registers دى فيه منهم 5 مسئولين عن ال
Clock دى هما أول اثنين ومن تحت أول 3 بعد أول اثنين
من تحت
وقولنا إن RCC كـ Clock Control بيستخدم فى حاجتين
إنه بكنترول ال input clock على ال Processor
وده دور الريجسترز دى CR و CFGR
هما ال بيختاروا ال Clock الداخلة على ال Processor زى
HSI و HSE و PLL ، وال 3 ← registers من 5
بيعملوا enable & disable لل Clock على ال
different Peripheral.

AHBENR
APB2ENR
APB1ENR

أى R بيكون 32bit [0:31]

Naming Convention
① Public function
SWC (Software Component)
SWC_ReturnDescriptiveVerb
② Argument
Copy_typeDescriptiveNoun
③ Macro
↳ All Capital
↳ SWC_DESCRIPTIVE_NOUN

→ on data sheet: (Rac)

$\boxed{CSS}$: clock Security system

0 ← disabled

1 ← enabled

هو SYS بيشتغل Security لو عملنا enable ، فلو كنت مشغل (HSE) والـ HSE خرفت مبقتش تطلع clock مظبوط

الـ CSS يعملها disable (يوقفها) ويديلك interrupt

ويقلب على الـ internal (HSI)

* يعني بتراقب الـ external clock.

* معلومة

علشان الـ RC دايما بيبقى فيها error عملنا 5 bits

اسمهم HSITRIM [4:0] ومكتوب فيهم by default

$\phi 0000_{16}$

كل زيادة تعملها بتزود 90 KHz وكل step بتنقصها

بتنقص 90 KHz

+ step → 90 KHz

step → 90 KHz

* RC ← 8M + error ممكن تكبر شوية أو أقل

هو قال هعملك حاجة اسمها TRIMing (تقليم) إنت هتقيس

الـ internal دي قد ايه وتشوف هي ناقصة قد

ايه عن القيمة الحقيقية إلا هي 8M وتبدأ تقلل وتزود في

قيمة الـ TRIMing هو كل زيادة إنت بتزود 90 KHz ومع النقص

كذلك

example

error

8MHz + 80KHz

هتنقص من الرقم دا ؟ في كل مرة 90 KHz

$\phi 0000_{16}$

بنجمع

هام

HSICAL[7:0] H.W calibration

يعني ال Hardware هو اللي بيعمل بيقيس ال error اللي عندك
ويحط قيمة هنا علشان يظبط ال 8M بس بردو ممكن يكون
فيه error علشان كدا سايب ال TRIMing إنت تعمله (user)
من أي حرارة

HSI

Measured = 8 MHz + 250 KHz

what is the value of Trimming?

10 steps 0 1 0 1 0

وساعتها هيفضل 10 KHz error

8 MHz + 10 KHz

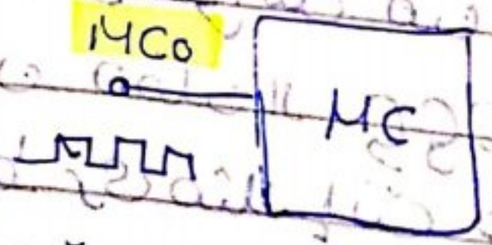
فدور ال TriMMing هو التحسين وآخره أنقص 16 step
أو أزود 16 step

CR

فال register دا دوره يعمل enable & disable
ال Clock بتاعتي اللي مقالش هيت فيهم
هيدخل على ال Processor

ال register (CRCCFGR) هو ال هيحدد أني Clock هتدخل على ال Processor

---



دي Pin؟ ممكن أطلع عليها أي Clock

لو طلعنا عليها 0xx يبقى مفيش Clock، ولو حطينا عليها 100
يبقى بطلع عليها System clock وهي ال Clock الطالعة على
ال Processor

lec 5

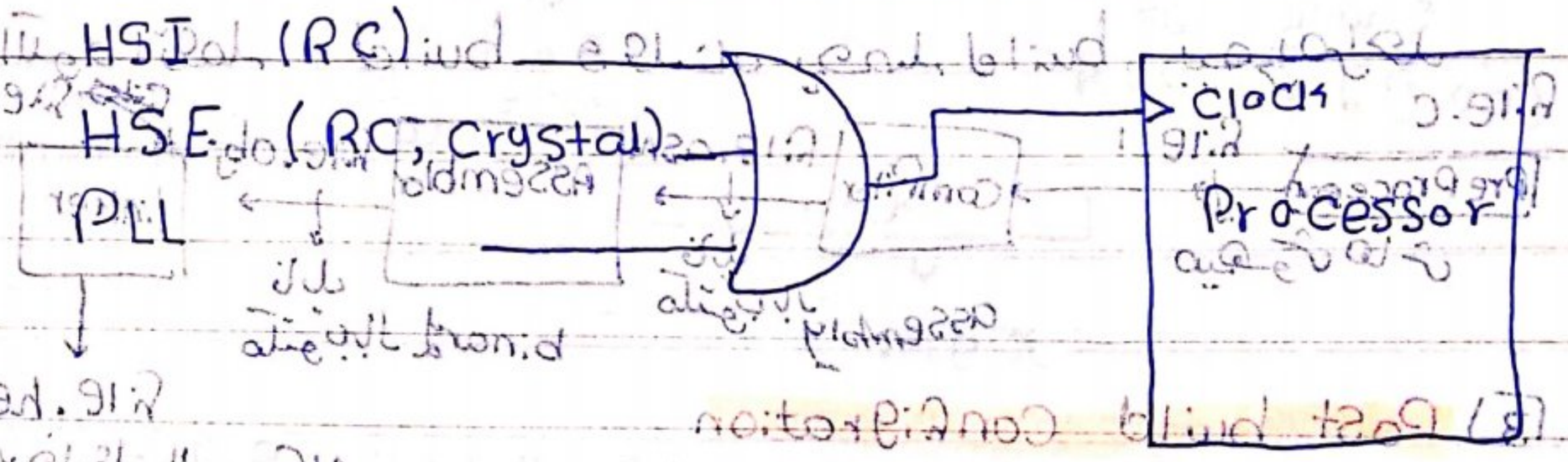
RCC

* Clock Control :

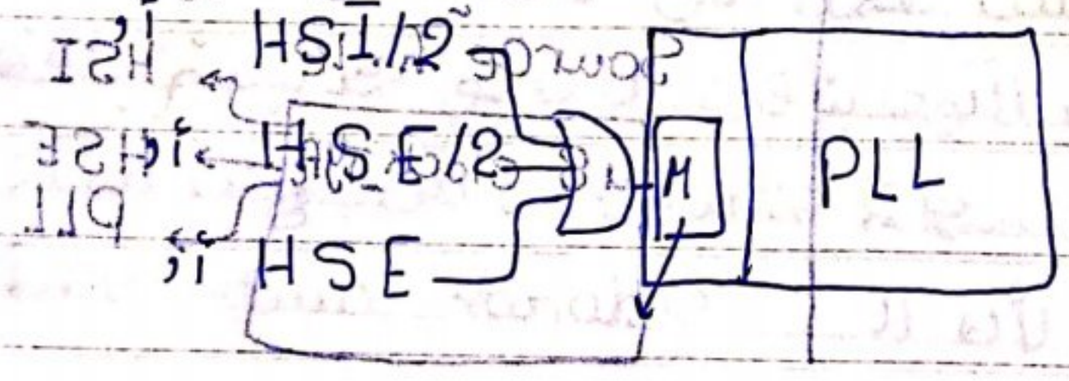
① enable & disable for ~~all~~ Peripheral Clock

② enable clock except
① RCC
② Core Peripherals (Processor

② Selection of System Clock (CR , CFGR)



طيب ال PLL ميطلع ال Clock بتاعته ازاي



Multiplier جوا ال PLL

PLL هو هيكون ال out .

Configuration Types :

دا اللي بعدل فيه وبناءً عليه الـ Code يتغير.

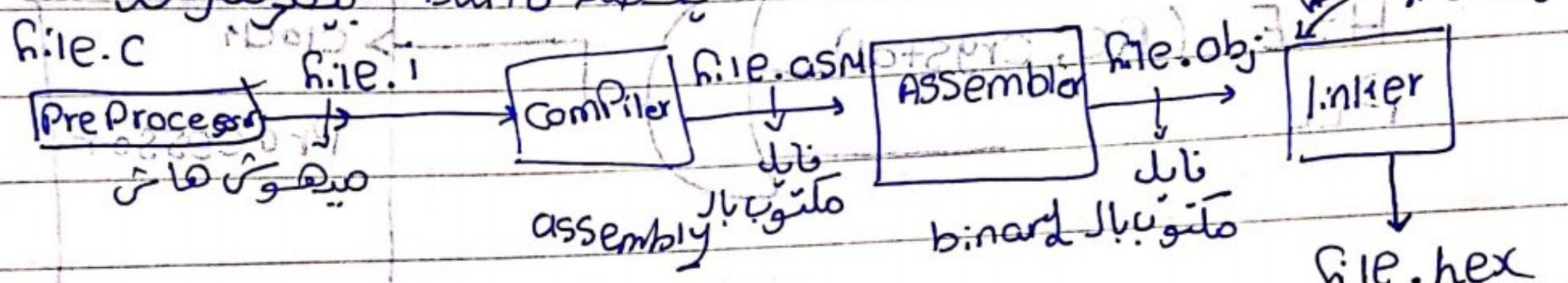
#define CLOCK_TYPE → HSI / HSE / PLL

1) Pre_build configration

اليوزر هي بتعمل Configration للكود قبل ما يـ build (Compile) ، ودا عن طريق ① PreProcessor directives هو اللي هيختار ، لكن عيبها إن بحتاج أعمل re build code

2) link Time configration

الكود اتعمله build وعلشان يحصله build بيمر بمراحل



3) Post build configration

وبعدين تعمل build فطلع تلك object file (file.obj) ثم link الفايل الأساسي مع file1.obj file2.obj الـ هو طالع من الـ variable مع بعض علشان تطلع file.hex ويدخل على الـ MC تكملة ، هو بيقول إن اللود نفسه عبارة عن object file وهديلك

Source file → HSI / HSE / PLL

u8 clock_type

link Time

RCC_Program.C

```
void Init ( )--
void Enable(--)
```

اعمل build للفايل دا
وطلع منه RCC_Program.o
الهو ال file.obj المرحلة قبل
الأخيرة المكتوبة بالـ binary

RCC_Config.h

```
u8 RCC_ClockType
```

وبعدين اعمل build
وهيطلع منه
RCC_Config.obj

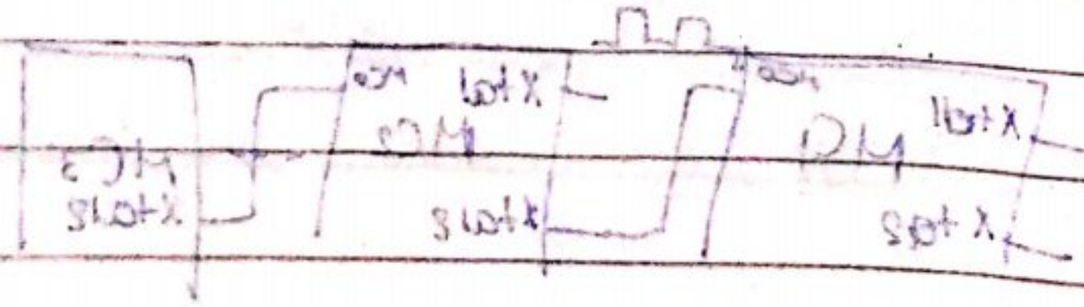
link

.hex

لو مختار HSI دعايز اغيره ، هروح افتح الفايل ال config
وأعدل وأغير فيه ال variable الدعايزه بس واعمل
built للفايل دا بس مش هعمل تاني للـ RCC_Program.C
وفي الحالة دي أنا حطيت الكود بتاعي لأن لو حد بيشتغل فهو هياخد
ال file.obj مش ال source file ، لكن في هنا مشكلة
هي ال Memory لأن ال variable مش زي ال #define
فهيقل ال Memory consumbsion على
أوضل واحد منهم

B) Post build configuration

قال هخليك تعرف تغير ال configuration أثناء ال runtime
مش محتاج كل مرة تحرق الكود من جديد



ازاى ؟ هناخد حتة من الميموري

Init → Read للميموري

والتغيير عن طريق أى Communication Protocal

من حيث تغيير القيم الى فى الميموري وأنادى على ال function تانى من غير ما أعرف الكود تاني.

Memory
Clock Type
Configuration

open Q

طيب عيبه ؟

Software

Why IMCO?

① for Diagnostics

لو عندي شب clock معينة وعايز أطبعها أو clock أدي بعلمه للوحدات مثلا مشغلة RC وعايز أطبعها شغالة على 8M مظبوط

أطلعها على Pin بتاعه MCO وأرسمها على الـ oscescope

وأعرف قيمتها قدايه والـ error قدايه واعمل Trimming قدايه

وكذلك للـ PLL

② clock cascading

Pros → Resource Reduction
Synchronization

Cons → Dependency (عيب)

شايفين نفس الزمن هوايه الأول

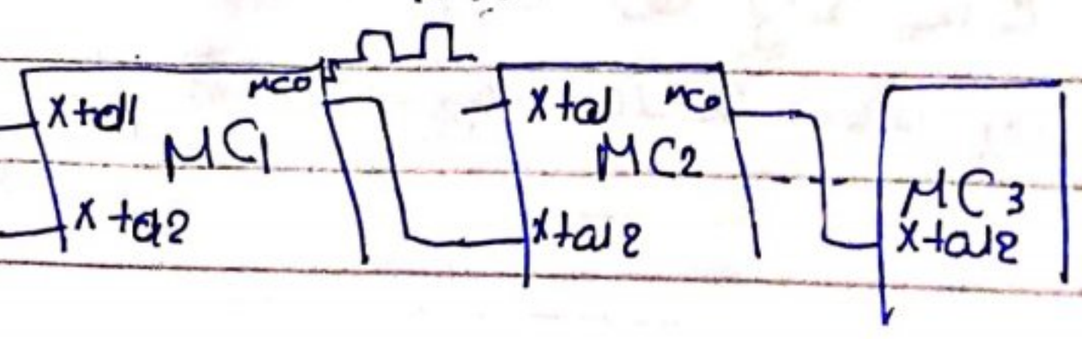
وعايز اشغل الاتنين على 16M

بدل ما أجيب 2 Crystal

أجيب واحدة للأول وأخليه يطلع على ال MCO ال 16M وأخدمه

على ال as input على Xtal2 لل MC الثاني

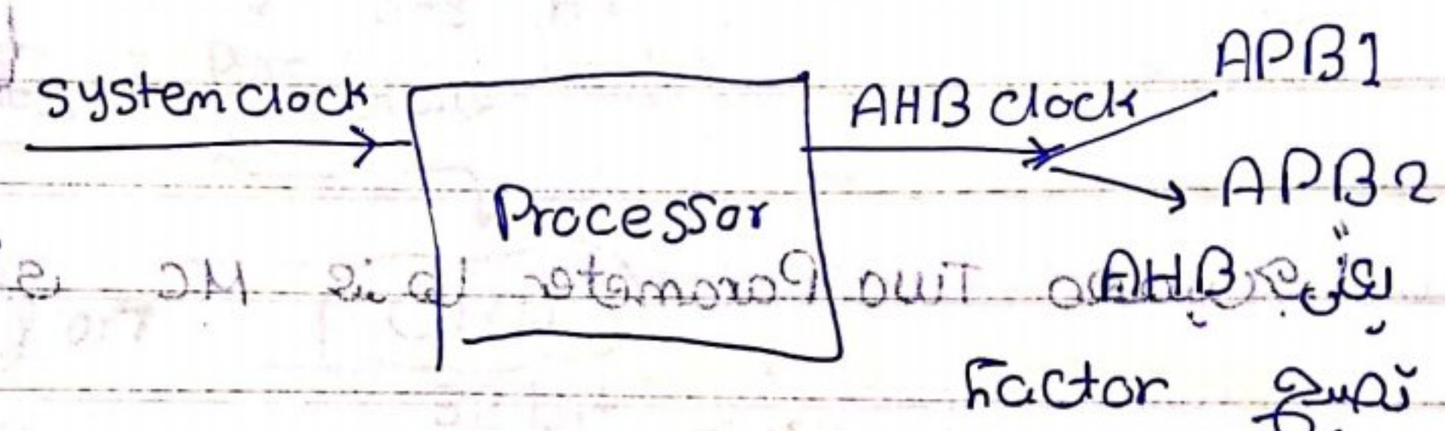
وده اسمه clock cascading



الميزة فيها إني وفرت فلوس وعملت Syncronization
لكن فيه عيب خطير هو
→ Dependency
انه معتمدين على ال Clock بتاع الأول فلو باظ هيبقى الي
Sys كله باظ وهكذا التاني

CFRG

فيه USB PRE
يعني ايه Pre-Scaler يعني الحاجات دي هتاخد Clock بكام



يعني AHB
تصبح factor

ساعات ال SYS Clock عايزين من AHB يطلع APB1 APB2
يعني كدا هما الاثنين factor من: AHB
يعني مثلا لو حطينا أرقام زي الفي البانا تثبت على HPRE
فا معناه إن أي Peripheral متوصل بال AHB هيدخل
عليه ال Clock دي وكمان هتكون APB1 و APB2

GPIO

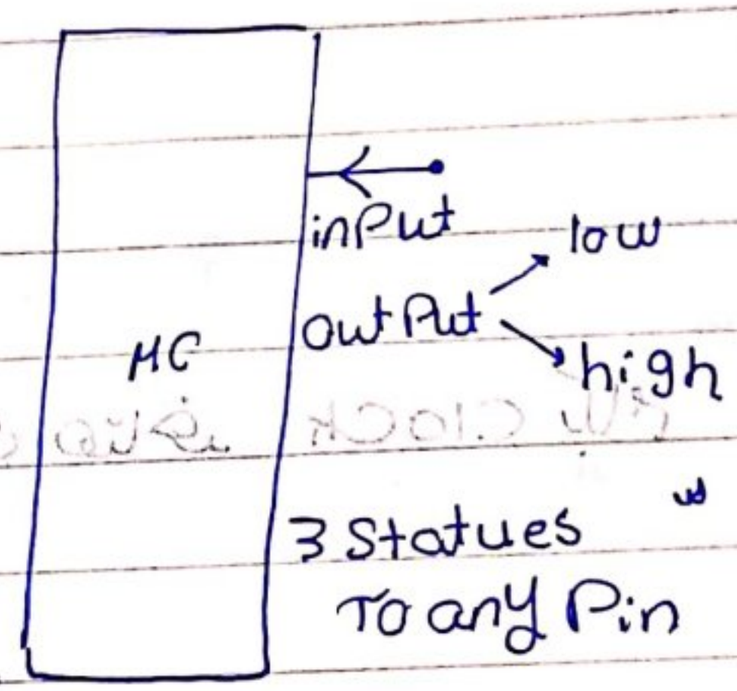
General Porpose InPut ouPut

3 Ports — 35 IO Pins — غير xtal2 ، xtal1 بتوع ال crystal

↳ A 0 → 15 IO Pins

↳ B 0 → 15

↳ C → C13, C14, C15 only Three



* أي Pin في أي MC عندها Two Parameter مهمين جداً

اسمهم

Source Current : يعني ال current اللي بتطلعه لو output

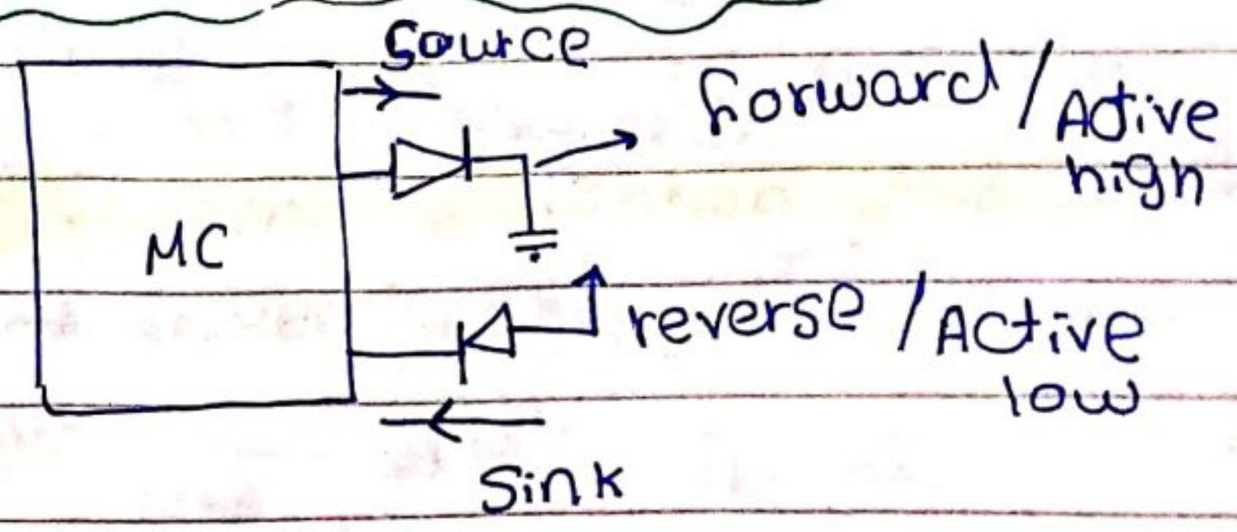
Sink Current : لو بتاخد current

* في الغالب ال Source بيكون أقل من ال Sink

الماتلوكنترولر :

Source Current = 25 mA

Sink " = 150 mA



ioPins بت Source قيمه أقل من قيمه ال Sink

ال Reversed أفضل

والـ 25 mA ، 150 mA دول خاصين بـ Port A , B

* تلت C

Source Current = 0 mA

Port C بتعمل Sink بس

AVR

DDR direction

PORT output regist

PIN input registe

u STM32F103

Data direction Registers

* each Pin have 4 bits for mode

يعني 16 Mode

Port 16 Pin ⟹ 16 * 4 = 64 bit

bits

علشان كل Pins عندي 2 Regs

CRL 0:7 — عبارة عن 32 bit

CRH 8:16 — 32 bit

CRL (Configuration Register Low) low علشان بيكنترول من (0:7) bin وكل واحدة عايزة 4

4 * 8 = 32

كذلك CRH ، (8 : 15) High

وكل Port ليهم 2 registers واحد CRL و CRH

* ⟸ ⟹

4 bit configuration for 1 pin

2bit	2bit
CNF	Mode

Configuration

output

00 Push Pull → low / High 1 (3.3)

01 open drain → ***

10 Alternative function (AF) (Push Pull or open drain)

11 يعني مش تبع DIO هو Pre آخر

input

00 Analog

01 Floating

10 Pullup or down ← يعني حتى لو متوصلش بحاجة فهي متوصلة من جوا

11 Reserved (Not Allowed)

00 → input

01 → output @ 10 MHz

10 → output @ 2 MHz

11 → output @ 50 MHz

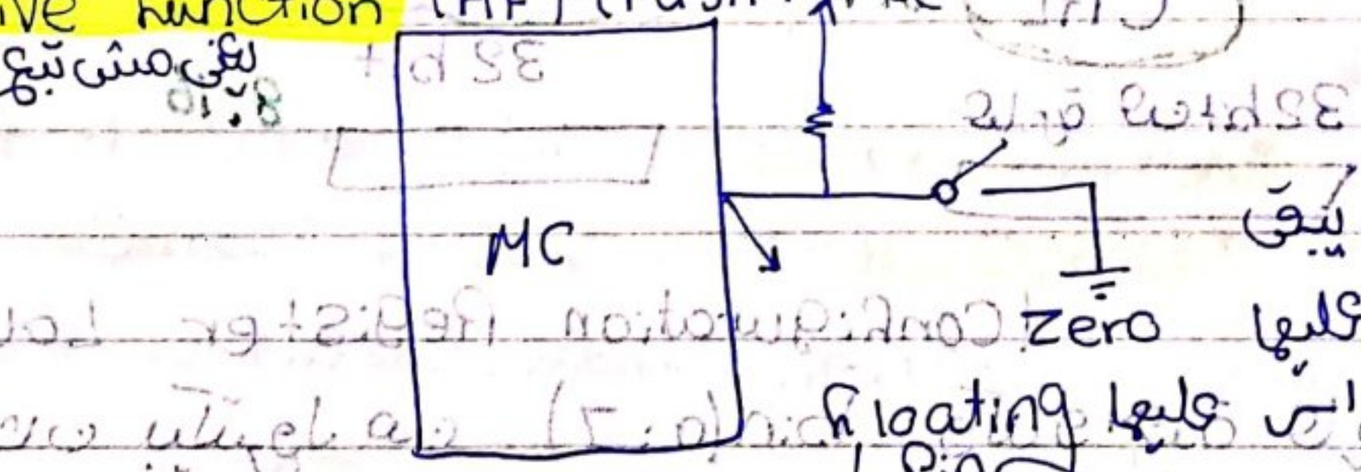
اهتماماً بال Power ، فلو هحول output لازم أحدد ال freq

* فال CNF لو input ليهم معنى ولو out فليهم معنى تاني

لو input و عليها 00 هتبقى input analog

* المفاجأة إن أي pin في ال MC بتاعنا ينفع تشتغل ADC

ولو 01 هتبقى input floating

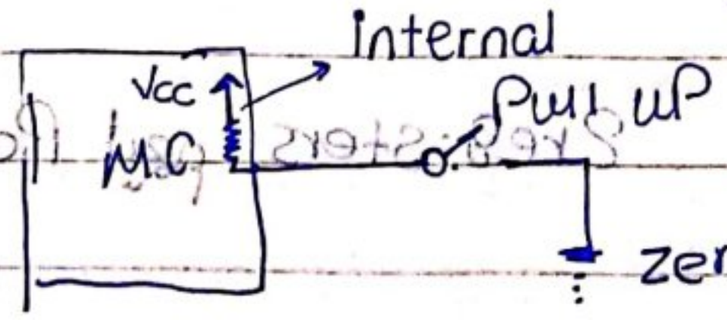


لو مش يبقى ال pin عليها zero

ولو مش عليها floating

وعلشان ميقاش عليها noise بنحط مقاومة

Sensitive to high & low



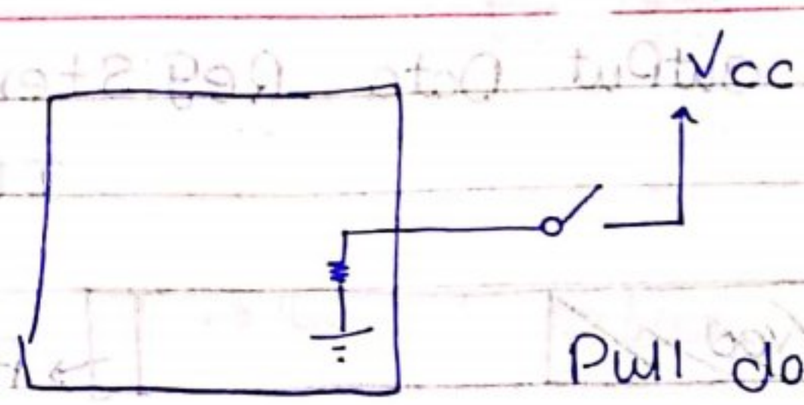
لو مش هتبقى zero

لو لا يبقى one

⇒ Pulled up pin ⇐

sensitive to low only

لأنها لو وصلنا عليها Vcc هي هي عليها Vcc ، لكن لو وصلنا low بيها هتتأثر وهتبقى low



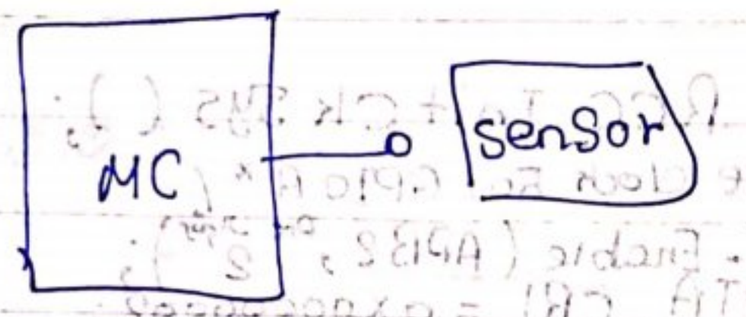
Pull down Pin
(sensitive to high)

ال low مش بيأثر فيها.

طيب امتى استخدم ال floating Pin أو pulldown أو pull up

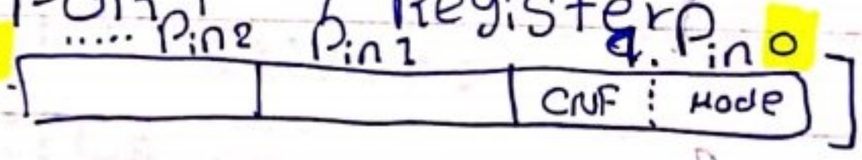
فلو pull up أو down مش فارقة مع Switch

، أما لو مع Sensors هستخدمها floating



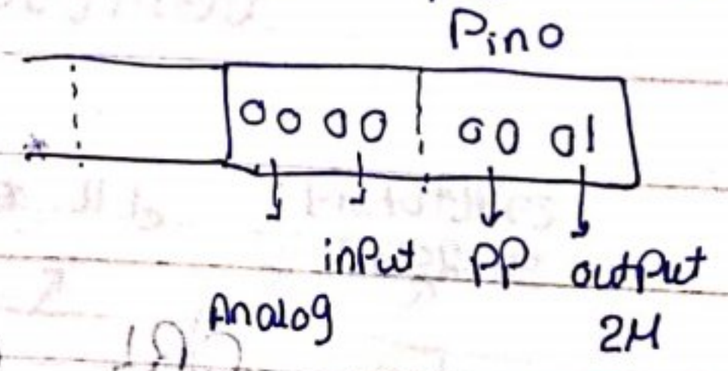
Each PORT 7 Registers

CRL
CRH
IDR
ODR

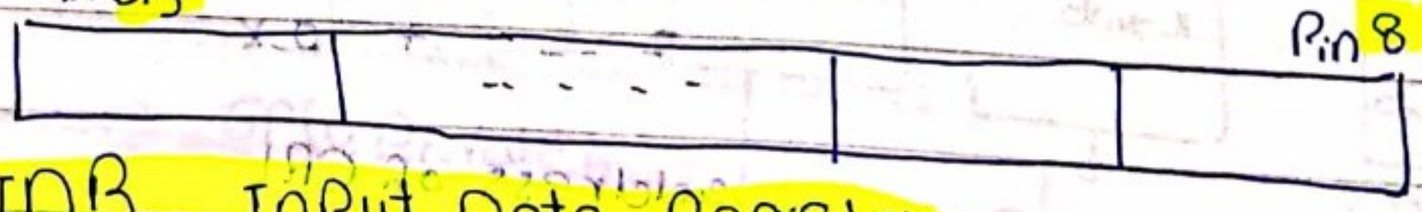


CRL ①

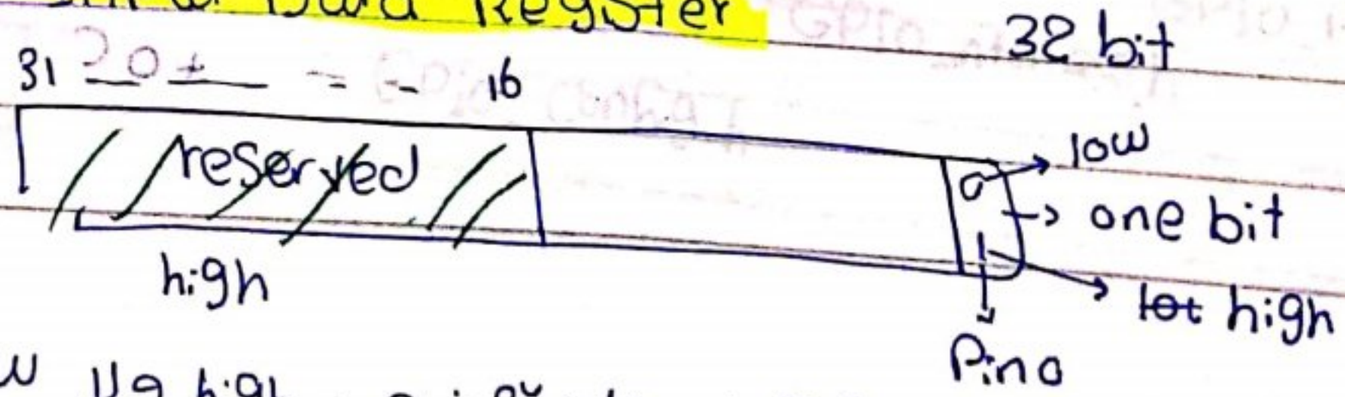
كل R فيهم عبارة عن 32



② CRH نفسه



③ IDR Input Data Register

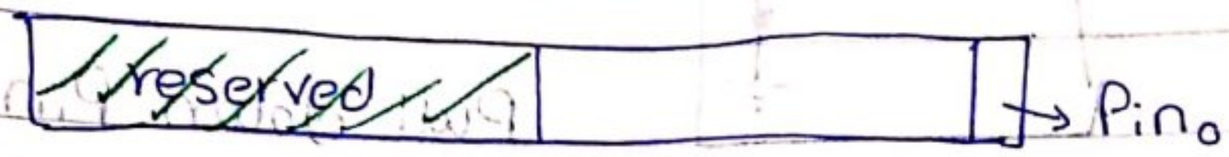


لو عندك Pit هي input و عايز تعرف هي high ولا low خش اقرأ ال bit

المقابلة ليها ، زي PINR في ال AVR

[4] ODR — output Data Register

عكس IDR



لو عايزها output وعايز أخرج عليها zero أكتب عليها zero

ودا زى ال PortR فى ال AVR

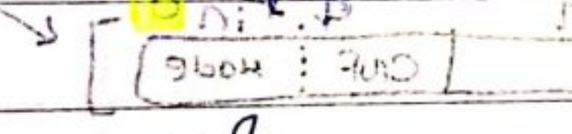
عايزين ننور ال led

```
main()
{
    [1] RCC_InitClkSys();
    /* Enable clock for GPIO A */
    RCC_Enable(APB2, Pre 2);
    PORTA_CRL = 0x00000002;
    PORTA_ODR = 0x00000001;
    while()
    {
    }
}
```

وبعدين نكمل enable للـ clock على ال Periph بتاعه

GPIO → APB2

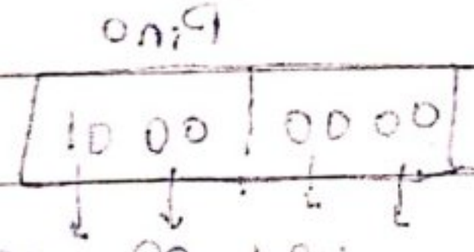
① CRL



output PM PushPull

01 00 CRH

IDR



* GPIO Based address

0x90010800

CRL offset 0x00 , ODR offset 0C

② لسيف CRH

∴ address of CRL

= — — + 0x

address of CRL

Eclipse

① SRC : بنحط فيه أي file.c

② System : فيه library جاهزة هو عاملها

③ include :

بنحط فيه أي file.h

④ ldscripts : هنشرحه في الآخر

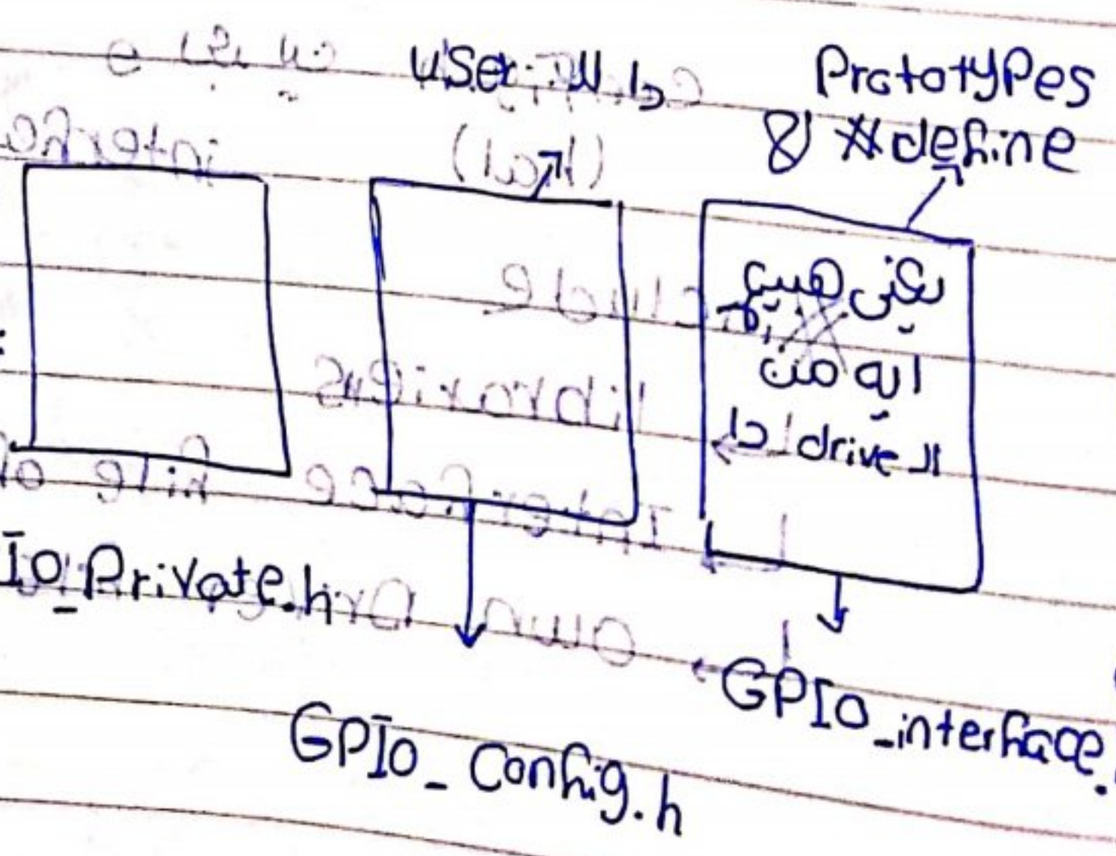
lec 6

⑤ GPIO Driver :

- LEDs
- Seven Segment
- Switch

⑥ ليه :

فيه أي حاجة هيستخدمها user الـ Programer مش الـ register زي أسماء وعناوين الـ



أولاً :

1 أول فايل بنفتحه هو ال Register file.h

* ال DIO هو مش Core PeriPheral

وهو متوصل على APB2 يعني تبع ال Vendor

2 نفتح ال data sheet

3 ونفتح فايل ال Private.h ونكتب ال registers وال Ports + عارفين نعرف ال registers الخاصة بال Ports

A, B, C

*

ال Registers

* 4 وبعدين نفتح فايل interface.h وده معناه انت هتبيع ايه من الدرايفر ده.

5 نروح على Program.c ونعمل

1 #include to LIB

2 #include to interface from low layer (MCAL)

3 to my interface (hal)

وبعدين config (hal)

6 أروح لل interface

#include

↳ libraries

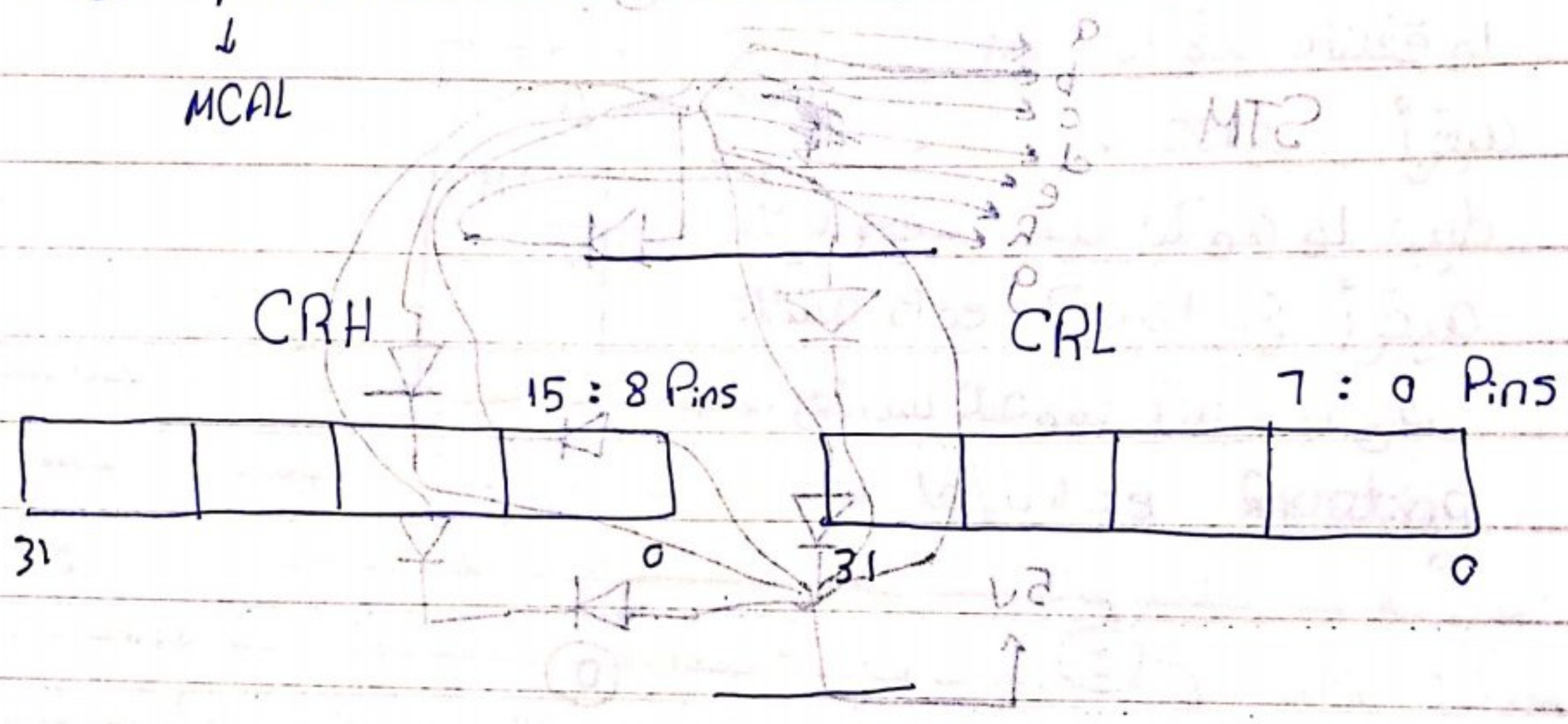
↳ Interface file of lower layers

↳ own Driver file
- ① interface
- ② Private
- ③ Config

لو جينا في ال register ... CRL اسمه
هنلاقي هو ال 31:0
وفيه 4 bits للـ Pin
فمن أول 0 : 3 هي 4bits
للـ Pin 0 وهكذا

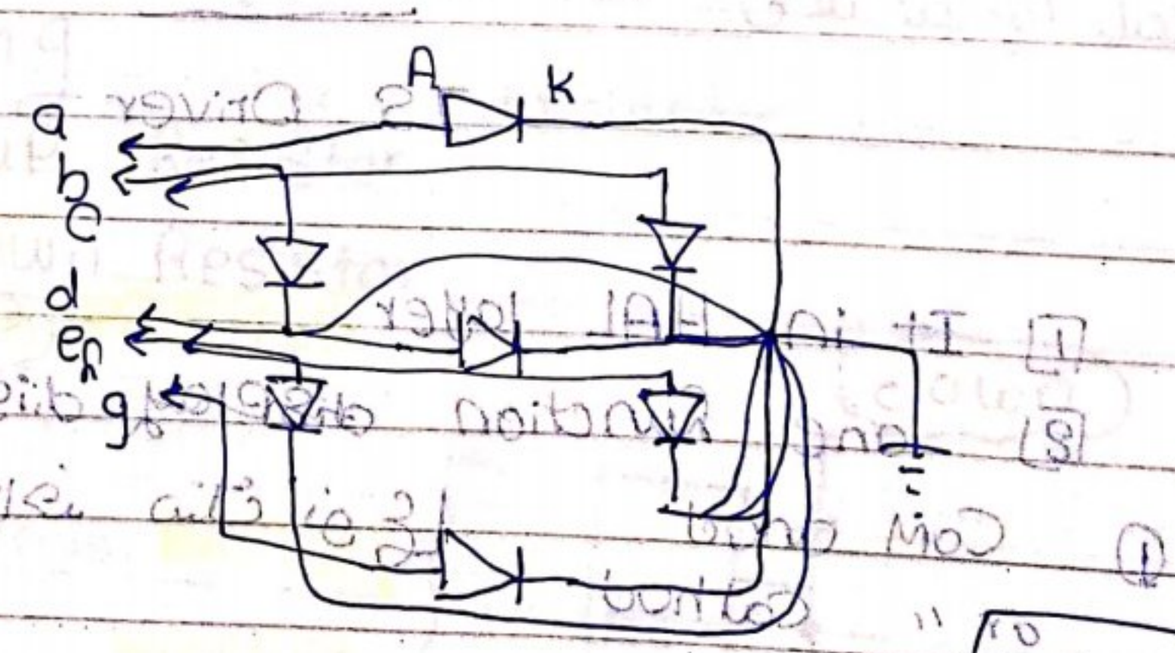
① why Reset before set of register?

② μGPIO
↓
MCAL

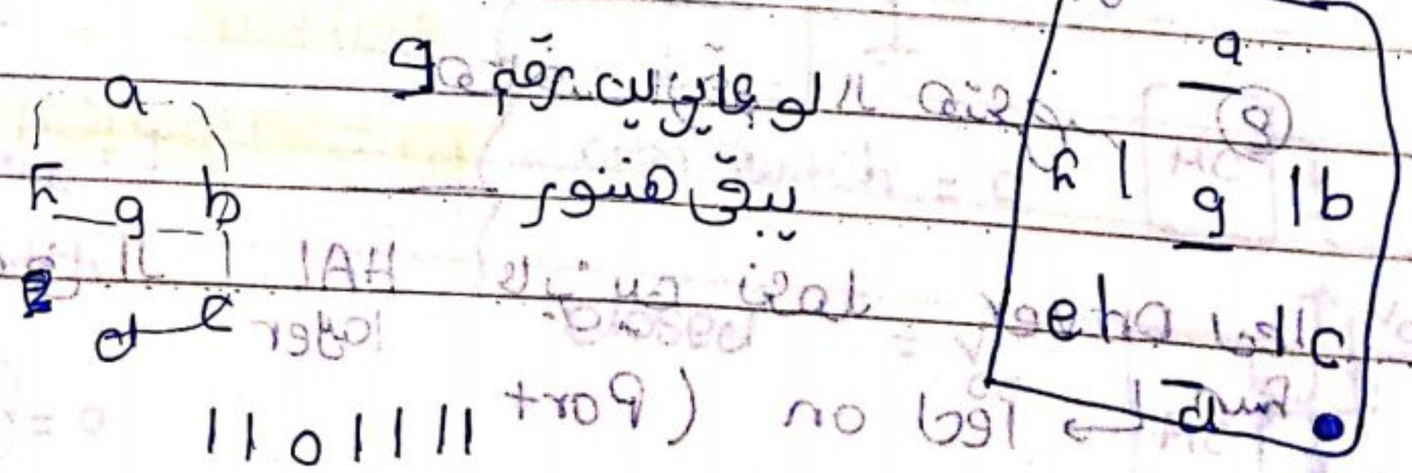


Seven segment

عبارة عن 7 leds وهي وسيلة عرض للأرقام بدل ما كنت بعرض حروف



لو عايز ليه رقم 9
يبقى هنور



1101111

لو عايز تستخدم floating number

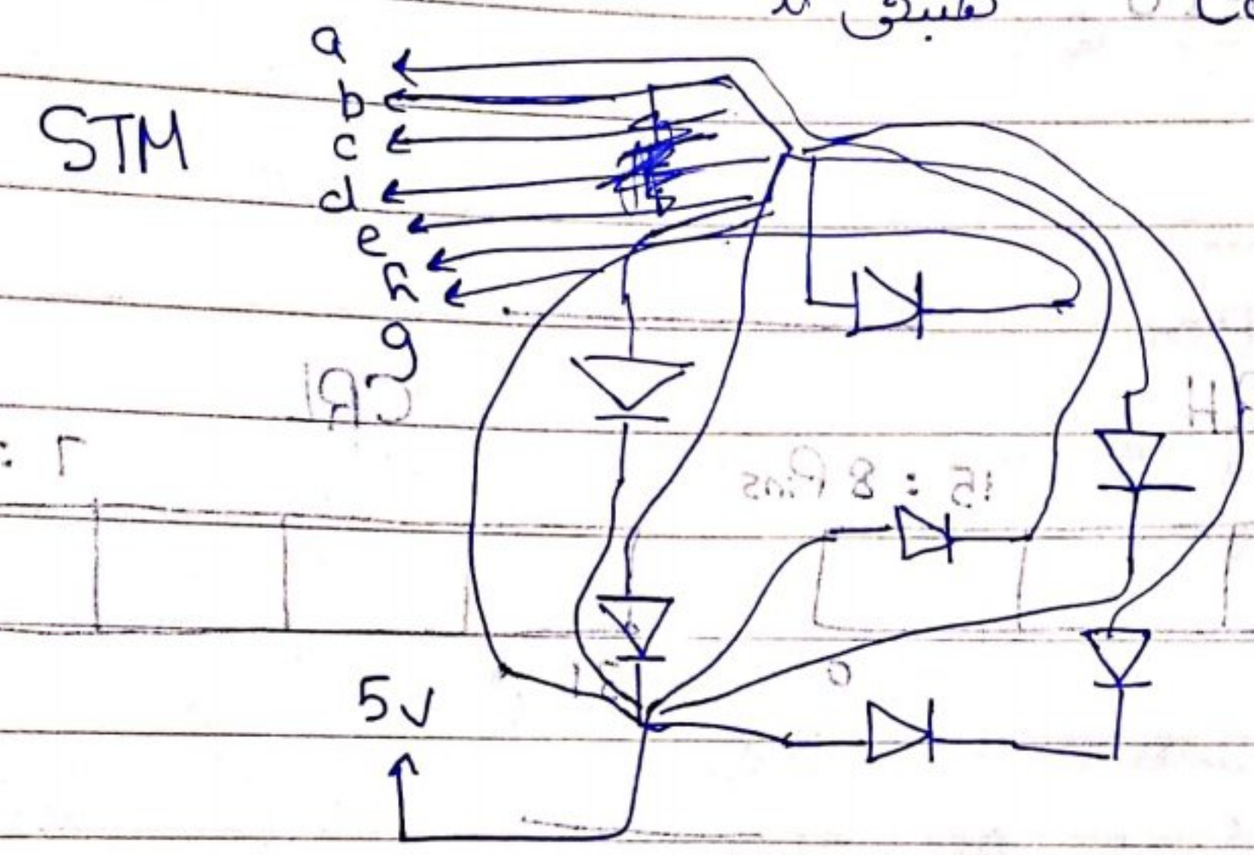
هنوصل ازاى

حاجه اسمها 7S هنختبر ال Com anod الأول هل هى Com Cathod ولا

لو الشكل ال احنا راسمينه ورا فدا اسمه Com Cathod

يعنى ال led يتنور بالواحد ويطفى بال zero

ال Com anod هتبقى كدا



ال Com هو ال anod

فى هنا ال بلنور بيبقى ال zero والمطفى هيبقى ال one قدام

7S Driver

[1] It in HAL layer

[2] one function display

① Com anod هتاخد منك نوعها

or " cathod

② هتاخد بالرقم ال هتعرضه

وكمان فى ال HAL layer عايزين نعمل Driver لل

led on (Port

Task

① PLL ← هندسيهالا Port بردو
HAL layer ② Seven Segment
HAL layer ③ LED Driver

* Switch :-
Mechanical Switch

هو عبارة عن سلك

لو هو مفتوح يبقى open circuit

لو هو closed هيبقى اكيد في short circuit

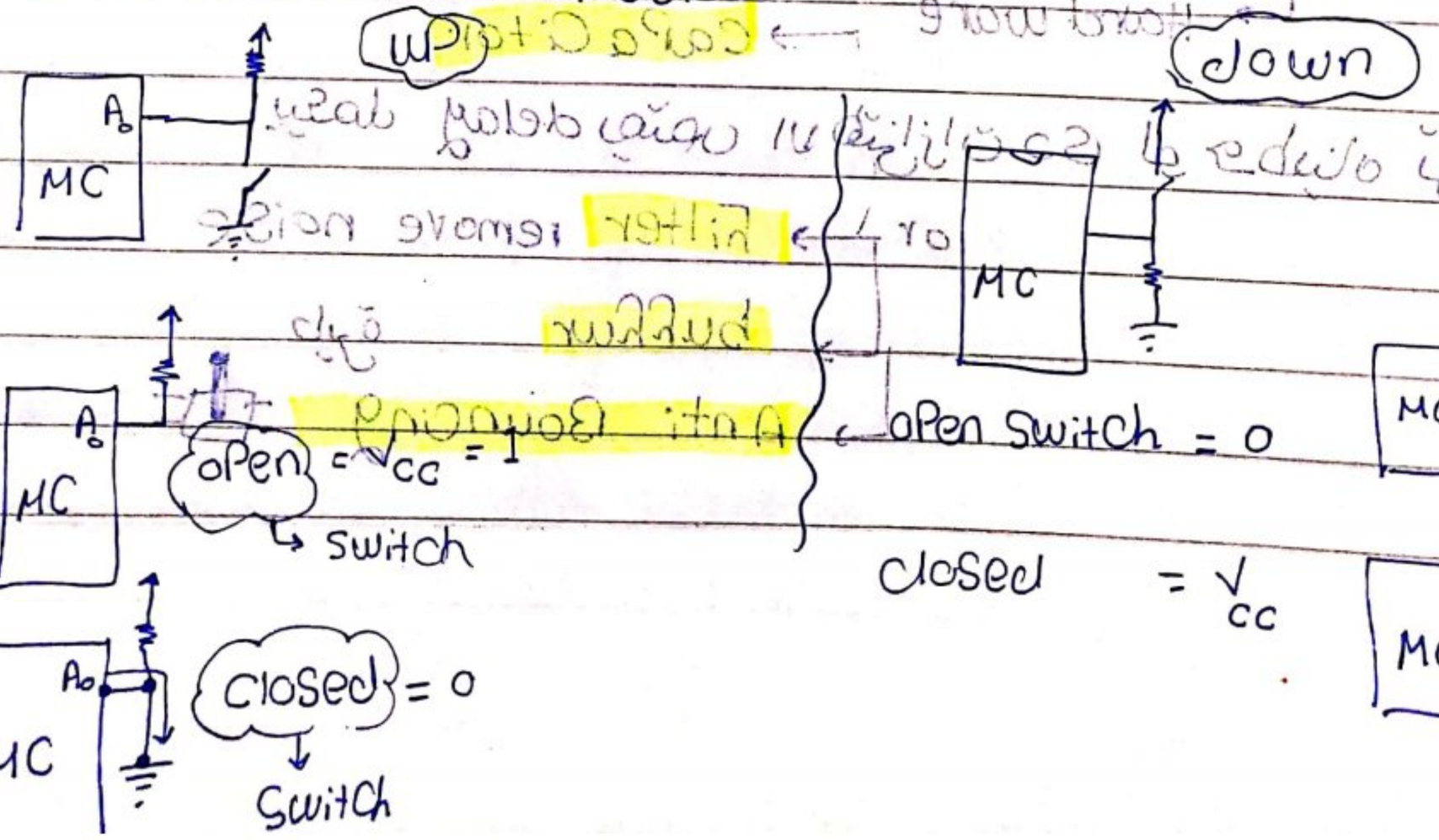
لو قفلناه يبقى ال MC
هيقرأ zero .
طيب لو هو كده مش مقفول
هيقرأ floating وده مشكلة
يعني ال Pin هتوصله سيبها floating
وده ياما

0 ← 1 → noise

بأنواعها

طيب نعمل ايه لما نلاقي Pin floating عندنا في حالتين

* floating
↳ Pull_up Resistor
↳ Pull_down Resistor

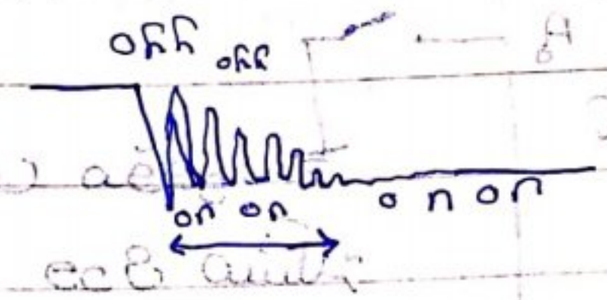


هنا بقى في الـ ST وفروا حاجه داخلية ، فهنوصل على طول على الـ Switch.

* لكن هنا في مشكلة تانية :

* Bouncing :

دى ظاهرة طبيعية موجودة في الـ Mechanical Switch وهى إنه ماده صالبة فلما تيجى تطبق السلكين فوق بعض مش هتعرف فهتزل و يفضل يهتز



فلو مثلا الـ µC بتاع الـ MC بيساوى 8MHz يعنى يقدر ينفذ 8 مليون أمر في الثانية الواحدة يعنى يقدر يحس زبذبة لكن ممكن الاهتزاز دا يوصل لـ 21 مثلاً فهيحس بيه الـ MC زى ما بيحس بالـ 1

* يعنى أنا و أنا بدوس على Switch 1 مرة واحدة بدوس 80 دوسة

حيث يتحلهلها زى : عادتها حاجات .

- Software → (delay) → 10 ms حد أقصى
  - if (switch closed)
  - delay (10 msec)
- Hardware → Capacitor
  - يعمل delay يقص الاهتزازات دى لو حطيناه توازى مع الـ switch
  - or → filter remove noise
  - → buffer دايرة
  - → Anti Bouncing

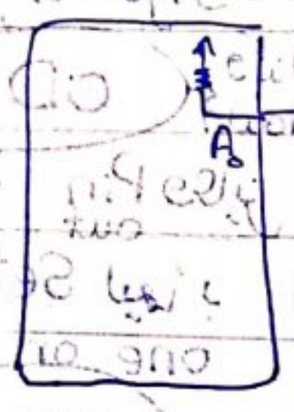
طيب هنا هنستخدم ال SW ولا HW

* على حسب ال APP :

لو APP معتمد على Switch يبقى ال HW ال APP زى ال Alarm

ولو مش فارق التكلفة يبقى SW أفضل لأن مش هتفرق لك السعر قوي.

* هنوصل ال Switch (Pull up)



0	GND
1	Floating

طيب لو Software هعمل (Pull up)

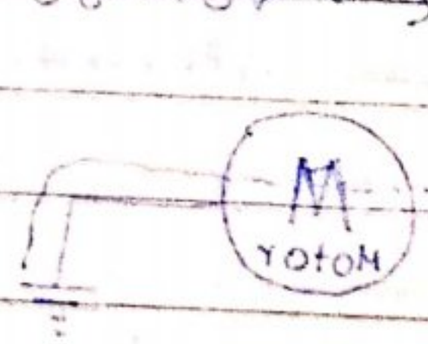
config → mode

if Pull down → ODR = 0

Pull up → ODR = 1

Software code.

End.



0	GND
1	Floating pin

هعمله بيعنى disconnect لما أي

1 open drain output:

Conf	mode
0100	01

	value	action
Push Pull	ODR في ال 0	GND ← Pull
	ODR 1	V_{cc} 3.3V ← Push
open drain	0	GND
	1	floating

CRL
CRH

كانوا بيعملوا Config للـ mode بتاع ال Pin بكلمة السر وهكذا.

ODR ← وكذلك في حالة * input Pull up down

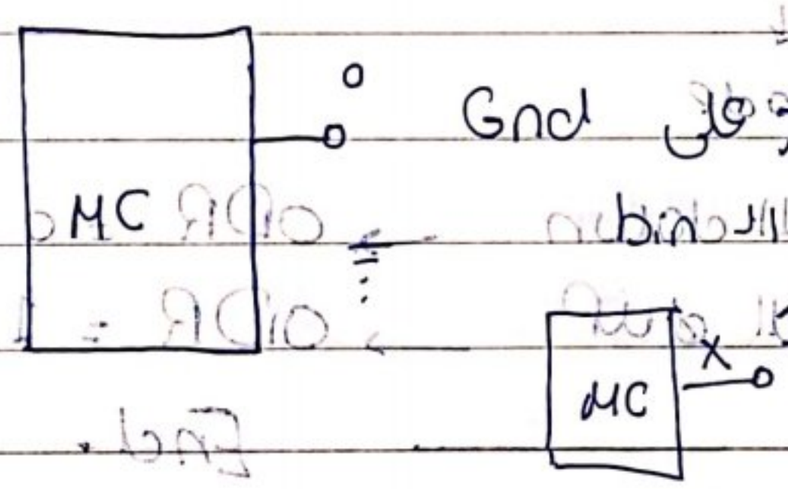
* ولو عندي Pin out وعايز أعمل Set ليها بـ one or zero

IDR

بتاع input وعشان أقرأ Pin ← input

يعني كأن ال Pin ده disconnect من على ال MC

open drain Pin



لو عليها zero فكأنها واصلة على Gnd

ولو عليها one كأنك قطعت الـ bin

ده كمان على ال MC

ده ميزة كبيرة جداً:

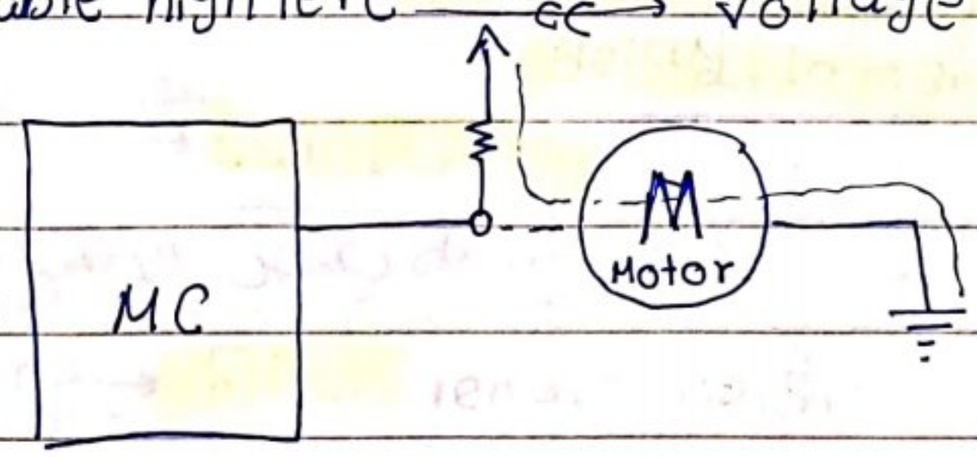
وهنا ال open drain ليه استخدامين مهمين جداً:

[1] ميزة Configrable high leve V_{cc} → voltage

0 → 0V ⇒ off

1 → floating Pin

فكأنها disconnect فبنجيب مقاومة على V_{cc}



فالميزة هنا إن كدا V_{cc} مش Micro controller V_{cc}

يعني حط أي volt إنت عايزه مش لازم ال volt بتاع ال MC

إنت الـ هتقرر لما تكتب 1 أن الـ Volt هيدخل على Motor ودا اسمه:

Configurable high level voltage

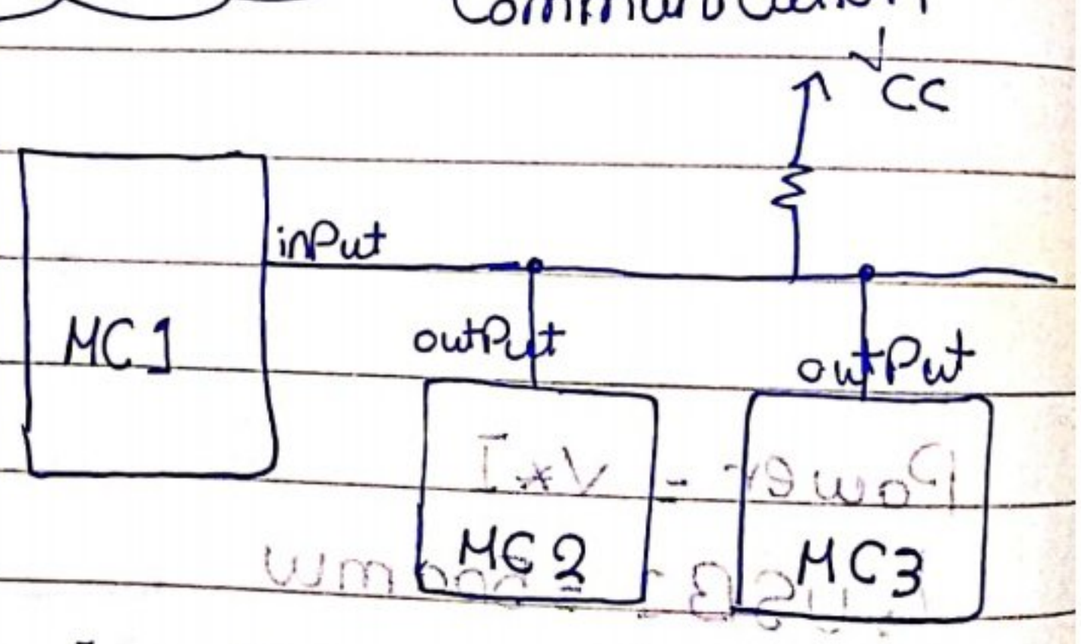
وده عن طريق أوصل مقاومة على الـ Volt العايزة وأكتب one على الـ Pin.

* أحنا ممكن نعمل كده في المايكروكنترولر العادية عن طريق إننا نوصل Transistor بره وهو يعمل القطع ده.
* وكان لازم نحط مقاومة قبل V_{cc} علشان في حالة الـ zero ميحصلش Short Circuit ويدخل على V_{cc}.
* كل الـ Pins بتشتغل على الـ Mode دا.

note [2] تاني مميزة

open drain Bus

بيستخدم في حالة Communication



واصليت على نفس الـ Pin وبقدروا يبعتوا داتا لـ MC1 في

* أول حالة في مود PushPull:
  1. MC2 , MC3 = 0 ، فهيبقى MC1 = 0
  2. " " = 1 ، " = 1
  3. لو حدفيهم zero والتاني 1 ، هيحصل Short Circuit ، هتولع
* هنا بقى لو الـ 2 Pins دول في open drain:
  1. MC1 → 0 = 0 0
  2. 1 0 1

يعني الاتنين بيفصلوا نفسهم عن الـ MC فال Pin دي floating

فبعد فتوصل ال pin دى على ال V_{cc}.

* هنا بقى لو جد وصل zero والتانى one

ال zero هيوصل نفسه بال Gnd وال one هيفصل نفسه عن ال line فهيقرأ line ← zero

وال حصل منك حصول ال Short Circuit

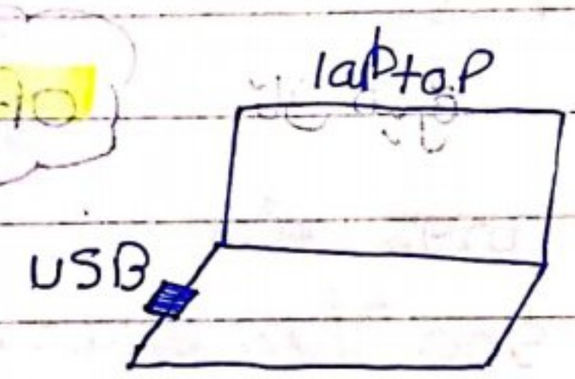
وظهر عندى Consept باسمه

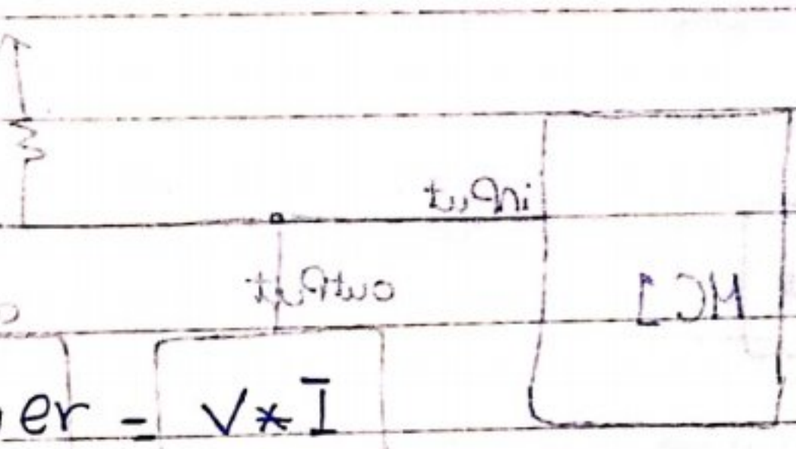
① Dominant bit → مسيطر ولو واحد

② Recessive bit → لو كله كتب one

تانى ملاحظة:

note [2] LED Power :





Power = V*I

↳ USB1 = 500 mW

↳ USB2 = 500 mW

↳ USB3 = 750 mW

		default	Max
USB 1	5V	100 mA	300 mA
USB 2	5V	100 mA	500 mA
USB 3	5V	150 mA	900 mA

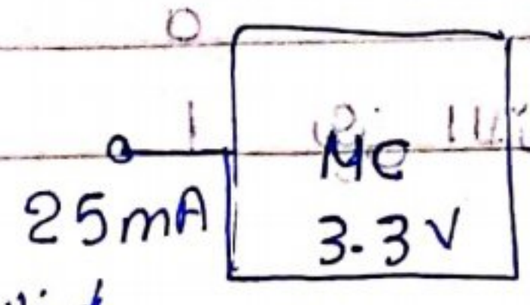
① COM, COM = 0 , COM = 0

② " LED = Max Power = "

③ Range 40 mW : 160 mW

آخرها 160mW يعنى كلهم متحرق عندى USB

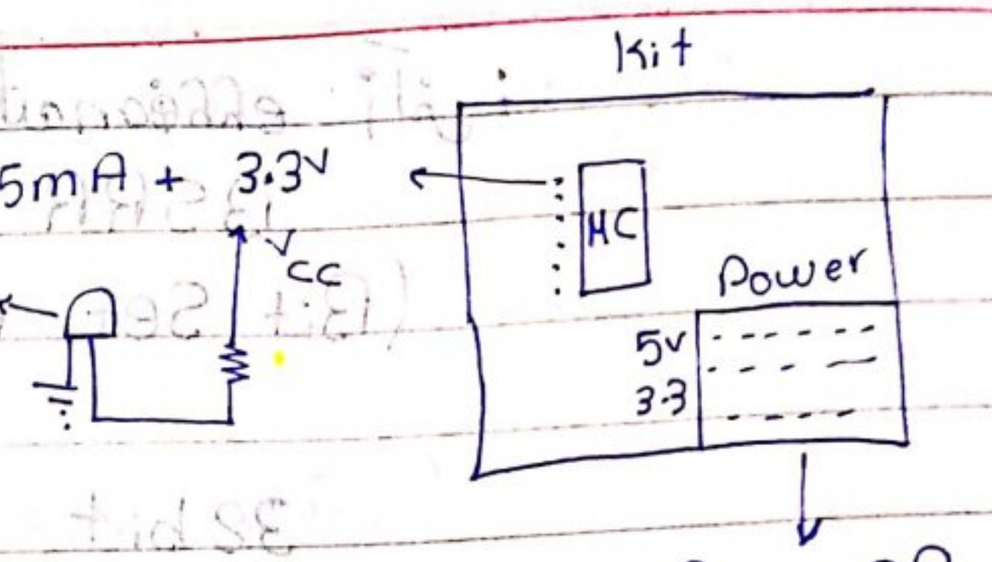
* 25 mA ← 0 = 0



Max level غالباً دا ال

outPower = 3.3 * 25 < 160 mA

outPower = 5 * 25 = 125 mA



مش هتحرق الـ led فلو كدا الـ 25mA + 3.3v

فاحنا نهتم ونتعامل على أساس الـ Power not Volt

مع led أو 7S أو buzzer أو غيره.

note 3 BSRR , BRR

2 ريجستر في الـ GPIO لكننا شغلنا من غيرهم

فهما بيقدموا حاجة زيادة عايز أستخدمها تمام أو لا تمام.

* مثلاً عايز أعمل Set Pin في الـ MC فأنا بعمل

ODR |= 0x01 لو أنا قولت كدا

ODR = 0x01 دا غلط لأن كدا بعمل SetPin0 وأعمل Clear للباقي

فعلشان أعمل صح لازم OR ODR |= 0x01

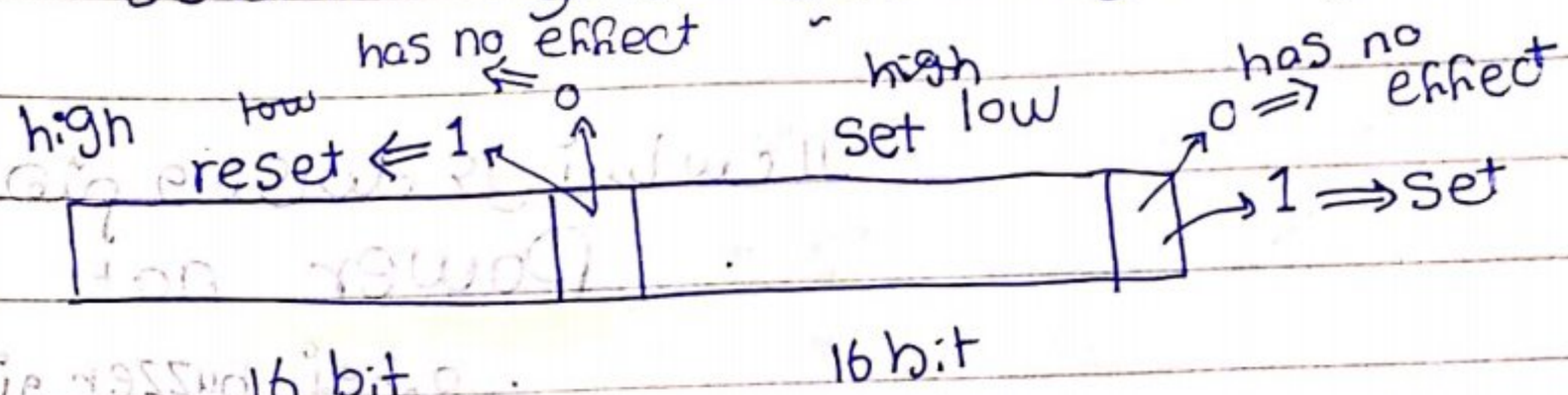
لو بال assembly ممكن ياخد 3 سطور.

↳ (Read) ODR

↳ change bit

↳ write back value to ODR

وهنا الـ MC قال لا أنا عايز أبقى efficient أكتر:
أنا هقدملك Reg اسمه BSRR
(Bit Set Reset Register)
دا بيعمل ايه بقى ؟!
هو مقسوم نصين وهو 32 bit



أي bit فيهم تكتب عليها 1 تبقى set
" " 0 تبقى has no effect

لو عايز أعمل set لرقم الـ pin أعمل ايه؟

BSRR = 0x01

من غير أهو أثر على الباقي لأن 0 هنا (has no effect)
وبكدا كتب one على pin 0 في instruction واحدة بالشكل مباشر
فكدا أقدر أسرع في عملية الـ write وكذلك جزء الـ reset جزء جزء

example

```
void setPin (u8 Pin)
{ //ODR |= (1<<Pin);
  // BSRR
  BSRR = (1<<Pin);
}
```

one instruction.
أسرع

طيب هايز اعمل reset

```
void Reset_Pin(u8 Pin)
{
    //ODR
    ODR &= ~(1<<Pin)
    //BSRR
    BSRR |=(1<<(Pin+16))
}
```

```
*BSRR = 0x00010001;
```

هنا 1 على set و 1 على reset

هنا بقى ال MC قال إن ال set

set has higher Priority

فكدا ال set ال هتنفذ وال reset لا

⇐ فدوره set و reset بشكل مباشر

① BRR: Bit Reset Register

32bit

reserved	1 Reset
	0 has no effect

* وهنا عايز مشكلة إنك اجمعي 16 على الـ Pin

عشان أعمل reset

```
void Reset_Pin(u8 Pin)
{
    BRR |= (1<<Pin);
}
```

Cases.

Example

↳ ① set Pin 0 & clear Pin 1 → BSRR

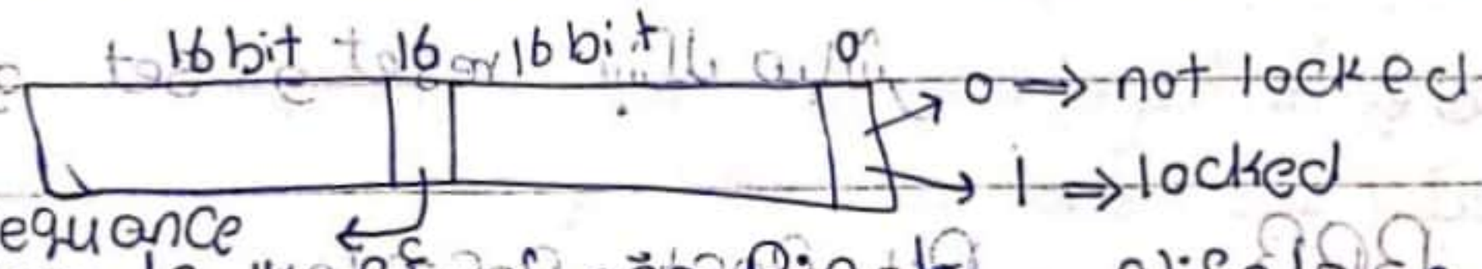
↳ ② clear Pin 0 & Pin 5 → BRR الأفضل

• ↳ ③ write value 0x3921 on PORTA ⇒ ODR

لأن هنا هقول القيمة دي بتساوي كدا الـ ODR مش
هنشوف عددالوحايد بقى وكلا زى BSRR و BRR.

Challenge:
speedup set Pin function
on DIO

5 LCKR → lock Register 32 bit



locked: معناه الـ Pin هقدرش أغير الـ mode بتاعها

ومش عايز حد مستقبلا يغير مود Pin معينة إلا لما أعمل
restant لـ MicroController.

بالنسبة لـ Pin ← 16 في sequance معين في الداتا شيت.

* يبقى كدا الريجيستر دا LCKR بيعمل lock لأي Pin
في CRL ، CRH مش عارفة تغيره كل Pin ليها anebit
في الـ register ، وكل Port ليه LCKR Register
مطلوب: في الـ function في الـ DIO تاخد رقم الـ Pin وتعملها
lock واسمها LockPin.

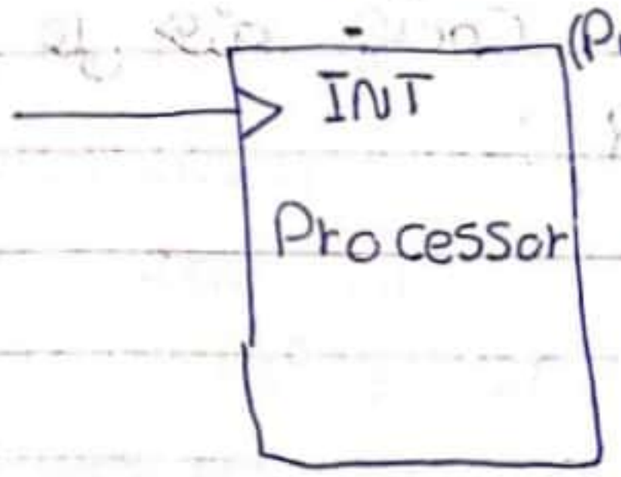
Interrupts

أولا: بشكل عام :

وهو معناه إني بعمل حاجة وفي حاجة قطعتني زي مثلا بالك وتلفوني ك سواء هترد أو لا.

* أي Processor في الدنيا عنده Pin اسمها INT. Pin وهي بتكون أول ما يجي عليها (Pulse) raising edge الـ Processor بيسيب اللود الـ بيعمله ويروح ينفذ الـ INT، فيبدأ ياخد action إن في INT قطعته سواء هيروح ينفذه أو لا، ولما تجيله الـ Pulse دي بيفهم إن فيه event مهم من الـ بيعمله، فيبدأ يـ act عليه.

raising edge



وهي عبارة عن Pin واحدة بس من الـ Control Bus بتاع الـ Processor

* Interrupt types

مين بقى بقدر يعمل interrupt ؟

- External INT → بالنسبة للـ Processor يعني حاجة جواه أو براه هي اللي بتعمل INT
- Internal INT → 3 أنواع جوا الـ Pr بيقدر يعملوا INT ؟
  - Core Peripherals
  - Trap (علطول انا عملتها في اللود)
  - Software Interrupt → وهي عبارة عن assembly instruction SWI ودي لما الـ Processor بيلاقيها في أي وقت بيقول أنا هعمل interrupt وينفذ لود الـ INT.

MC Peripheral
1) UART
2) ADC
3) Timer

Trap:
1) Division by zero — لما تكون بتدخل data أثناء الـ runtime
2) wrong op code — لو غيرنا opcode في hex ومش موجود مش لود عادي
3) Stack overflow — لو عرفنا var أكبر من مساحة الـ Stack

What happens when Processor got interrupt..?

بيبدأ ال Processor

→ Jump to specific Port of code called Intrrupt Service Routine (ISR).

يعني دلوقتي ال Processor شغال في ال main عال
ينفذ سطور وفي سطر معين جاله ISR فهيسيب الفي ايده
وهيروح على حته Code اسمها ISR

```
main()
{
  =
  =
  → ISR
}
```

```
ISR()
{
}
```

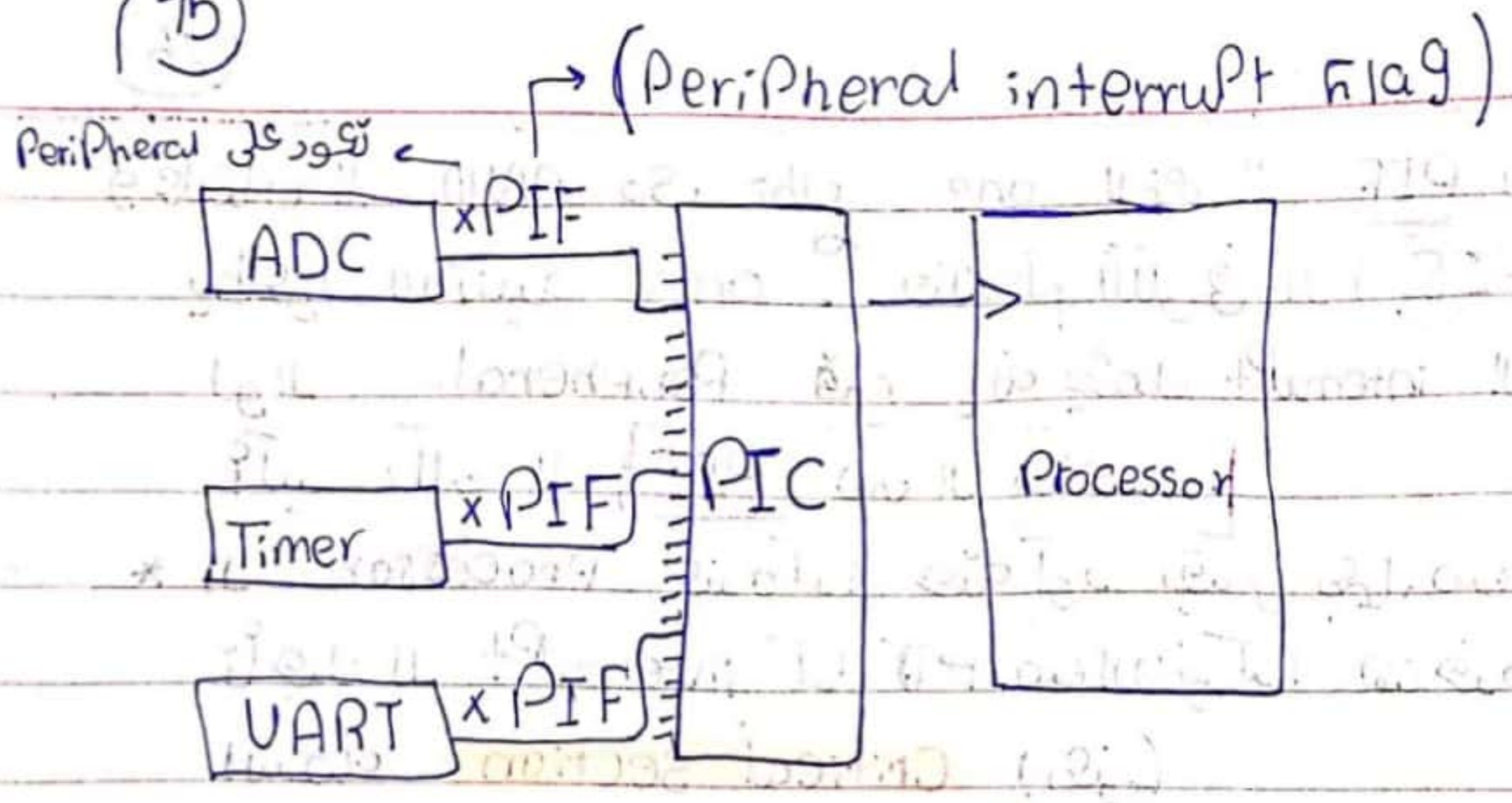
يبدأ ينفذ اللود بتاعها ويخلصها ويرجع لنفس المكان ال وقف فيه
سواء ال ISR دي جالها عن طريق hardware signal على
بناية INT بتاعته ☐ أو من Code Periph أو من سواء
اللود ال كان بينفذه عمل لقى فيه Trap أو لقى instruction
اسمها SWI.

يعني ايه يعني SWI؟

أولاً: مطلوب: ايه الفرق بين Call func و SWI ؟
امتى أعمل func وامتى SWI ؟

Interrupt Handling Techniques:

ازاي كل ال Peripheral هتوصله على نفس ال Pin
INT → ☐ ؟
هتودي ال INT



PIF : السلكة دى بتبقى بـ one لما ال Peripheral حاجة يعمل interrupt ، مثلا لما ADC يخلص Conversion ال PIF يبقى one ، وال Timer يعمل overflow ال PIF بتاعته تبقى بـ one وهكذا

والسلكة دى لو وصلت direct بال Pin بتاع ال Processor كدا هيدخل فى interrupt على طول

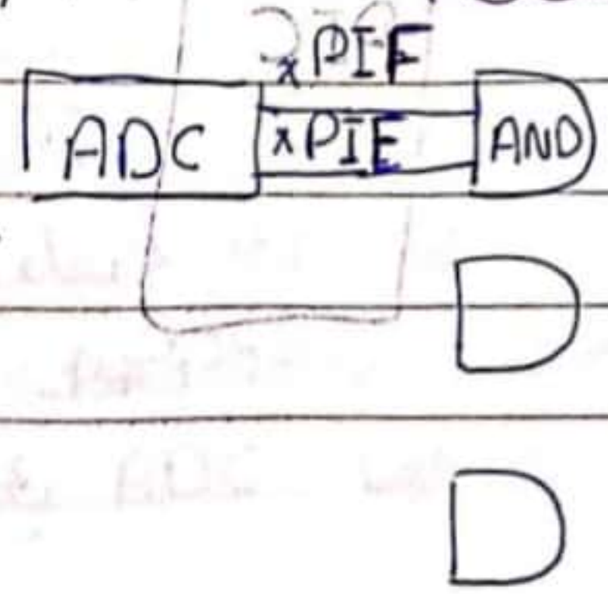
طيب ازاى أوصل Peripheral كبير بـ Pin واحدة

هنا هنحتاج Peripheral فى النص اسمه PIC (Programmable Interrupt Controller)

فكدا لما حد فيهم يعمل interrupt بتبقى الـ PIF بتاعته بـ one فال PIC هيباعت الـ Signal للـ Processor

طيب افرض إنا أنا مش عايز الـ ADC يعمل interrupt أو مش مهتم بيه ، عايز أعمله disable ، قالوا خلاص احنا مش هنوصل الموضوع بشكل direct كدا .

هنعمل لكل bit هيكون ليها xPIE (Peripheral interrupt Enable) ودى بنتحكم فيها بالـ Software ،

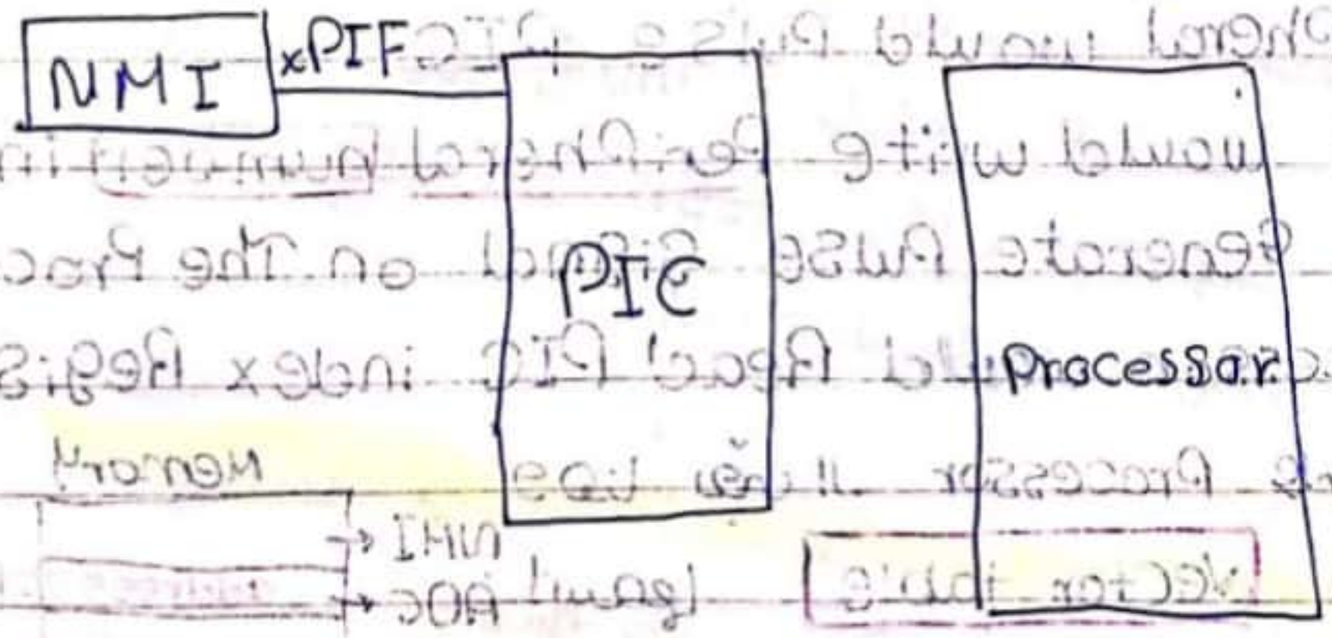


وكدا خليت كدا Peripheral يقدر يعمل INT على ال Pin الواحدة الاعلى ال Processor.

بالا Peripherals معينة وهى كالعادة عندها xPIF وداخل direct على ال PIC من غير AND Gate ولا ليها GIE ولا PIE وده اسمها NMI (Non Maskable Interrupt) اما تيجى مينفعش أقوله ل Peripheral ده reset متحصلش زى مثلا ال reset وال بيأبلاى Signal على ال Processor و بيقوله سيب ال في ايدك وأعمل apply على الكود تانى ، فمش منطقى ال Processor يقوله لا استنا ، فكان كد متوصل direct على ال PIC

* لحد هنا كل ال Processor متوقفة و بتشتغل بنفس الطريقة.

و الباقى زى مثلا ال ADC و Timer و UART بنسميهم Maskable: ينفع أقفلهم يعنى من خلال GIE و PIE



[1] مثلا بقى ال هيحصل لو جالى من Interrupt عمل Pulse فى نفس الوقت

تعالوا نجاوب سؤال تانى الأول هو لما ال PIC يبعت Pulse لل Processor ويقولى حاجة بالك فى interrupt حصل ازاى ال Processor بيعرف أن ~~ده interrupt~~ ال Peripheral عمل interrupt؟

ماهو برده الكود الجاى تعمل لما ADC يخلص Conversion

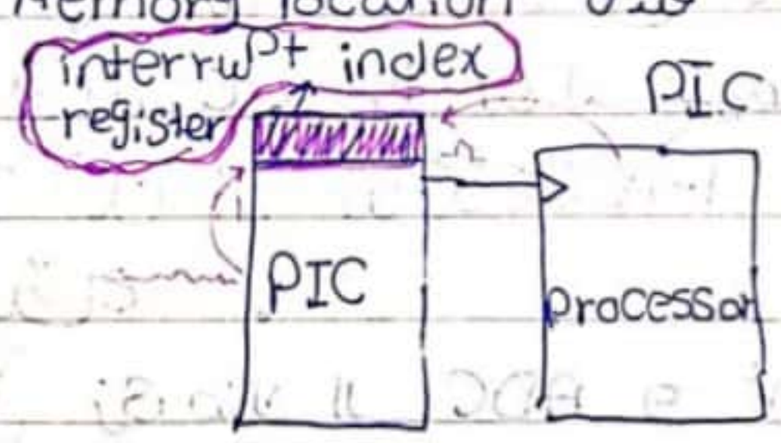
غير اللود الحاجز تعمله لما UART يخلص recieve 2
ف ال Processor هيفهم ازاي لما تجيله Pulse
من ال هيقوله إنت عندك ال Peripheral اللي عمل INT
هنا بقى تنقسم الناس إلى اتنين قبل ال ARM طبعاً

1) Vector table Based System. دا نوع للتعامل Processor

← بيعمل ايه بقى:

بيكون عنده register عنده Memory location اسمه

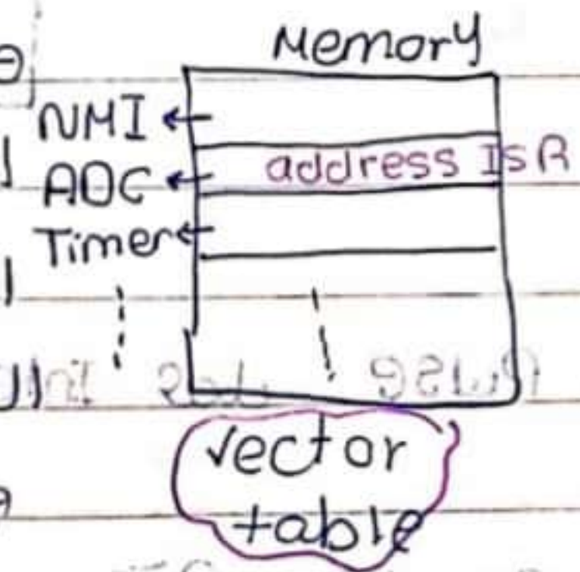
← Interrupt register index دا في ال PIC



*** Scenario ال هيحصل كالآتي

1) Peripheral would Pulse PIC
2) PIC would write Peripheral number in index.R
3) PIC Generate Pulse signal on The Processor
4) Processor would Read PIC index Register

وهنا بقى ال Processor عارف حته ميموري
ممكن تكون حته من اسمها Vector table
ال Ram أو ال Flash ، هنا أول ما
ال Processor بيجهل interrupt يقرأ رقمه
و يـ jump على حته الميموري دي
وهتكون عبارة عن جدول locations بنفس
عدد ال locations اللي في ال PIC وبنفس ترتيبها ، كل location
فيها أبي responding لل Peripheral معين .



5) processor Jump on vector table to corrosponing index

6) processor would find address of the corrosponding ISR.

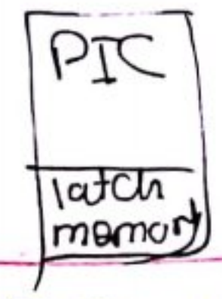
مثلا ISR_Timer
{
}

ISR_ADC
{
}

any location in flash

كل ال ISR هينزل فين في ال flash ؟
مش عارفين دا حسب مساحته فيه كل ال Tool و إنت بتبني build
اسمها linker هي اللي بتقول كل function عندك هتنزل فين
في الميموري وإمتى مكان ثابت.
بس ال location اللي هو اتحط فيه ، فيه عقوبة
بيجي بيشوف كل function من دول اتحطت فين في الفلاش
وياخد ال address بتاعها ويكتبه في ال vector table
في مكانه الخاص بيه ، ولما يجي لل Processor ، INT
مثلا بتاع ال Timer يعمل jump على مكان ال Timer
في ال vector ، هيلاقي فيه address بتاع ال ISR يعمل jump
عليه هيلاقي الكود هينفذه وهكذا

سؤال ال vector table ليه مكان ثابت في الميموري ؟
مكانه ثابت صح
لكن ال locations اللي جواه data فيه مش ثابتة الا ال address



طيب نرجع للسؤال لو حصل 2 interrupts هنلاقي ال ADC وال timer في نفس الوقت ؟

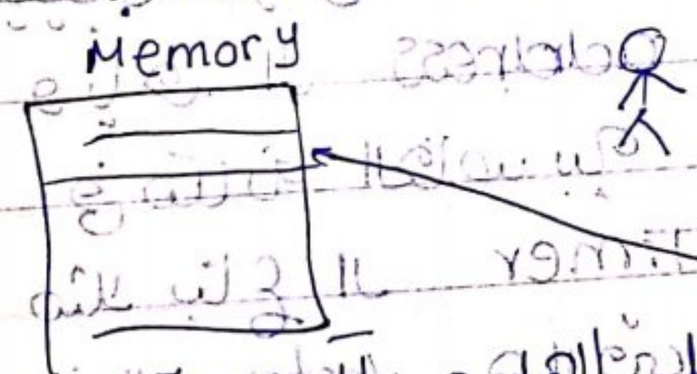
ال PIC قال هيبقى عندي حته مبعوري كدا اسمها latch memory لو حصل أكتر من interrupt هعملهم save في ال latch دي وأطلعهم واحد واحد ، طيب هنطلعهم بأنى Priority ال PIC ، قال أول واحد فوق هو أعلى Priority وال جاي بيقل ، والترتيب بتاع ال Peripherals العليا هوال بيمثل ال Priority يكون على حسب رسمنا إن NMI أعلى وبعده ADC وبعده Timer وهكذا وبالتالي ال Priority بتاعه ال INT سيتعتمد كلياً على المصنع بتاع ال MC الركب ال Peripheral .

∴ أول نوع ال هو ---- vector table سهل جداً

سؤالين :

① فين ال vectortable في ال Flash ولا RAM ؟

② فين الكفوبت ؟



لو حصل INT ؟

عيوبه ان ال هينفذ الأول معلومة على الهارد بالك مقدرش أغير

النوع التاني

2 Flexiable Priority System

لما يحصل INT ال PIC يقول أنا معندش index register أنا عندي معلومة أقولها لل Processor أني Peripheral ال Processor كدا في ورطة ! .

هيعرف ازاي ؟ كدا ال Processor قال أنا كمان هلغي ال vector table وأياً يكن ال Pulse ال هتيجي هعمل Jump على location اسمه Interrupt vector

address ISR

Senario

1) PIC Generate Pulse to Processor

2) ~~Jump~~ Processor Jump to Interrupt vector

* إنت بقى كـ developer هتعمل ايه؟

هيعمل function اسمها ISR واحدة لكل الناس

interrupt vector الـ address بتاعها موجود في الـ

فالـ Processor بيعمل jump to ← interrupt vector لـ find

الـ address بتاع ISR ؟

إنت بقى developer إعرف الـ INT بتاع

أي Peripheral فتعالى نعمل ISR

ISR

```
{
if (PIF_Timer == 1)
{ ≡
}
if (PIF_TimerADC == 1)
{ ≡
}
if (PIF_UART == 1)
{ ≡
}
}
```

5

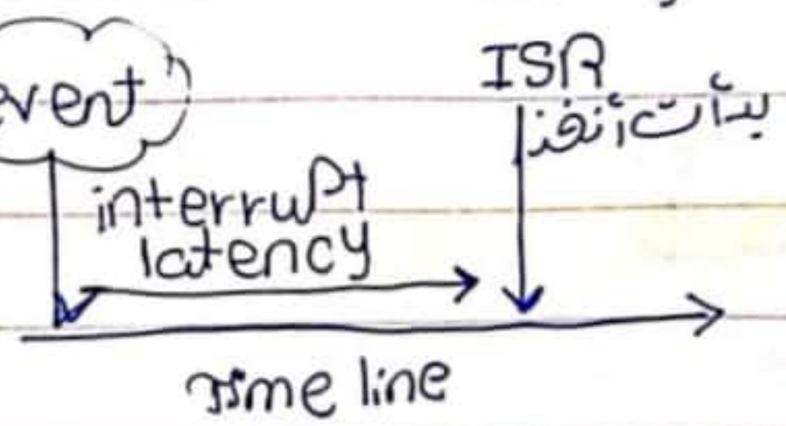
طيب استفدت ايه ؟

* الحصل إن لو حصل 2 interrupts في نفس الوقت
ال هينفذ هنا هو ال إنت سألت عنه الأول في ال ISR
كلوه ك if ← conditions
ودا Flexable Priority بال SW (software)

كدا بقى أنى أفضل vectortable or flexable Priority ؟

ATMel AVR → vectortable	PIC Microchip → flexable Priority
* fixed H.W Priority	* Software Priority ①
* low interrupt latency (fixed) وهو نفسه لك ... ①	* high latency & variable.

تخيل هنا



INT ← generate ال event ال بيـ دا ال

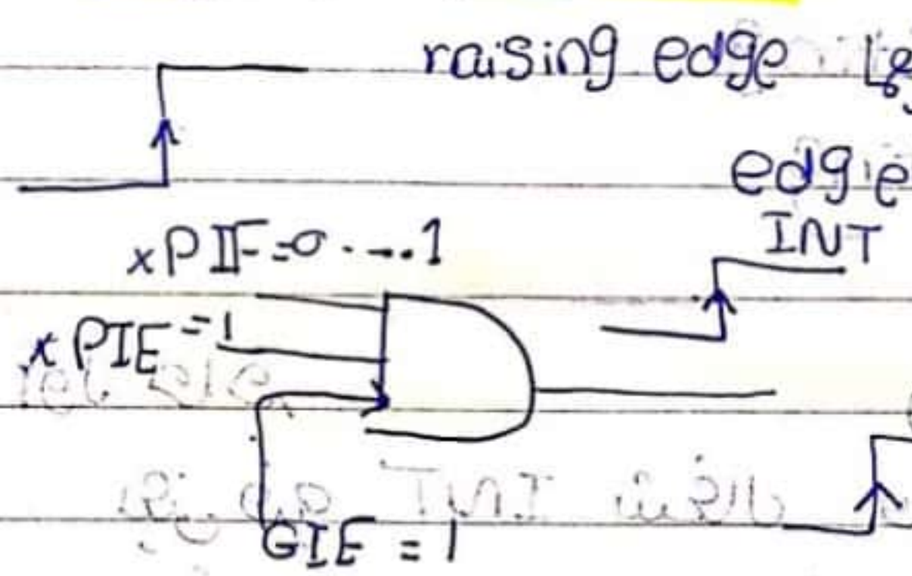
interrupt latency : هى الوقت بين حدوث INT وبين تنفيذ أول سطر في ISR ، وكل ما تقل تكون أفضل وأسرع
وهى أقل في حالة vector table لأنه شغل H.W
، لكن في ال S.W مع أول سطر كدا خلاص يا إما كدا يا إما كدا
وجد هيزود ال latency

طيب الـ ARM بقى :

قال أنا هديني الـ تدبير وهعملك اختراع عظيم جداً اسمه NVIC ← ودا بتاع الـ interrupt بتاع الـ ARM.

ولما احد دلوقتي دا الـ consept بتاع الـ INT لأي MC

[9] Flag Clearance :



لازم يحصل عليها raising edge

يعني هي edge sensitive

هنا مثلاً الـ Timer عمل INT

والـ PIF دلوقتي one

وخلاص الـ INT اتنفذ الكنترول الـ flag الـ PIF

لسه one ، فلو جه INT تاني هو مازال one

هيحصل INT ، فلازم نعمل clear

علشان أسطح إن لو في INT جديدة هيعمل edge جديدة

وأقدر أحسس بيه

* لازم نعمل clear للـ flag بعد تنفيذ الـ INT :

* عشان نعمل clear فيه 4 طرق :

كل MCU هيختار 1 يعمل بيهم clear للـ flags بتاعه

[1] By Hardware, once ISR executed.

[2] By writing zero to the flag

[3] By writing 1 to the flag

[4] By Reading the flag مجرد ما نقرأ الفلاج هيعمل clear

xPIF;

5 Interrupt Nesting :

وأنا شغال في INT حصل INT تاني :
يقطعني ولا لأ؟

لو قطعه اسمه nesting ولو مقطعهوش مبقاش nesting

وأغلب ال Processors مش بيـ Support ال nesting

* طيب ال Processor ال بتـ Support ال nesting
يتفرق بين نوعين من ال nesting :

* Normal nesting
* Self nesting

1 Normal nesting

أول حاجة

يعني فيه INT شغال وفيه INT مختلف عنه
وأعلى منه في ال Priority حصل :
↳ Interrupt 1 is executed, INT 2 (high Priority) happened

⇒ * what is the max nesting level ?

ممكن واحد يقول والله أنا عندي واحد بس وممكن 2 or 3
وهكذا على حسب نوع ال Processor

Max level

فكدا في حالة ال Normal تحديداً لازم نسأل عن ال

2 Self nesting :

↳ INT 1 is executed, INT 1 happened again

يعني مثلاً ال ISR بتاع ال Timer لسه مخلصش
راح ال Timer حصل overflow تاني، ودي حاجة وحشة
بتبوظ ال design وبتاعك.
لازم نخلي ال General Interrupt finished الأول :

```
func()
{
    u8 x;
}
```

→ Save context assembly → Context Switching

Push x

هنا الـ Compiler بيعمل Save للكود

سؤال عن

The NVIC

Nested Vectored Interrupt Controller

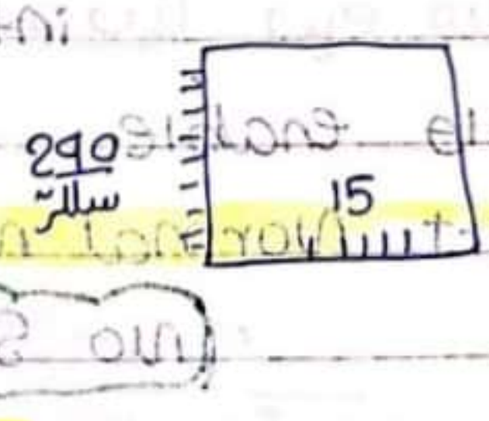
↓ vector table

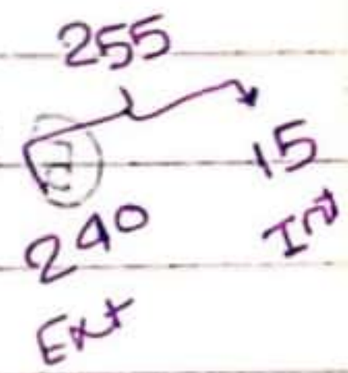
1] Core Peripheral ⇒ ↓ latency + Standard (ARM تبع)

2] Support 255 interrupt

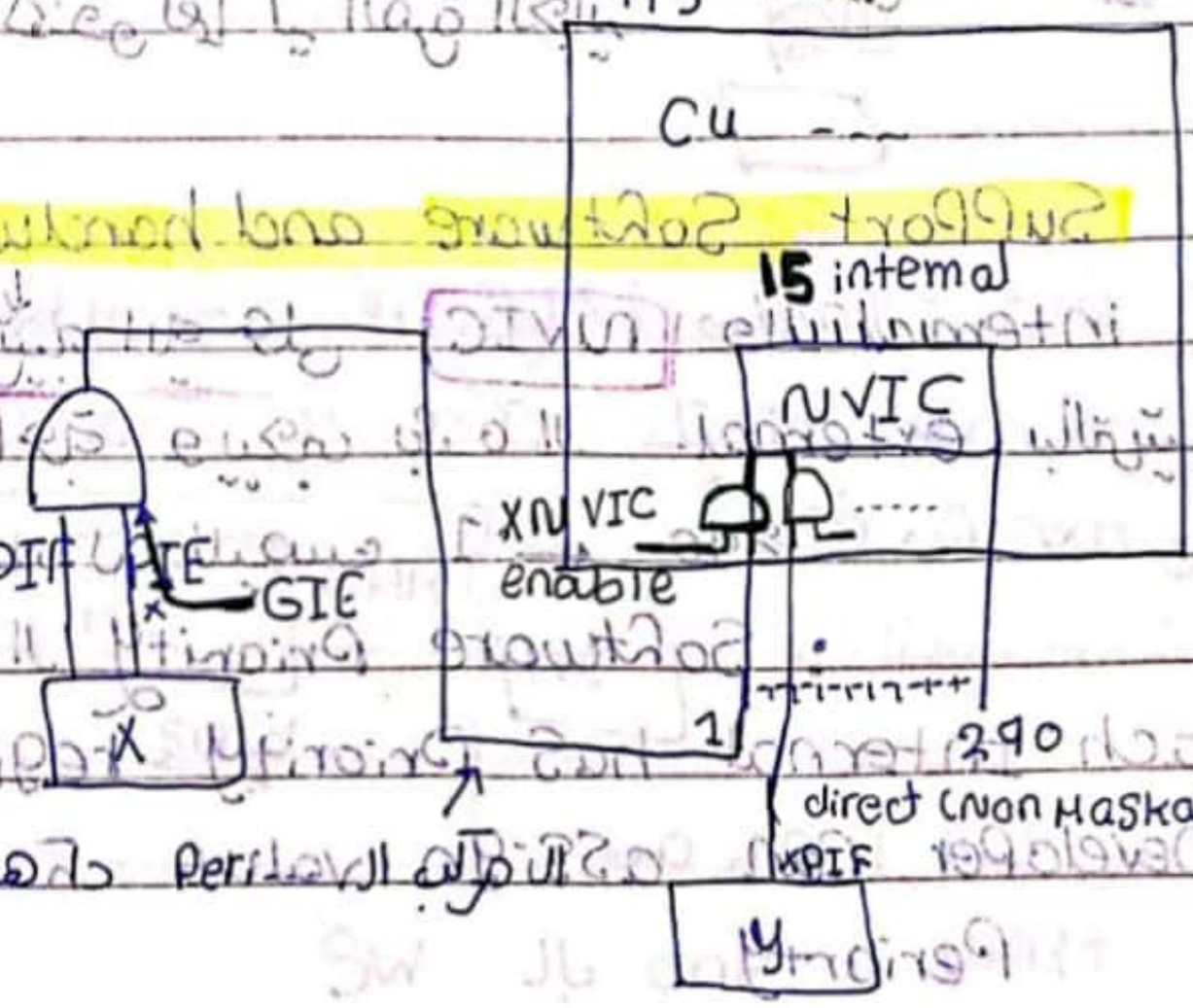
240 — 15

(Processor برة الـ) external — Internal





3] Vector Table Based System.



هنا بقى NVIC قاله أنا معندیش الكلام دا
أى Signal طالع من دول (Peripheral) قبل ما تدخل
على ال NVIC بتدخل على AND Gate ، داخل عليها ال Signal
ال جاية من ال vendor وال Signal بتاعه xNVIC enable
وبتبقى يا 0 يا إما 1 ، وفيه enable لكل سلك
* وفى 2enable :
واحد عند ال Peripheral بتاع ال vendor وواحد عند ال NVIC
* ليه ال NVIC عمل AND Gate عنده لكل Peripheral
لأن ال NVIC أول ARM عايزين ال developer هو المسئول
عن اختيار ال System بتاعه مش ال vendor ودى رابع ميزة
[9] Independent Enable Pin for each external interrupt.
كل external ← INT... عنده enable واحدة.

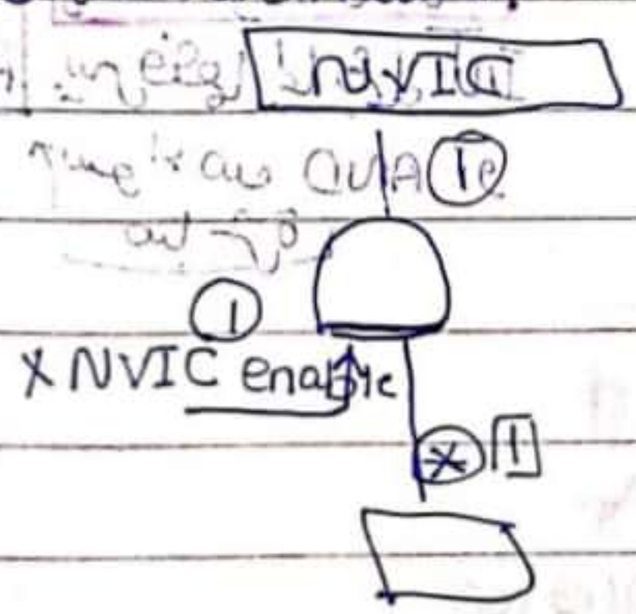
(5) Support Normal nesting
(no Self nesting)
(كام level بقى بتجاوب عليها المرة الجاية)
In General (255)
Cort M
بس (16) لكن عندنا فى ال Micro controller بتاعنا
Cortex M هنقول لها ليه المرة الجاية

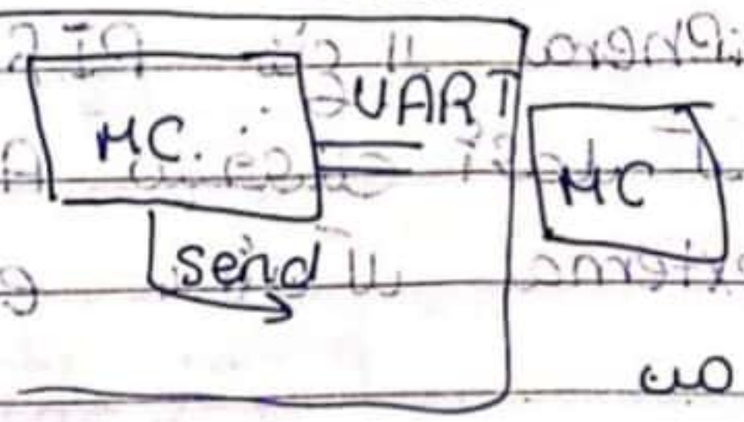
ميزة خطيرة (6) Support Software and hardware Periorty
حسب ترتيبه اللى على NVIC وال internal
أعلى حاجة ويجى بعده ال external. بالترتيب كدا
من الشمال / لليمين 1 ← 255
* ، فى حالة بقى ال Software Priority :
* for each Internal has Priority register
* SW Developer can assign value for Interrupt Periorty

لو 2INT حصلوا في نفس الوقت
ال NVIC هيروح الأول يبص على ال Software Periorty
ال dev قررتها ويشوف فيهم مين عنده S.W Priorty
أعلى يعني قيمة أقل 0 → highest
255 → lowest
وكل واحد فيهم (INT) ليه Regis. مخصوص، هيبص على ال
registers بتاعتهم، ~~شوف كدا إنت~~ ويشوف كدا إنت كاتب
مثلا في Register 5 والتاني 10، فال 5 أعلى Periority
بغض النظر عن ترتيبهم في الهاردوير ازاي، هو بص على ال SW
ال قررها ال user، طيب ما يمكن الاتنين فيهم 5
ما ينفع وفي الحالة دي هلجأ لل hardware Priority
حسب مكانه فيه لأن مش هيتساووا أبدا.

في ال مميزات

7 Independent Software
set flag for each interrupt.



هنا بقى قالك أنا هعملك أخليك بالسوفت وتكتب one :
overflow عمل Timer هنا على فتمثل وتأخذ
وهو مش عملش فعلا


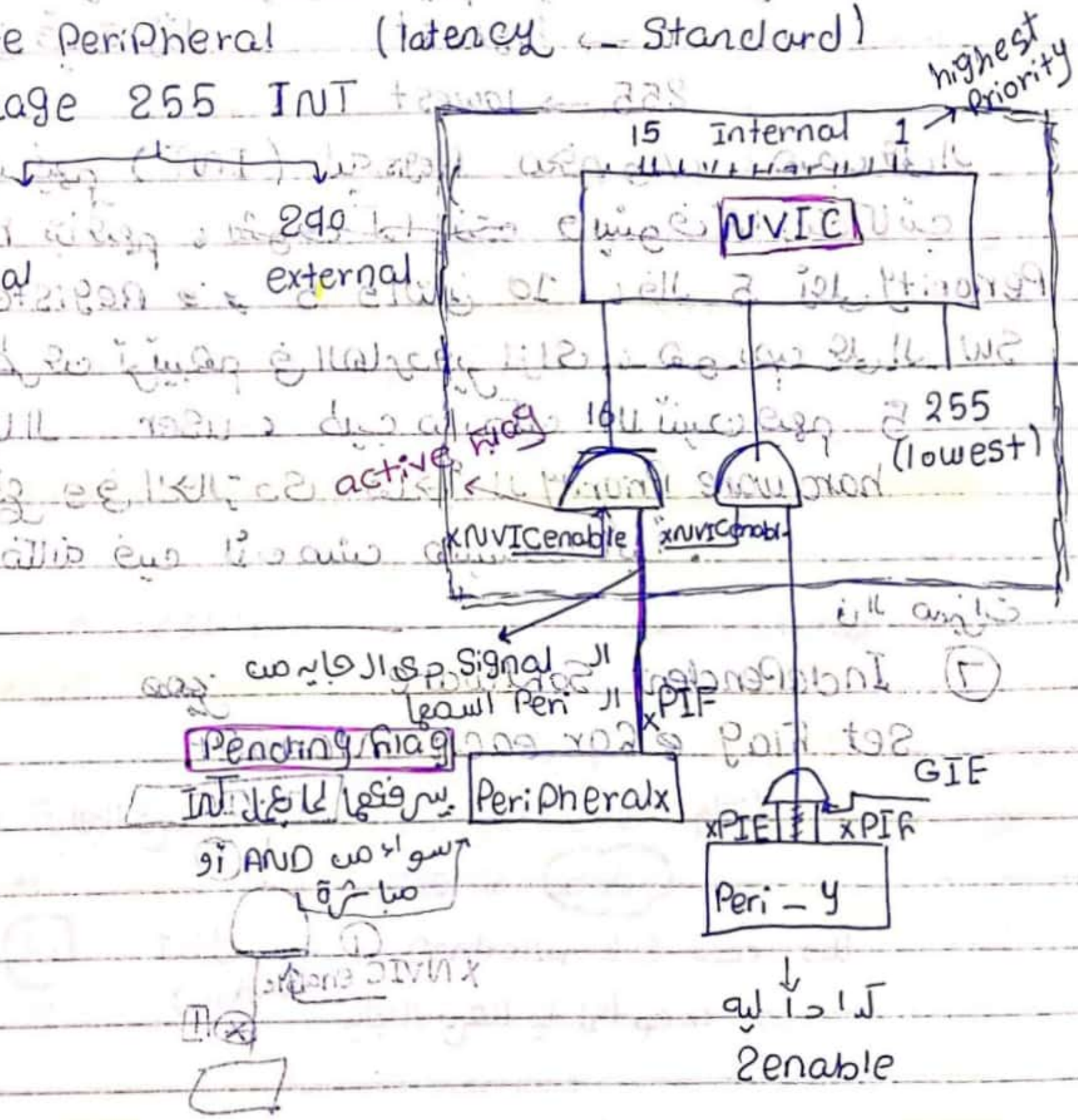
فهتكتب على السكلر الجاية من
ال UART ← one بال SW

End of lec 7

Lec 8

NVIC

1. Core Peripheral (latency ← Standard)
2. Manage 255 INT



سؤال : كل الـ external Peripheral are Masked ليه؟ جاية في
الكلام طلع Timer في general لأن ممكن
حد يوصل الـ PIF بتاع الـ Peripheral مباشرة على الـ PIC
* لكن في ARM هينفعش أعمل كدا ، لأن تقدر تـ Mask
الـ external لأن كل external ليه enable خاص بيه

ناخد معلومة

③ each External interrupt has Two flag flags → Pending flag
→ Active flag

مش باين قدامنا

لزمنه لما ال INT يرفع ال Pending flag بتاعه ، فإما هنا ال NVIC يدخله لل Processor وينفذ وساعتها بنشيل Active أو ال NVIC مش هيدخله ينفذ ، طيب ليه؟

* لأن بإما disable من عند ال NVIC فمش هيقطعه
* أو إن في INT أعلى منه في ال Priority شغال فيقطعه
* أو إن نفس ال INT موجود قبل كده وبنسميه self nesting ودامش موجود في ال NVIC.

Active flag	Pending flag	State
0	0	No Interrupt
0	1	Interrupt happend but not excuted
1	0	notenable higher Priority interrupt is executing
1	1	Same interrupt is excuting (not self nesting)

State : Interrupt is executing

State : Interrupt is executing and another version of it fired
(معناه إن INT حصل)

interal external

4] NVIC Manage (255) all interrupts but Controls (extenal) only external.

- enable
- disable
- Set Pending flag
- Clear Pending flag
- Change Priority

يعني أي INT عايز يدخل الـ execution يبدي على NVIC
والـ NVIC بقى على حسب هيديه ولا لأ
فدا الـ Manage
(Pass, not Pass)

* فأنا لأني لك internal بنظم الموضوع بس
* طيب هل مقدرش أعمل Clear و Set لو هكذا لـ internal؟
لا ينفع بس مش من خلال الـ NVIC.

ex SysTick (Timer, Core Peripheral)
هنا الـ NVIC مقدرش يعمله Clear ولا Set
ولكن الـ SysTick نفسه أقدر أعمله

5] Each Interrupt has Two Priorities:
- Software Priority 0 : 255
- hardware Priority → Position of interrupt @ NVIC

Software Priority:
يعني Register للـ INT بتبنا قيمة للـ S.W Priority بتكون من 0 : 255 ، وعنده H.W.P على حسب الـ Postion
فاهم على NVIC لما كان أقرب يبقى أعلى (Priorty)
، الـ Positions دي فلها أهمية إلا لو أكتر INT حصلوا مع بعض أو واحد حصل أثناء ما واحد شغال

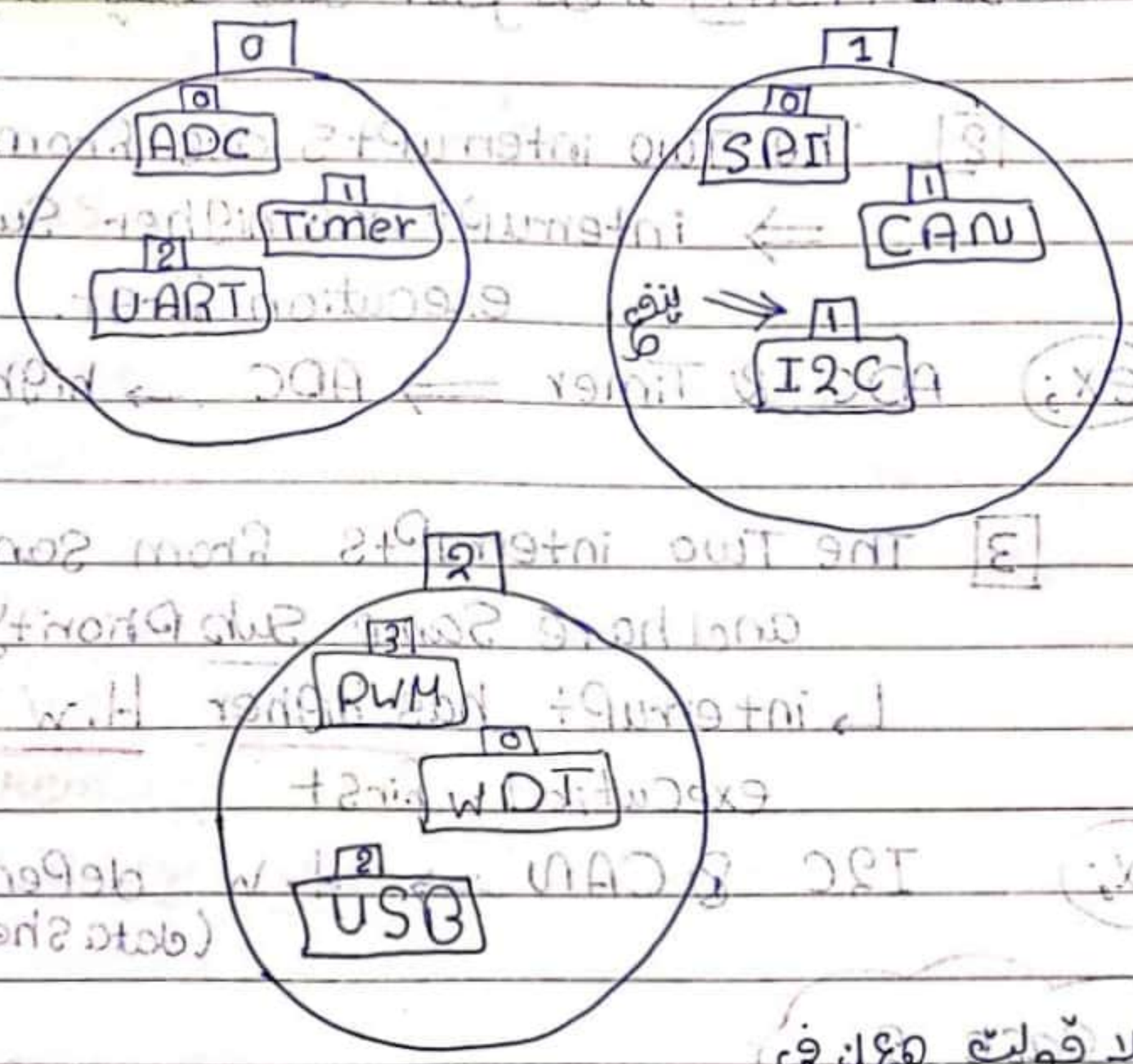
6 Nesting, NVIC supports Interrupt nesting (normal nesting)

no self nesting,

* Interrupt Grouping → Group Priority (خيالية) / Sub Priority

دا الهيخدمنا ال nesting دا عامل ازاي:

example:



أنا مثلا قولت هعمل في System بتاعي 3 groups وكمان جواب أنا هختار ال Peripherals اللي فيها example interrupt جوا ال group بتاخد Priority يعني مثلا ال Timer

وينتمي الجروب ده Priority 2 وال x Priority بتاعته نفسه one. وهنعتبر إن ال Prepherals اللي في نفس الجروب إخوات والاخوات ميقطعوش بعض يعني مفيش nesting الا بقطع الجروبس المختلفة

Case 1:

⟹ Two interrupts happened at some Time:

1. The Two interrupts are from different Groups

ex; Timer & SPI ⟹ Timer → higher Group.P execution first.

* مادام في جروبين مختلفين يبقى ال عنده Group Priority أعلى هو ال يتنفذ بغض النظر عن ال Sub Priority

2. The Two interrupts are from same Group:

⟹ interrupt has higher Sub Priority execution first.

ex; ADC & Timer ⟹ ADC → higher Sub.P

3. The Two interrupts from same Group and have Same Sub Priority:

↳ interrupt has higher H.W Priority execution first

ex; I2C & CAN ⟹ H.W dependant. (data sheet)

Case 2:

Interrupt happened while another interrupt is executing.

1. Leading interrupt has higher Group Priority ⟹ nesting, otherwise No nesting هيستنا

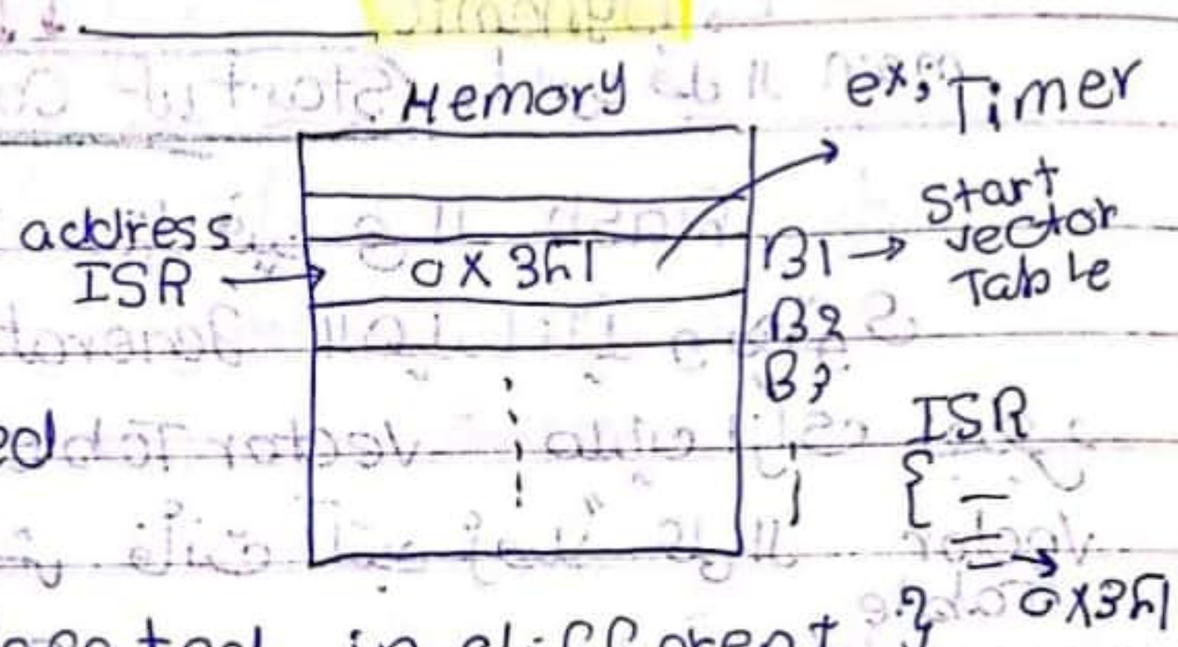
* How to disable nesting ARM Cortex M?

نحط كل ال INTs في نفس الجروب وكدا مفيش nesting ومفيش حد هيقطع التاني

* How to allow nesting between all The INTs ?
  بكون كل واحد يبقى في جروب لوحده .

* Maximum level of nesting بال
  ((مساوي))

vector table



Facts:

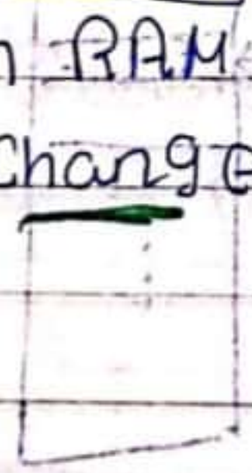
① vector table has fixed location.

② ISR Routiuen are located in different location in flash memory (based on Project)

Vector Table

① Static vector Table
↳ in Flash Memory
↳ no change in runTime So This Static

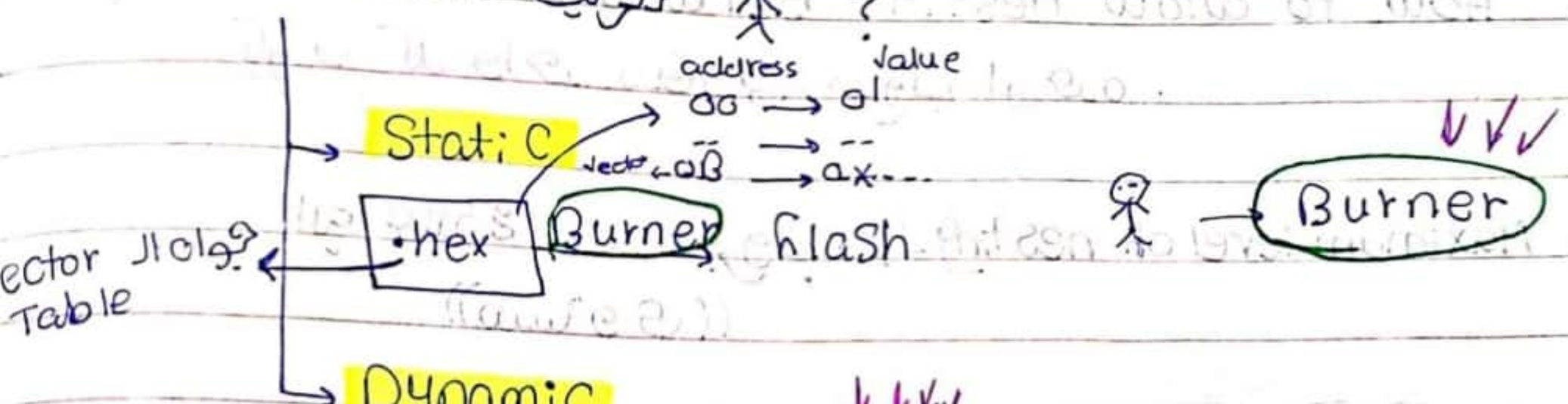
② Dynamic Vector Table
↳ in RAM
↳ Change in Possible in runTime



ISR_Timer
{ }
0X3F1

ISR_Timer
{ }
0X5A0

Who is The عفريت ?



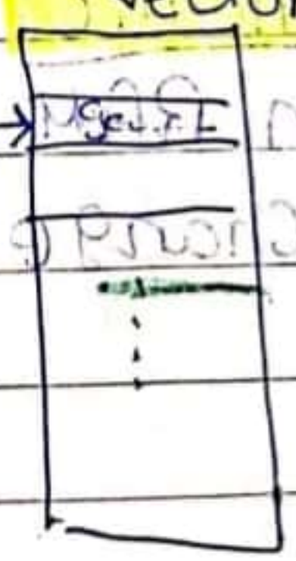
* معلومة :

① ال linker هو اللي بيحط كل ISR هيتحط فين في ال flash وهو نفسه نفس ال بيـ generate الهيكسا فايل ويحتوي على المعلومات اللي بتقول vector Table مليان ازاي بمجرد ما تحرق ال hex. على الفلاش فأنت كتبه أصلاً على ال Vector Table

② * من وظايف ال startup code : إنه يكتب على ال vector table ، إنه يقرأ ال addresses بتاعة ال ISR دي فين في الفلاش ويكتبها على ال vectortable اللي في ال RAM.

تخيل لو جه INT قبل ما ال vectorTable يكون اتملى بال addresses اللي هي صح ؟ هيلاقي garbage والله أعلى ايه اللي هيحصل

vector table at Ram



لكن هيلاقي بقول ال startup code قال هيعمل disable all interrupts initialization علشان يعمل vector table وبعدين يفتح ال INT تاني ، أحسن بيجي INT

(96) Technical Sat 2:6

كدا ال Static هو الجاهز إنه يستقبل INT في أي وقت
لأنه مكمله initialization قبل ال runtime بواسطة
ال (Bumer)

أما ال Dynamic فهو فيه إننا محتاج أستنى
فترة في ال runtime علشان أعمل initialization
لل vectortable بواسطة ال (Startup Code)

vector Table

طيب هنا ال (ARM) ال default بتاعه هو (Static)
هل دا معناه إنه مبيعرفش يشتغل Dynamic؟ لسه هنشوف

Software, End

معلومة : ؟

① Private.h & Config.h مفيش component تاني
معايا في ال Proj ممكن يشوفهم ، ال interface.h هو الوحيد اللي الناس الباقي
بتشوفه

② Private.h مينفعش اليوزر كمان يعدل فيه ، لكن Config.h
يتفع يعدل فيه

NVIC

ترتيب ال Periph
Peripherals order
Connection
يعني عايز تبص على شكل ال
vectortable من واحد
على ال bins علشان ال Priority (h.w)

Register Definition
& addresses
at ARM.

at data sheet

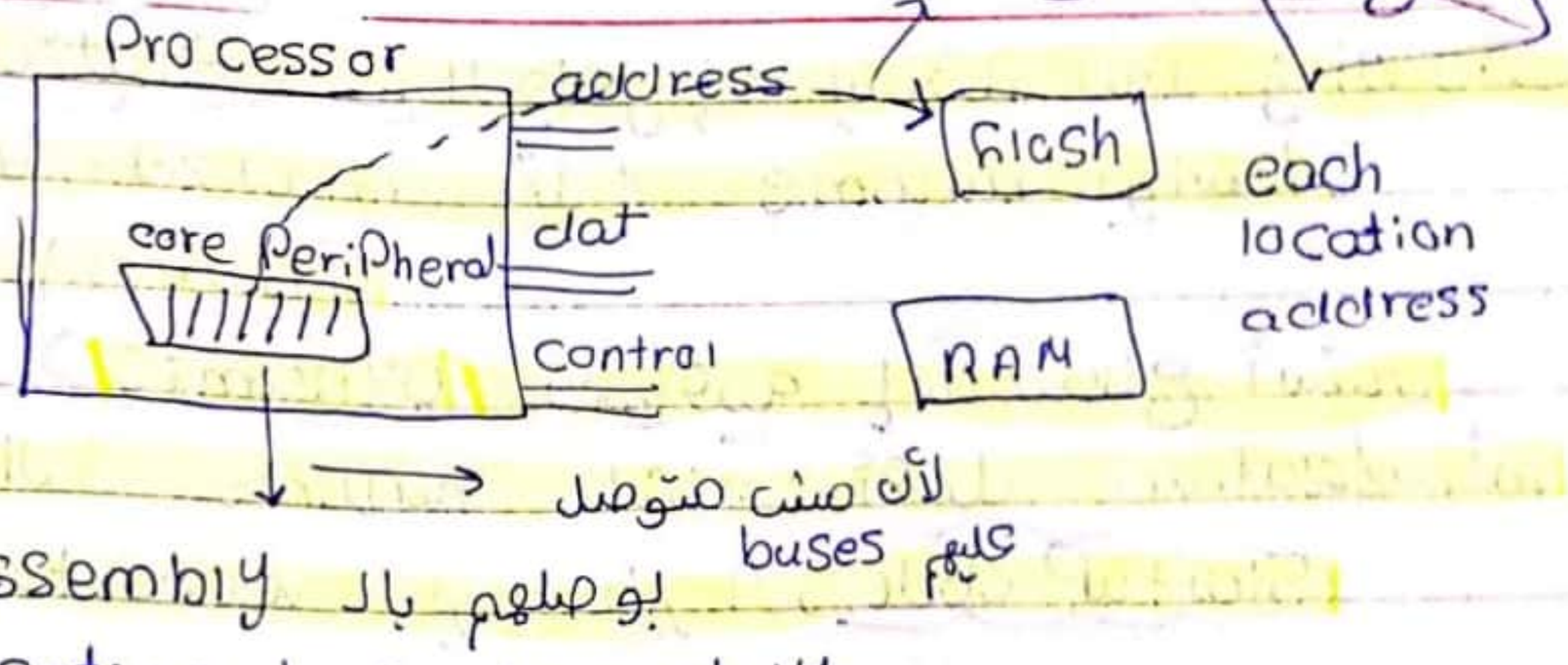
Settable : Can change
Priority

Hard fault : Trap

ST

(Mirroring) نسخة تانية بعني

Memory Mapping



* each location has Assembly instruction "index"

كل ال Core Peripheral أو مقدرش أعمل access بالسي
لأن أصلاً ملوش address ، مش واصل على address bus .

طيب هنعمل ايه هل هنكتب assembly ؟

هنا بقى ARM قال لك ، أنا هعملك حاجة تساعدك
بدل ما تكتب assembly ، أنا هعملك حاجة اسمها

Memory mapping

وهي إني هجيب الـ addresses بتاعة ال Peripherals
الداخلية ومعملوه Mapping على الميموري برو
* وأخليه ياخد address على ال address bus وأكلمه
بالسي عادي .

هل ARM هيعمل كدا في كل الـ Core Peripherals
قال لك بعضهم بس وبعض ال registers بس

هنا بقى NVIC ← Memory Mapped
بالسي بقى

لأن ليه addresses ، لكن بعد كدا هنلاقي
البعض ملوش
وهنكتب assembly

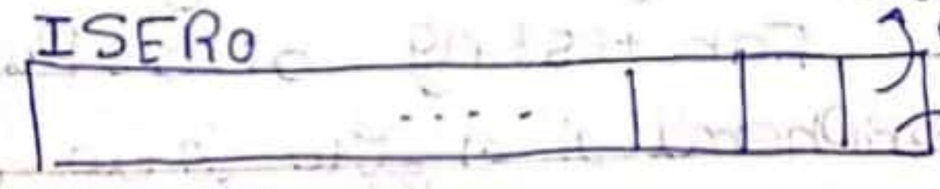
→ NVIC Registers

1. ISERx → Interrupt Set Enable Register

32 bit ← يعني بيتحكم في 32 ← INT ، NVIC بيتحكم في 240 بس

فكذا بيتحكم في ال

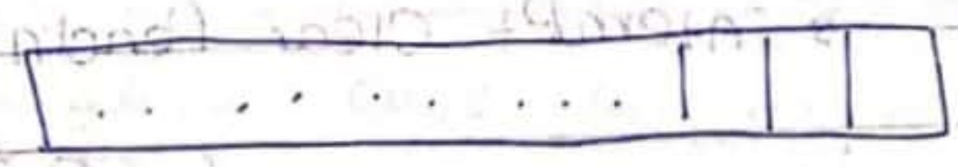
ISER0



كل INT ليه bit لو 1 بيقي enable external INT

0 to 31

ISER1



32 to 63

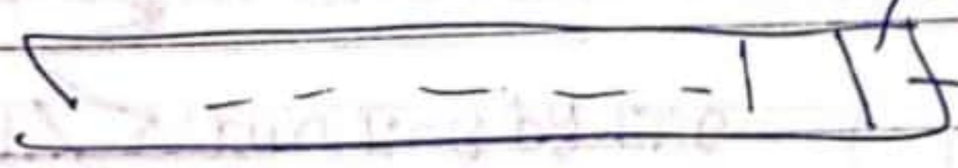
يعني هو واحد لكن 32 INTs وأساساً في ال data sheet لحد 59 فأنا هيهمني ISER0,1 only .

ال ISER هو اللي بيكتب على ال Pin دي.

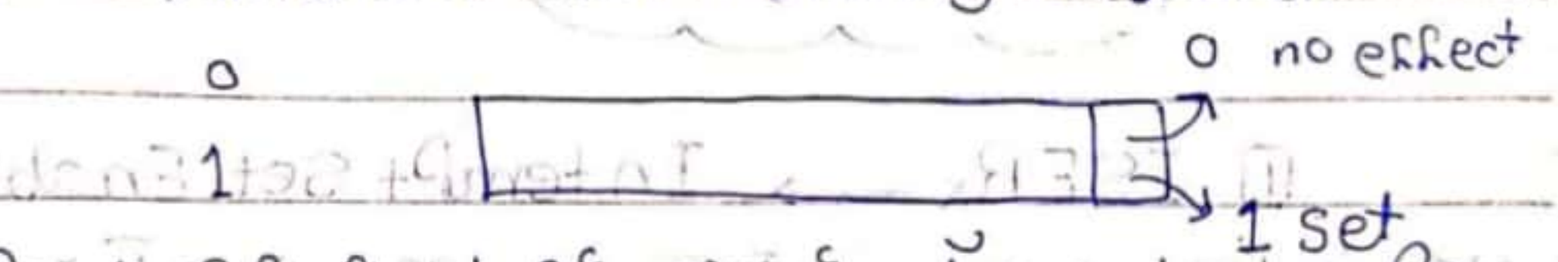
* هنروح بقى على ال memory map علشان نشوف ال offset بتاع ال register كام.

---

2. ICERx → interrupt clear Enable R..



3) ISPR → interrupt set Pending Register

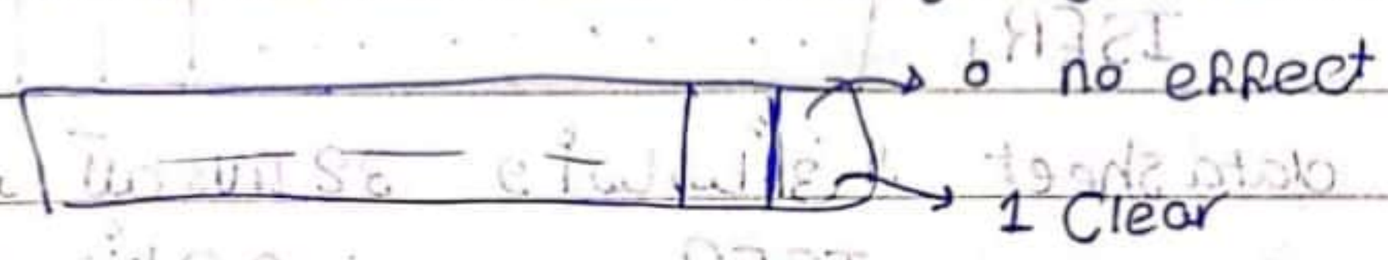


? Set to Pending flag ليه ممكن أحتاج أعمل
For testing or debugging

* يعني من غير ما يكون ال Peripheral شغال

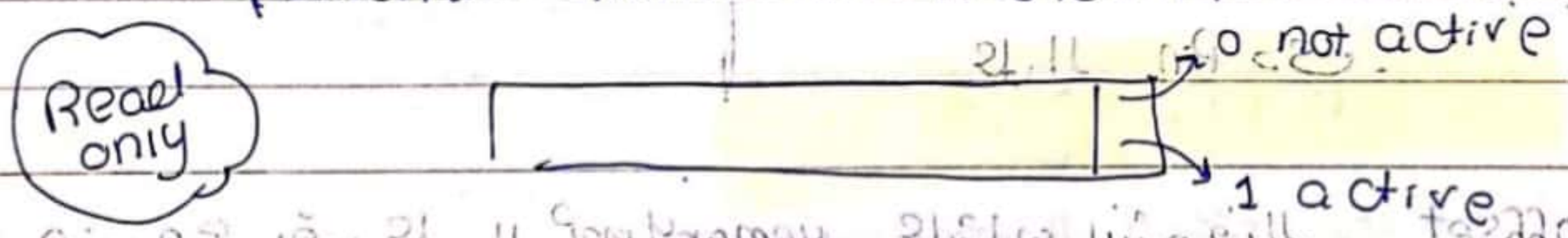
---

4) ICPR → interrupt Clear Pending Registe



---

5) IABR (Active) Read only
(Interrupt Active bit Register)



Lec 9

Eng: Assaf.

* Revision on NVIC

[1] Development

يعني بكتب كود وبعده بعمل

[2] Building

بعمل Generate من ال Source file ال hex file

أو على حسب ال Tool Chain بتاعتك.

[3] Programming / Burning / flashing / downloading

* وبعدين لو مطلعش الأوت ال أنا عايزه

فكدا الكود فيه bug بس هو صح

لكن مش بيتصرف صح

(logical error)

* فأبدأ أدور على ال bug علشان أحلها.

[4] De bugging → Cross review (طريقة)

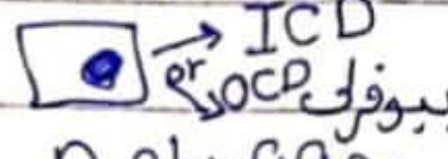
معناه إني بدور على ال bug فين

مملكة متنجحش فظهر

Debugger ← تساعدنا أكتر Tool

هي H.W & S.W Tool

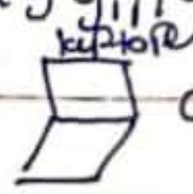
Processor support debugger



بيوفرلك Debugger

debugger H.W Tool

debugging Tool



open OCD ← Tool ودي بتركب على ال eclipse

اسمها (Plugen)

Debugger →

- Stop Processor @ any Point
- run line by line
- Read Register :
- Read/W Memory location

لأني دخلت جوا ال Processor

لازم تتعلم ال Skill دي.

لازم يكون ع الـ Procesor هيكون فيه دايرة صغيرة اسمها
اسمه ICD أو OCD

ICD : In Circuit debugger.
OCD : on Ship debugger.

هتحلين أعرف إتحكم في الـ Processor.

* berner sf مش شرط يكون بيه Support الـ ← debugger
لكن بتاعنا شغال

وقف ع إعادة lec 9
Eng: Ahmed Atia

debugging is the process of finding and resolving bugs (defects) or misbehaving logic in computer software or hardware, to make it behave as expected. Debugging tools (debuggers) help developers trace the execution of computer programs and monitor the internal state of hardware or software systems.

H.W & S.W Tools

debugger

Tool → debugging tool

H.W debugger Tool

Processor Support debugger

ICD

OCD

Stop Processor @ any point (Breakpoint)

Run line by line

Read Register :

Read/W Memory location

Processor will go Still

Lec 9

في ٣ طرق تبني بيها ال driver بتاعنا

1. #define
2. Struct
3. union

→ Priority :

↳ group + sub group

255 INT

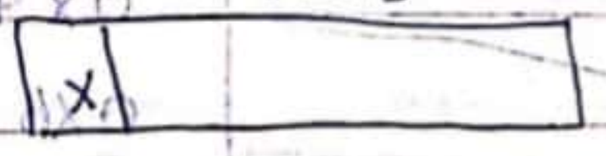
15 — internal | 240 — external

نفيد

128 | 128

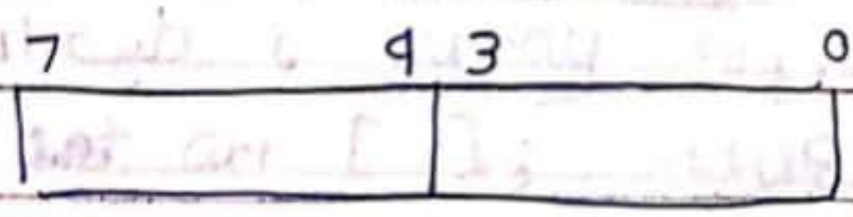
$2^7 = 128$ (7bit to access The group)

يعني محتاج register فيه 8bit هاخد منه 7



in general

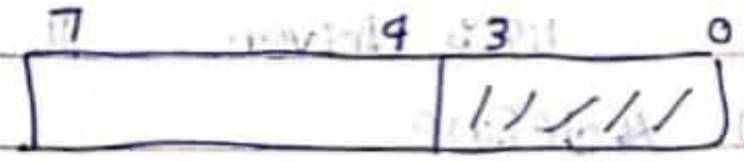
لكن ال STM قالت مش محتاجه كل دا ، أنا محتاجة عدد صغير جداً ال هما 4bit



4bits كفاية Software Priority لو عايز تعمل

for STM families : 4 bit have 8 Group

طيب كدا هناخد ال



LSb low و ال high

reserved هيبقى high والجزء ال MSb هناهيا خد ال

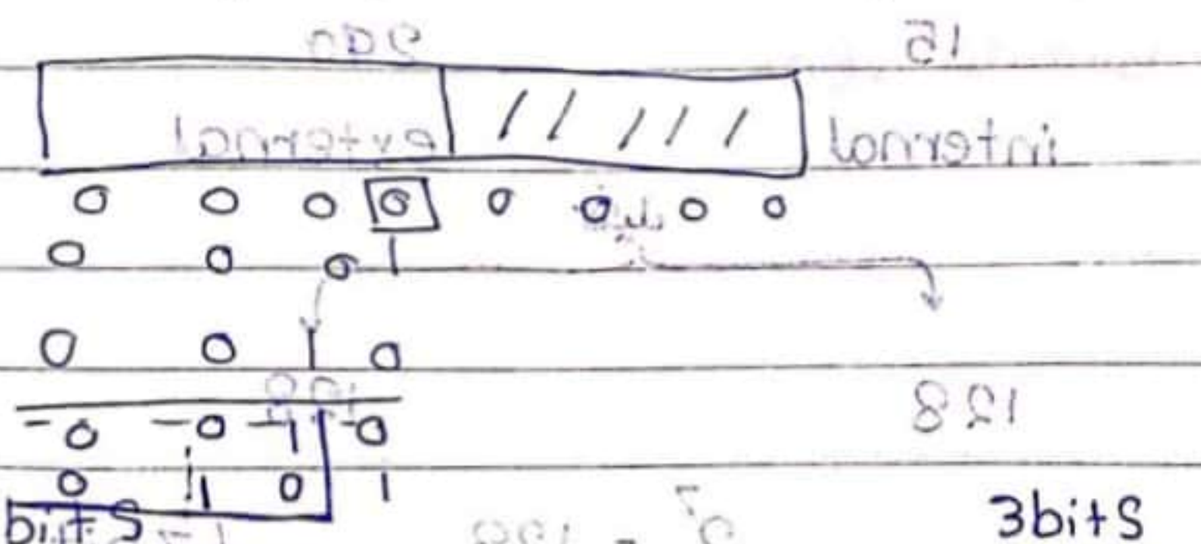
* ليه ؟

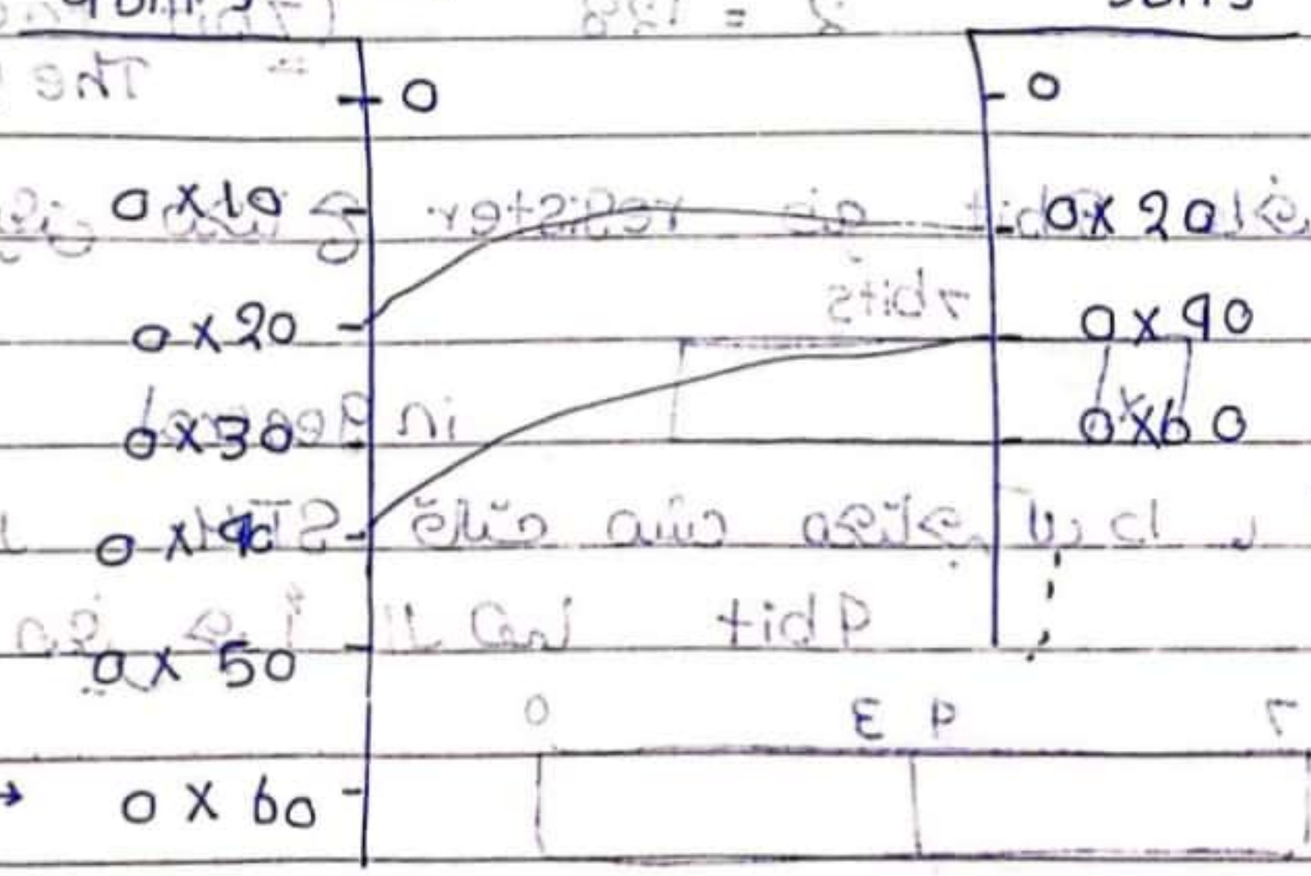
* قال إن في عائلات في ال STM

For STM families:

9bits or 3bits

بتاعنا





INT → 0x60

PORTing :

يعني بكتب driver لكل الناس (general) مش بس لل STM

* مش فاهمة :

STM → S.w.Priority → 9bit

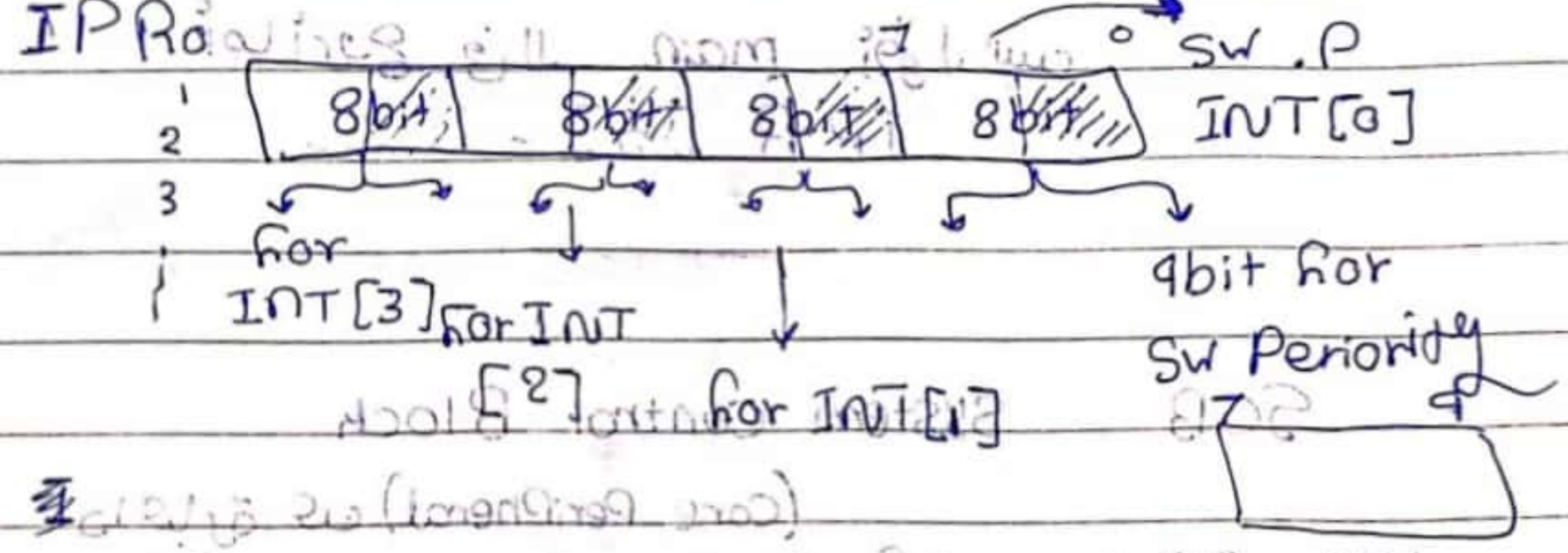
* عندنا في Programming manual

IPRx

* define دا بال register ازاي أتعامل مع ال

* مكتوب في ال dataSheet إن ال R دي byte accessible

ولأنه bits 32 ← sw.P



* define NVIC_IPR0 ((volatile u32 *)Baseaddress)

ومش هنحط (derherence) لأن لما أجي أعمل access

هعمل access على إنه array ، طيب ايه الفرق ؟

u8 ~~set~~ arr[ ]; u8 *Ptr = arr;

u8 arr[1] = ...

يعني أنا ممكن أعرف Pointer للأراي address

بيشاور على ال array دي

طيب عايز access

Ptr[5] = ...

= بكلمة : *(Ptr + 5) = ... 5 → offset

طيب ايه رأيك ن access ال address دا على إنه array
يبقى أنا كدا هخلى

IPR0 → element 0
IPR1 → 1
IPR2 → 2

index
↓
IPR[ ] =

يعنى من خلال ال Index دى هيروح لل IPR 0 أو 1 .. 2 ..
وهكذا.

```
#define NVIC_IPR ((volatile U32* ) Base address)
```

Pointer لل ((U8*))

أما أروح فى ال main أقول بس
IPR [5] = [ ]

SCB System control Block
دا عبارة عن (Core Peripheral)

	Register	5 بت	Password	A	
SCB_AIRCR 31	zero	Key	vectkey stat		16
15					0

3bits
↓
Priority دول هما اللى بيحطوها فى
Sub group ولا group ال = [ ]

علشان أكتب على ال R دا لازم أكتب 0x5FA
على ال vectkey.

EXTI

* هنعرف ازاى نعرف driver بال Struct
* لما عملنا الكود بتاع ال Struct فى EXTI_Config.h

when ! مش هنكمل dereference

```
#define EXTI ((volatile EXTI_t *) 0x40010400)
```

Pointer to Struct

EXTI

IMR		u32
EMR		u32
RSTR		u32
FTSR		u32
SWIER		u32
PR		u32

روحت بقى جبت Pointer وسميته EXTI
بيشاور على ال Struct

فخليت Pointer يشاور على أول واحد

0x400 10400
40 4
408
40C

Pins EXTI0 PA0 PB0 PC0

هيختار لى line مشتغل على A ولا B ولا C

EXTI1 PA1 PB1 PC1

هو عايز يعمل AFIO_EXTICR1 Register.

driver

EXTI 15

Eng. A. Assaf

today

① Review on NVIC.

② SYSTick (core Peripheral) many Projects.

wed

[1] Exam (MCQ)

[2] LED Matrix

Saturday

① Review EXTI

② OS Scheduler

---

[1] NVIC

↳ control external interrupt

- ↳ enable
- ↳ disable
- ↳ Set Pending
- ↳ CLR Pending
- ↳ Set Priority

Terminology:

[1] Development بكتب كود

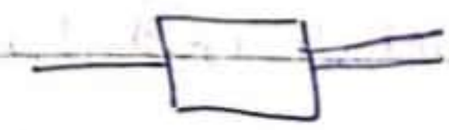
بعد ما بخلص كود بعمل [2] Building بعمل generat.

من الـ Source file الـ excutable file الـ .hex أو .exe

على حسب الـ Tool Chain بتاعك

[3] Programming / Burning / flashing / downloading

طيب دا ال MC ودا ال Tool محتاجين بقى حاجة
واصلة بينهم هو بقى اسمه



debugger

* أى debugger فى الدنيا بشغل بحرق الكود ما لكن مش
الكلس ما يعنى مش burner أى (flasher)
يقدر يـ debug
STLink دا بقى يقدر يـ debug احنا كنا نستخدم flasher

* معلومة هامة :
علشان أكتب أى external Interrupt

name of Peripheral - IRQ Handler
مثلا EXTI

(InterruptRequest)

لازم الشكل دا
ودا ال INT الهيجى لما ال EXTI يعمل INT...
وهى دى ال func المتروع لـ Vector فى مكان فى
مكان ال EXTI Table
والتسمية دى تحديداً اجاية من ال Linker Script
هنعرف وقت خروج ال bootloader

برنامج محاكاه → Not real Time as (delay)

(لتحقيق فكرة مش آلة) (software)

* Simulation ⇒ S.W Program runing on PC
↳ run on PC Processor

زي برو تس

عمرها ماهتكون حقيقيه ١٠٠٪ ؟ معدش بيستعمل و بيشتغل عليه (device)

* Emulator ⇒ Target + debugg (غالي جداً)

مش محتاج ال Micro controller ، لأنه جهاز فيه ال MC نفسه وال debugger بتاعه ، بيتوصلوا على ال PC و بعدين أعمل download للكود و بعدين debugg .

* Burner ⇒ flashing code on MC flash Memory

أفضل اختيار من حيث السعر

* debugger ⇒ flashing + debug on H.W MC needed

* في حالة ال Emulator :

ايه الميزة في كدا ؟

غالباً هو بيـ support more than Micro controller جواه

↳ هو بيكون غالي جداً

* بيجيبه وقت ما نبقى مش عارفين هنشتغل بأي Controller جديداً ،

أفضلهم هو ال Emulator لكن غالي جداً

* عايزين بقى نغير ال Periority

* Each Interrupt has Two Priorities

Group Priority — Sub Priority

(الاتجدد ال nesting لو حصلوا مع بعض في نفس الجروب)

* هنا ARM قال لل Interrupt عنده register باسمه Interrupt Priority وهو 8 bit لل Int ال external ال 8bit دول هيتخلفوا في ال Group & Sub Priority

8bit	low
?	//////// not used

الاحتمال بس 2 = 16 0000

رغم إن بعض ال MC يكونوا موجودين

Group — Sub

EX 1; →

Group (2 bit)	Sub (2bit)	
0 0	0 1	Group 0 Sub 1
1 1	0 0	Group 3 Sub 0

4 groups

Group (1 bit)	Sub (3bit)	
1	100	Group 1 Sub 4

2 groups

يكون في كل group 4 sub 0, 1, 2, 3

8 Sub

* في الأول لازم أسأل إيه تقسيمه ال bits كام Group وكام Sub

يعني كام group و كام sub

طيب سؤال ؟

لو أنا عايز أوقف ال nesting في ال NVIC أعمل إيه ؟

أحول كل ال bits رايحة على ال sub

دا معناه إن كلهم في نفس الجروب .

* هو ليه ST رجعوا وقالوا مش عايزين ال 9bit ال على اليمين ؟

لأن دا هيزود ال nesting كتير جداً .

← طيب دلوقتي أنا كـ software بقدر أتحكم في تقسيمة ال bits جوا كام للجروب وكام لل sub

ودا عن طريق

* Core PeriPhirals "SCB"

(System Control Block)

واحدة من وظايفه إنه يعمل تصنيف

* Define 9 bits of Priority classification

يعني تقسيمهم عاملين ازاي

في بقى Reg. فوا هيقولي التقسيمة ماشية ازاي

بس ال Register دا غريب جداً ليه ؟

لأن ليه مفتاح ، علشان أكتب عليه تديله المفتاح بتاعه

يعني Password وأنا بكتب عليه لازم أديله ال PW

فمن خطورة إن ال System ممكن يبوظ بسبب حالتين

متشابهتين زي 00 11 لكن معناهم مختلف

100 1 عملوا PW ليه

ال R دا SCB AIRCR 32bit

16 bit	low 16 bit
Password	Conf.gurcation العايزها

مش هتعرف تدخل في ال Config إلا لما تحط ال Password

SCB_AIRCR = 0x05FA (زي ما عايزة)

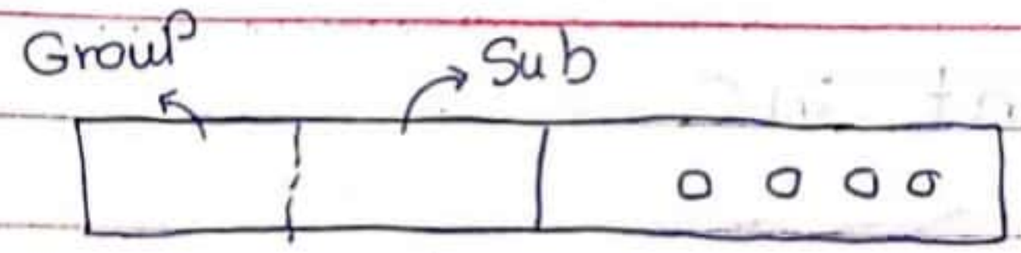
Password

هو عايز يأكد فعلاً إنت عايز تكتب.

جوا ال Register دول بقى ال محتاجهم 3bits : PRIGroup id
علشان أعرف التقسيمة

3bits (Write only) بتعملوه Software
reset و بتكتب zeros علشان منعملش reset
عليهم ، هيبقى وضعهم

* النص الفوق بتاع ال key
نبص لقى على 3bits بتوعنا



EXTI0 (6) — EXTI1 (7)

0100 0000 — Group1, Sub0

0011 0000 — Group0, Sub3

Counter / Timer

أي Counter في الدنيا عبارة عن register

Signal →

Register zeros bydefault

0	0	0	0

عرض السجل

Resolution

مع كل edge داخلة على الريجستر ما بيعد عدة و هكذا

ال input signal (أداة بتبقى حاجة من الاتنين

Time

(Timer) 1msec → 100msec ← 100 Count

synchronous) clock

Register

event (Counter)

(Asynchronous)

أي Timer في الدنيا هو بالأساس Counter بس هو بيعد على مراحل متساوية وتقدر تقول عدى Time قدايه

كل دوسة بكمل raising edge فبعد عدة

لو لفيت 100 فيه بيبقى دوست 100 مرة

لكن هل تقدر تقول خدوقت قدايه ، لا لأنه Asyn يعني ممكن يكون داس على مراحل مش ثابتة يعني بيعبر (الرقم الا فيه) عن count

Timer = Counter That haS a clock Source as its inPut (F_Timer)

$$T_{Timer} = \frac{1}{F_{Timer}}$$

@ any Point of Time,

* ConSumed Time = Register value * T Timer

* overflow → reaching a

* over flow Count = $\frac{\text{resolution}}{\text{?}}$

* $\boxed{\text{overflow Time}} = \boxed{\text{overflow Count}} * \boxed{\text{Timer}}$

$$= \frac{\text{resolution}}{\text{?}} * \frac{1}{F_{Timer}}$$

frequancy

• Processor بتاع ال Timer مش ال

SysTick

* Features :
  1) Core Peripheral (Standard)
  2) Memory mapped
  3) $\boxed{24}$ bit Resolution
  4) Timer only mode ، not counter

AHB/8 الداخله عليه

AHB

event مينفعش clock بتاعته طبعا input (يعني ال
Switch زي

  5) Count down Timer as 5 → 4 → 3 → 2 → 1 ...
  6) Provide underflow Interrupt @ 0

ISR
```
void SysTick_Handler(void)
{
}
```

Assume AHB clock = 8 MHz

SysTick = AHB/8 = 1 MHz

$\therefore T_{systick} = \frac{1}{1 * 10^6} = 1 usec$

```
Systick_load = 100;  // delay 100 usec
             = 1000;  // ... 1000 usec
             = 1000000;  //
```

Micro

and = 16000000

2^{24} = ...

ASS : Led Animation

open questions

[1] volatile key↔word

ex;

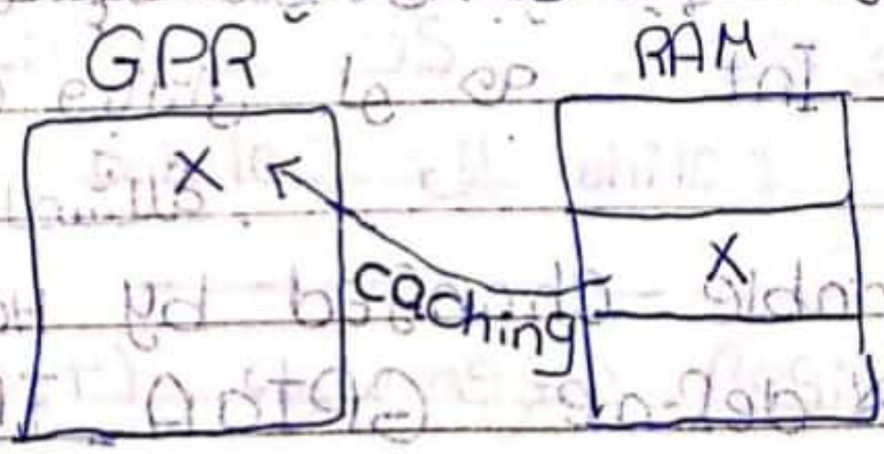
```
main
{
    u8 x = 10;
    while ( x = = 10)
    {
        ≡ no change in x
    }
}
```

[1] no change in The variable

→ Read x from Ram
→ compare x to 10
→ jump in while if true.

* فيه part في ال compiler اسمه optamizer
بيقول أنا هحاول أوفر شوية في اللود ال الراجل دا كاتبه
بدأ يبص على أول instruction الهي Read دي
ومش هيتغير وكمان هشوف ال Ram دا هياخد وقت
طيب ايه رأيك آخد نسخة من x وأحطها في GPR
يعني copy من x فيه

وكدا هعمل Read من ال GPR
الهيخلي يحصل كدا
الشروط دي

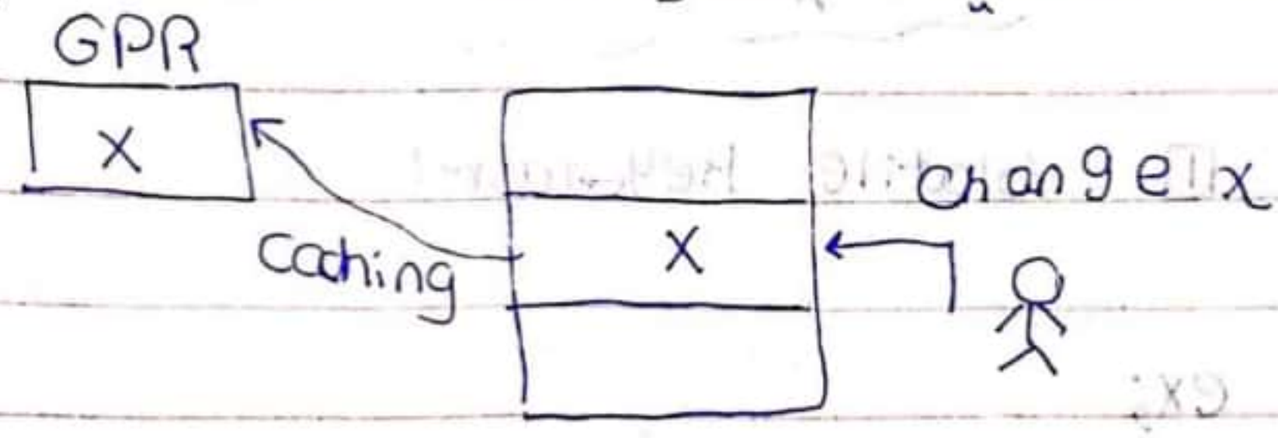


[1] No change in The variable.
[2] variable x is read many Times.

optamize دي الشروط علشان يحصل

* volatile

علشان يحصل caching لازم الشرطين دول يتحققوا



فيه عفريت جه يغير ال x الفي ال Ram فال optamizer
عمال يقارن بالنسخه القديمة الفي ال GPR
* فأنا خايف من الحاجة دي تحصل ..!
والكافرلت دا ليه احتمالات

* Three option for failing

1. Variable changes inside interrupt

```
ISR
{   x=50;
}
```

* معلومة:

linker بيمسح ال functions اللي الماشي معملها calling
طيب ISR هيمسحه ؟

لا ، هو بيتخيل إن محدش بيناديها وبالتالي x
مش هتتغير ، وبالتالي لو جه int هـ ال x هتتغير
وبالتالي هتحصل المشكلة.

2. Variable changed by Hardware

```
# define GPIOA_IDR *((u32*)(0x...))
```

↓ read only register

```
while (GPIO_IDR == 0xffff)
{
}
```

ال optimizer هل في حد ، بيكتب على IDR فمش هيلاقي

* Volatile

فهيعمل Caching فيقوم شايف القيمة الـ في IDR
ويحطها في الـ GPR وبعدين يقرأها منه.
لكن في الحقيقة في H.W من بره بيغير قيمته.
فهتحصل المشكلة.
علشان كدا بنعمل (Volatile) للـ (registers) لأنها
بتمنع الـ Caching وبتروح تجيب قيمته من
الـ address الأصلي في الـ RAM

* في طيب لو الـ variable هو readonly مش غمره
ما هي Caching ما علشان ملغبطش نفس زحط
Volatile.

③ Multi tasking

```
while(1)                  while(1)
{                         {
  (1)  متغير  ≡ Check       (2)  ≡ Change
}                         }
```

لكن في في الحقيقة الـ operating system
في عالم الـ OS بقدر يخلينا نتنقل من
while 1 إلى while 2 ويحصل Change

أي Register هنعرفه سواء (r) أو (w/r) هنحط
Volatile

قاعدة

* معلومة Cash Memory ممكن تكون موجودة في ال Processor وممكن لا ، هناخدها تاني. Prob. Qs

---

[2] Function Call (vs) SWI (Software Interrupt)

```
void func (void)
{
  /* code */
}

main
{
  func();
or
  asm(" SWI ");
}
```

assembly instruction in C

```
ISR_SWI
{
  /* code */
}
```

اللوديد هيتفذوا :

* لحد الآن مفيش فرق

طيب امتى استخدم دى وامتى دى ؟

function

① can be normally interrupted unless Critical section is used

function call ممكن تعمل Int وجوه.

SWI

① May not interrupted if nesting is not allowed.

function SWI

مهم جداً جداً

② Have different Context Switching

هو ايه دا ؟

هو عملية نقل من Context معين ل Context تاني

مثلا لو بتنفذ Code و هتـ Jump للكود تاني ،

بتعمل Context switch و قبلها بتعمل Save

لشوية معلومات ، زي مثلا المكان الهرجعله

أو شوية registers و هكذا

* هنا بقى المعلومات ال Saved في حالة ال function

مختلفة كبيرة عن المعلومات ال Saved لما يحصل SWI

بتاع ISR, Context switching في included

ex: Register PSW (flags)

with function (not) included in function

C.S

PC = 500

0x500

Code

0x100

main

PC = 101

① ② ③

مبقاش

Save بـ 101

علشان أرجعله

في ال func

لكن لو هيـ Save طبيعي

وهيـ Save ، وكمان أ Save ألدانا الفي GPR علشان

لما أرجع استخدمها

ال PSW مش بـ Save في حالة ال function

وهو بيعبر عن آخر سطر موجود هنا ③ لأن func متوقف

، لكن في حالة ال SWI بتعمل Save لـ PSW

وترجعه بالمكان القديم ، لأن دايما بيفترض

بأن ال INT حصل على غفلة ، وغالباً كل ال Rs بتعملها

Save.

بار بيتكتب عليه assembly

ليه لأن ال Control.R متعرفش تعمله access
غير لو إنت في ال Privileged مود ، إيه
لو ال INT ، تقدر تعمل كدا
لأن كل ال INT بتتم في ال Privileged Mode

All Interrupts run in Privileged Mode

ودا أهم فرق في ال ARM
لأن ال function بتتم في ال Privileged Mode.

* Non Privi (مش Some register) To Security

لو إنت جوا SWI تقدر تأكسز على أي حاجة إنت عايزها

ال Interrupt مش بتتعمل من ال API ، لكن بتعمل generate
instruction بتعمل jump عليه وبتنفذه

SysTick

↳ Control → clock = AHB or AHB/8
→ Enable to systick
→ Enable interrupt
→ flag

بترفع لما ال Int يحصل . وبيه CIR لما نعمله Read

↳ load → بنحط فيه القيمة اللي عايز عدها (مش بتتغير)
→ 29 bit

↳ value → القيمة الحاليه

ex; load 1000 [1000] value

مع ال Pulse ال بعدها يبقى 999 وهكذا
لكن load.R بيبقى فيه القيمة الأولى ومش بتتغير
غير بال Software

API (Application Public Interface)

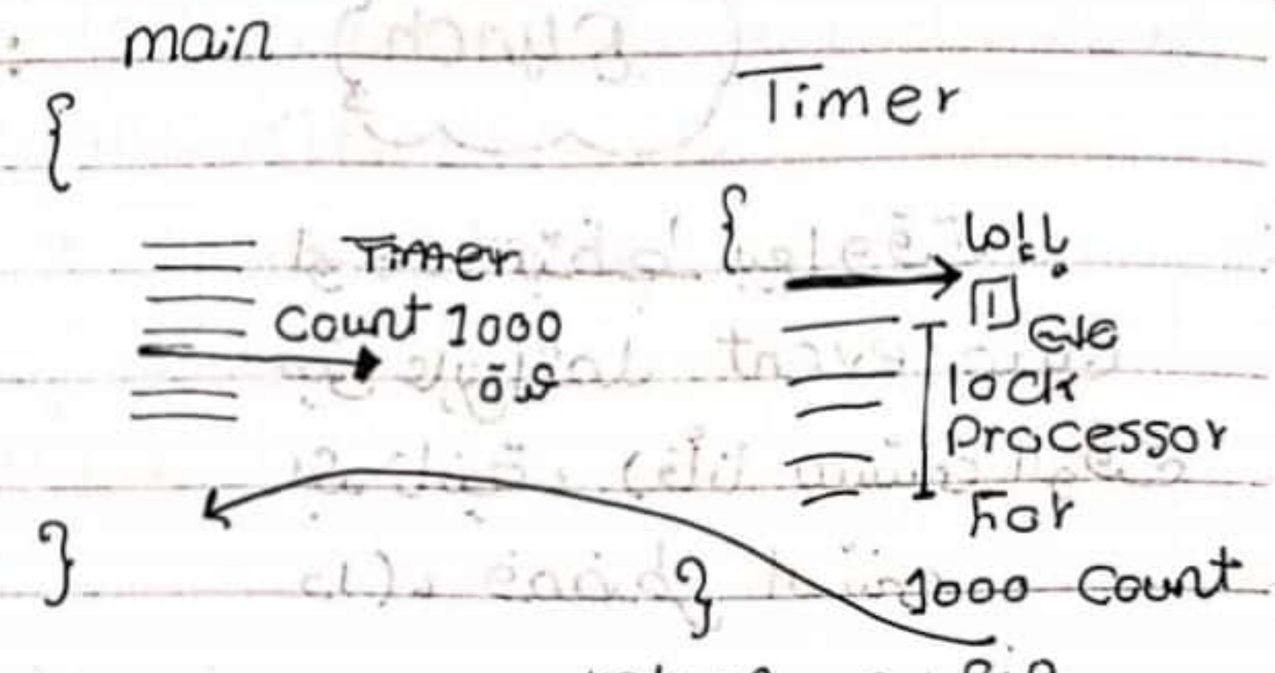
في علم ال design أي function بنقدمها للناس بنسميها API .
ممكن يقدم 10 API أو أي عدد .
* وقدام هنا خد ال Private .

API : (ال هيقدمها ال SysTic)

1. MSTK_voidInit ⟹ Choose Clock (AHB or AHB/8)

2. من أهم مميزات أي Timer إنه يقدر يعدلي ال Time بطريقتين مثلا 1000 عدة ، في حد من الاثنين ياما أبدأ
① أعد 1000 عدة وأفضل في ال function لحد ما 1000 عدة يخلصوا وأعمل return ، أو لما 1000 عدة يخلصوا يعمل Interrupt

2



ونجدين return

، أو Timer

هيعمل return ⟷ على طول وبكدا مكملش lock لل Process

وبيرجع لل main

تكمل ال كانت بتعمله ، وهيعد ال 1000 عدة بتوعه ولما يخلص

هيعمل Interrupt ←

* أبسط الأفضل ، يفضل متعلق في ال function ولا ال main

تكمل وهو بيعد مع نفسه ولما يخلص يعمل Interrupt

* Synchronous function :

يعني لما يطلب منها

حاجة بتعلق ال Processor لحد الحاجة ال انطلبت تتعمل

ومبتعملش return غير لما تخلص . دا اسمها

Asynchronous function (only start)

لما بطلب منها حاجة بتعمل return وتقول روح كمل ال بتعمله

ولما هي تخلص هتعمل return وتديك notification

بكدا.

Synch	:	Asynch
لو عاوز بتظبط بيها وقت معين عايز أعمل event بعد ثانية، (فأنا مستنى الوقت دا)، فمضطر استنى		مش بتعلق ال Processor فلو أنا بشتغل ومش عايز أوقف حاجة تانية، يبقى دى أفضل

(2) MSTK_ void SetBusy wait (..) → Sync

↓

هتعلق في waiting state وشاغلك ال Processor (Synch)

هنا هحط القيمة في load وأقعد في while لحد ما ال flag يترفع و أعمل CIR

شبه ال delay بس دى مفيها Timer

(3) MSTK_ void Set Interval Single(....)

(4) MSTK_ void Set Interval Periodic(....)

→ ~~Asyn~~ Async

معناه هديله Time يعده مرة واحدة بس

هديله Time يعده كذا مرة يعنى أما يخلص 1000 مث يعمل INT وبعدين يعد تانى 1000 عدة و INT وهكذا

(5) MSTK_ void StopTimer(....) بتوقف ال Timer

Asyn وده بس في حالة ال

[6] MSTK_ ~~void~~ u32 GetElapsedtime(--). → load - value عديت قبليه

[7] MSTK_ ~~void~~ u32 GetRemainingtime(.....).
↳ value register.

Pointer To function :

```
void func I(void);
```

Pointer: ↓ ↓ ↓

```
void (*Ptr) (void);
```

↓

اسم الـ Pointer الهيشاور على func بتاخد void

[1] كدا بعرف الـ Pointer هيشاور على function ايه

[2] عايز بقى أخليه يشاور على الـ function وترجع void
الـ اسمها func دى :

* اسم الـ function هوال address بتاعها

[3]

[4]

```
void (*Ptr) void = func;
```

لو عملت كدا (Ptr() = func() مفلوقة

بيتعمل في الـ SysTick Program بس مش في الـ main .c

ISRs

→ ① Hardware Circuit

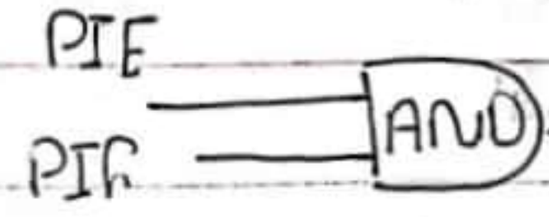
علشان أشغل Peripheral لازم أعمل :

① PIE

② PIF (P.I flag)

لازم يترفع

الاتنين مع بعض داخلين على And Gate



* لحد هنا لسه ال interrupt محصلش

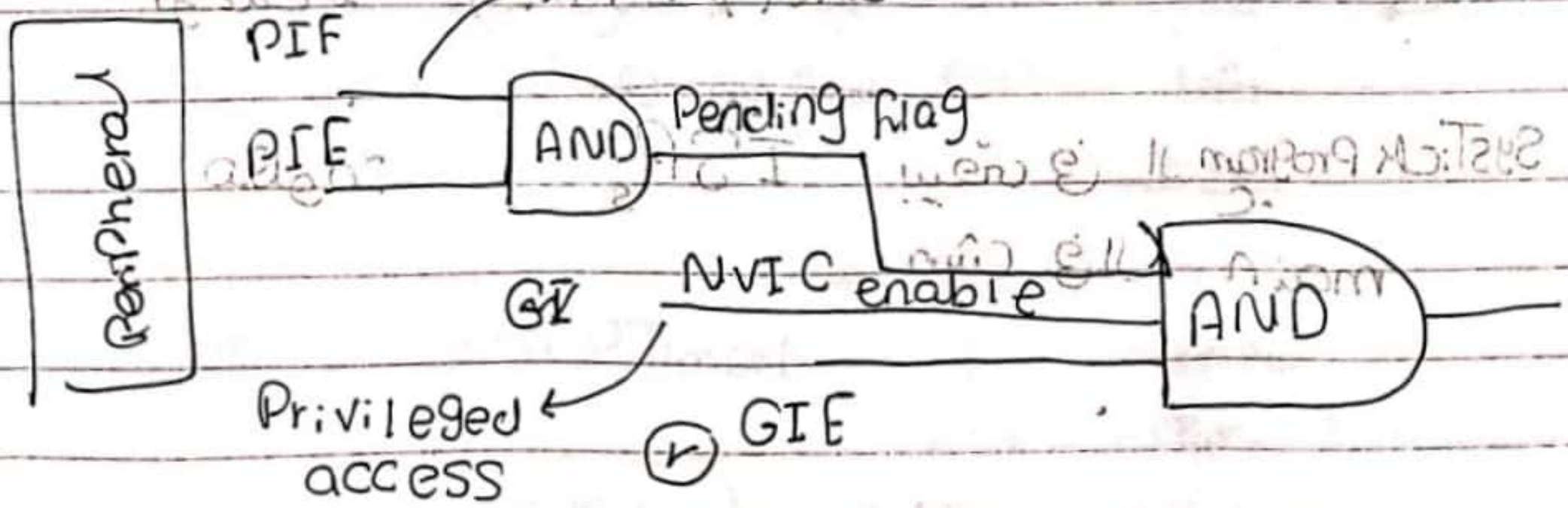
لازم شرط كمان أساسى

③ NVIC Peripheral enable لازم دى كمان

④ GIE ولازم دى كمان

Global I. Enable

unPerivileged access



لازم دول يحصلوا علشان ال INT يحصل

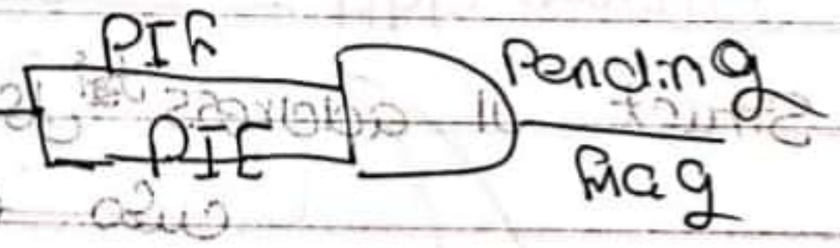
* Global interrupt enabled by default (by H.W)

255 INT

15 internal	240 external
"Control"	"Manage" (تحكم كامل)
driver request to the processor	① enable | disable
dependent on Priority	② Priority mangement

* Priority is configured SCB (core peripheral) (System Control Block)

* every external interrupts:
have ① Pending flag
② Active flag



ISR{ → Pending flag clear
Active flag set (ISR أول ما أدخل)
} → Active flag clear (ISR بخلص)

* Internal interrupt — SCB → Priority
* external interrupt — NVIC → Priority

* higher SW Priority will be excuted first

لو حصل إن 2 ليهم نفس الـ Priority

* if S/W Priorities are Same → higher H. W Priority هتبقى

Drivers

* NVIC
* SYSTick
* DIO
* RCC
* STD_Types
* BIT_Math

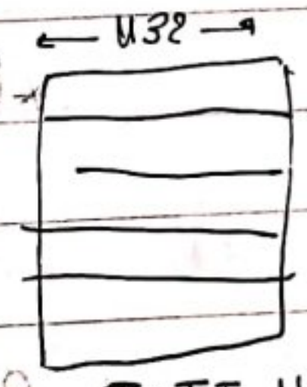
Pointer to Struct

define volatile الأدرس

EXTI (EXTI_t *) 0x400.....

اسم ال struct

EXTI → 0x400



مش هنعمل dereferance (*)

آكتر بعمل → ar0 احنا هنعمل dereferance EXTI_t

* وبكدا ال EXTI بقى بيشاور على address ال struct
كله طيب عايز أكتب على element معين
فهنستخدم بقى → ال هو اسمه ar0
EXTI → PR

هام

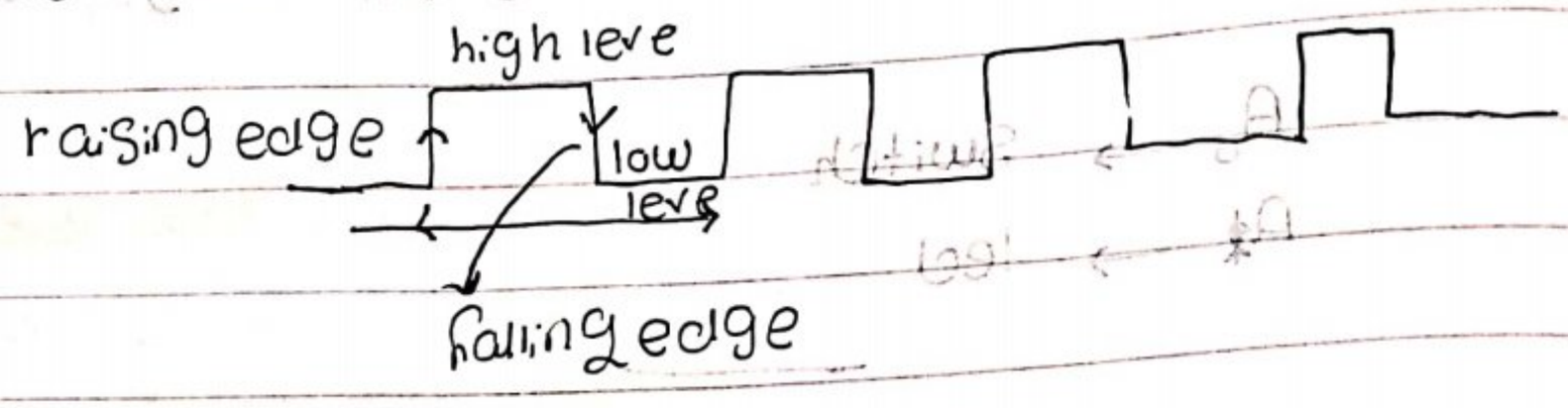
وبكدا في طريقتين بنقدر نأكتب بيهم الـ struct

1 * (EXTI.PR) = 0 ✓

2 EXTI → PR = 0 ✓

Int sf

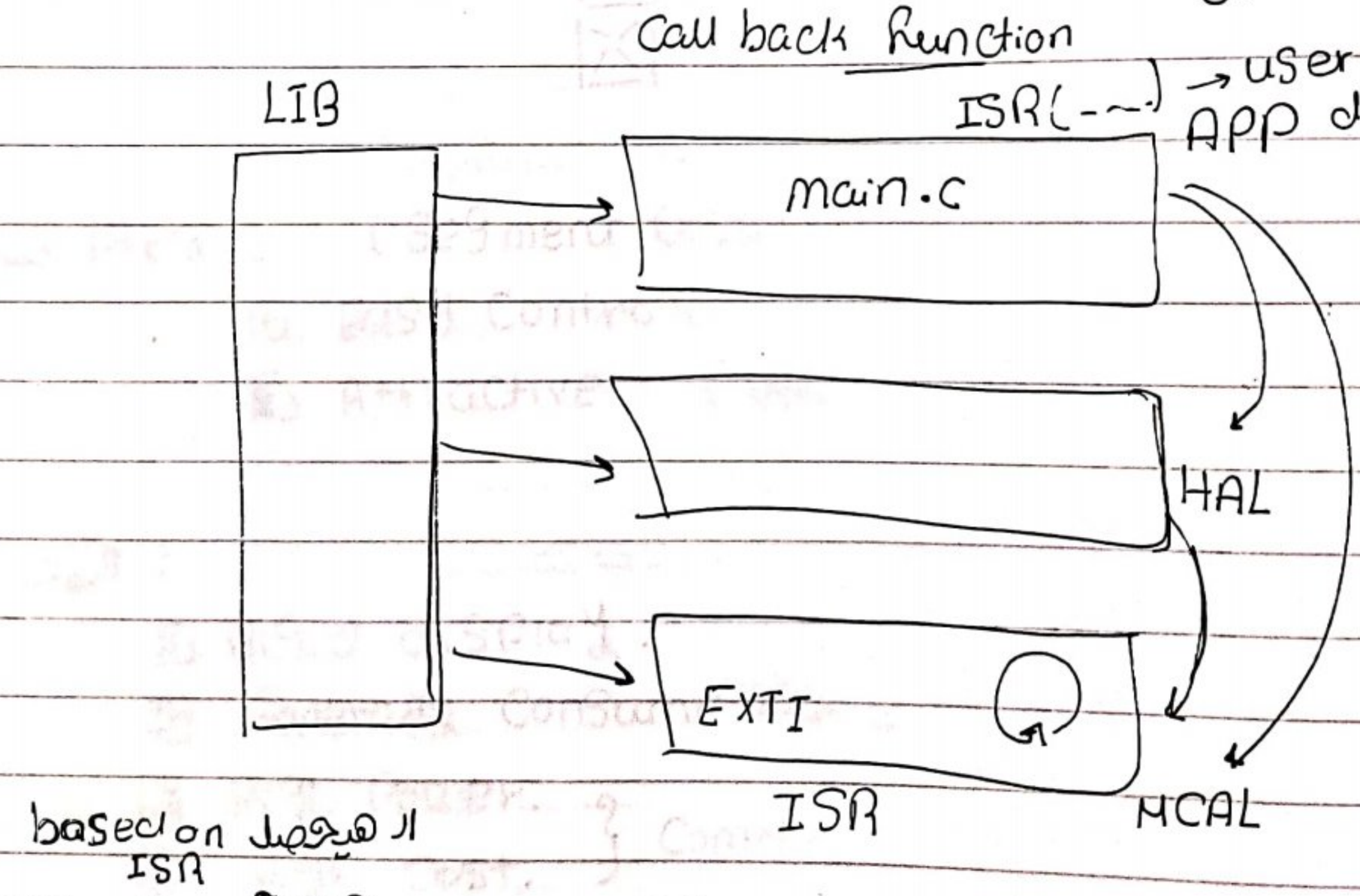
* interrupt → sense mode



Modes:

1. Rising edge
2. falling edge
3. (IOC) raising or falling

لو أنا شغال raising وعايز أغير ل falling
، مش هيدفع بطريقة ال define *
علشان كدا عملنا function اسمها SignalLatch وظيفتها
إنها تغير دا.



يعني لما يحصل ISR ال بيد design ال جوا ISR هوال user
و في ال APP layer تلك ال ISR في ال EXTI في
ال MCAL و مستحيل ال MCAL تنادى أو تعمل
include لل APP

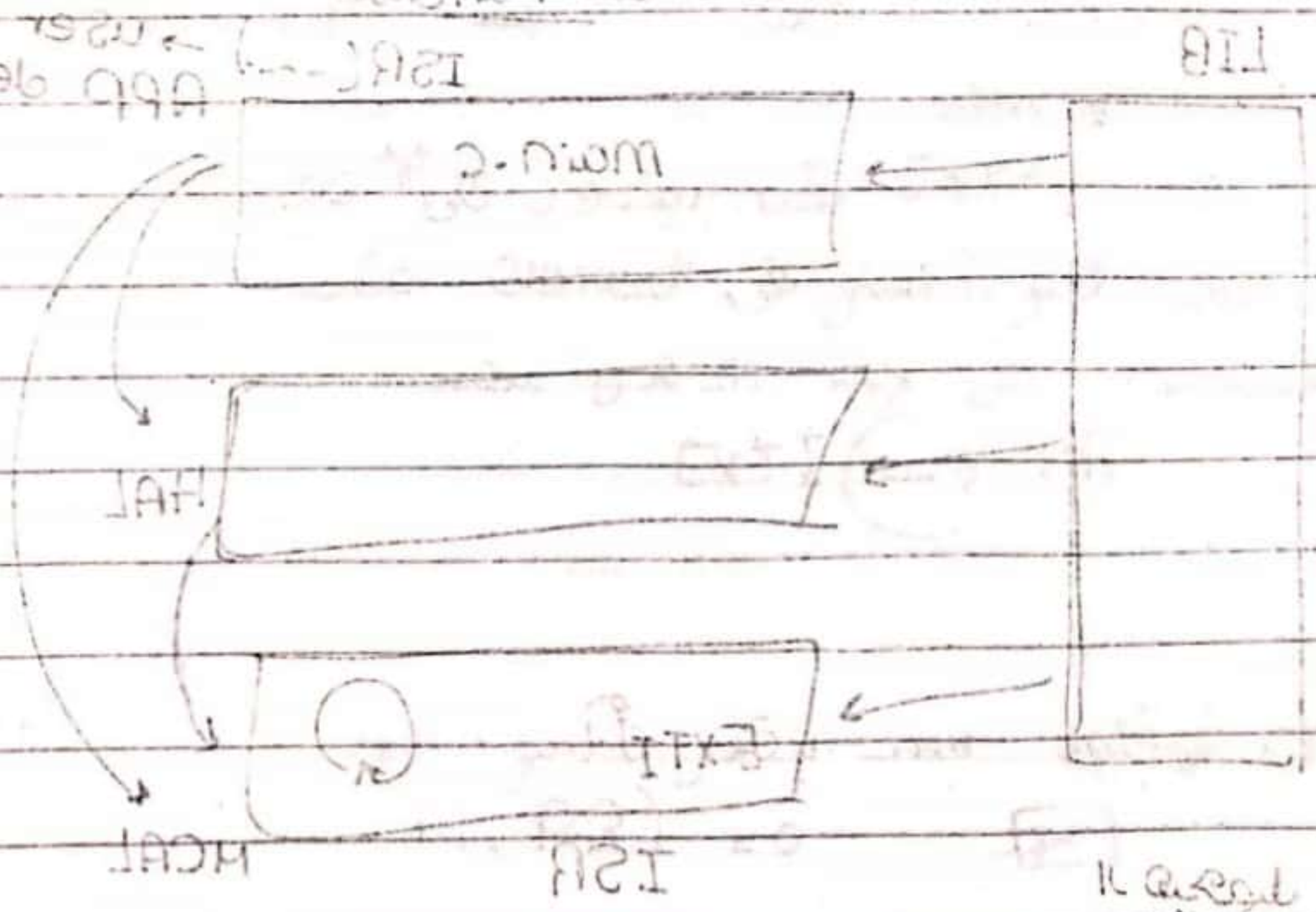
طيب ايه الحل ؟

الحل هو Call back Function

$A_0 \rightarrow$ switch

$A_1 \rightarrow$ led

end L12 → التوصيل



Lec 13

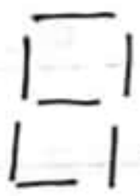
Eng_Ahmed Assaf

Displays : أغلب المشاريع بتحتاج حاجة أعرضها

1. **Segment Display**

كل Segment عبارة عن led عريضة شويه مرصوصه بشكل معين

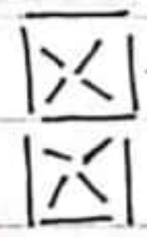
↳ **7 Segment Display** أشهر أنواعها



→ leds to numbers

↳ **Alphanumeric Display**

زي الـ 7S كدا لكن بيكون فيها Segment أكتر علشان تعرض حروف وأرقام



→ leds

مميزات Pres : (Segment Display)

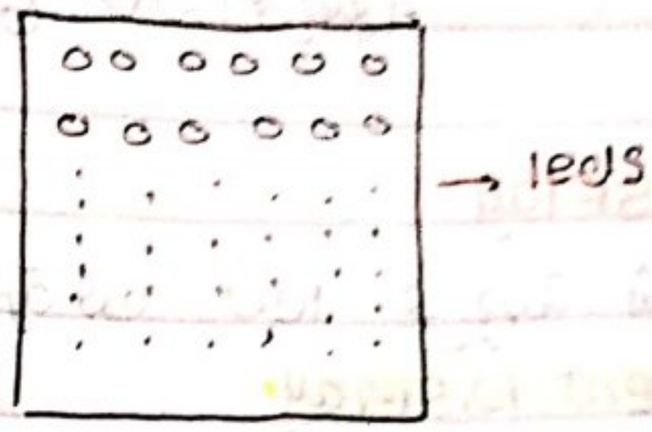
1. Easy Control
2. Attractive (للرؤية علشان ألوانها)

عيوب :

1. Rigid display. مقيد للشكل (x * y)
2. Generally consume many (I/O) Pins.
3. High Power.
4. High Cost.

} Comparing with other types

2 LED Dot Matrix or Dot LED Matrix



Pros :-

1. very Attractive بتستخدم بشكل كبير في الاعلان
2. flexible

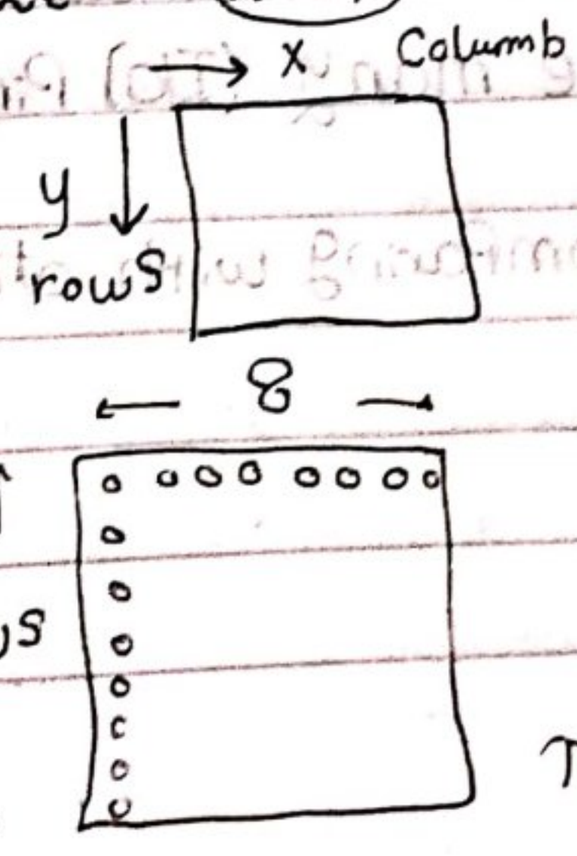
Cons:

1. Little bit harder Control.
2. High Power
3. Generally Consume big (I/O) number

LED Matrix Display :

1. specifications: مواصفات

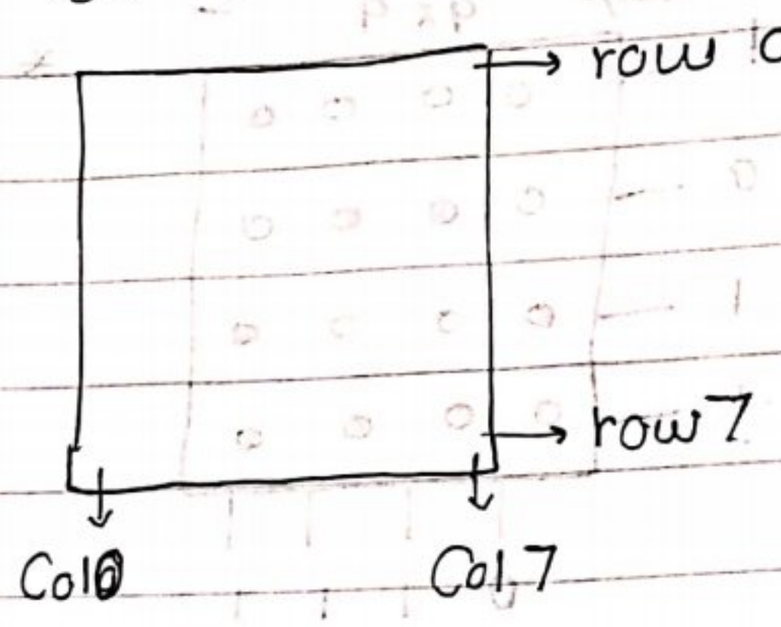
↳ Size (X * Y)



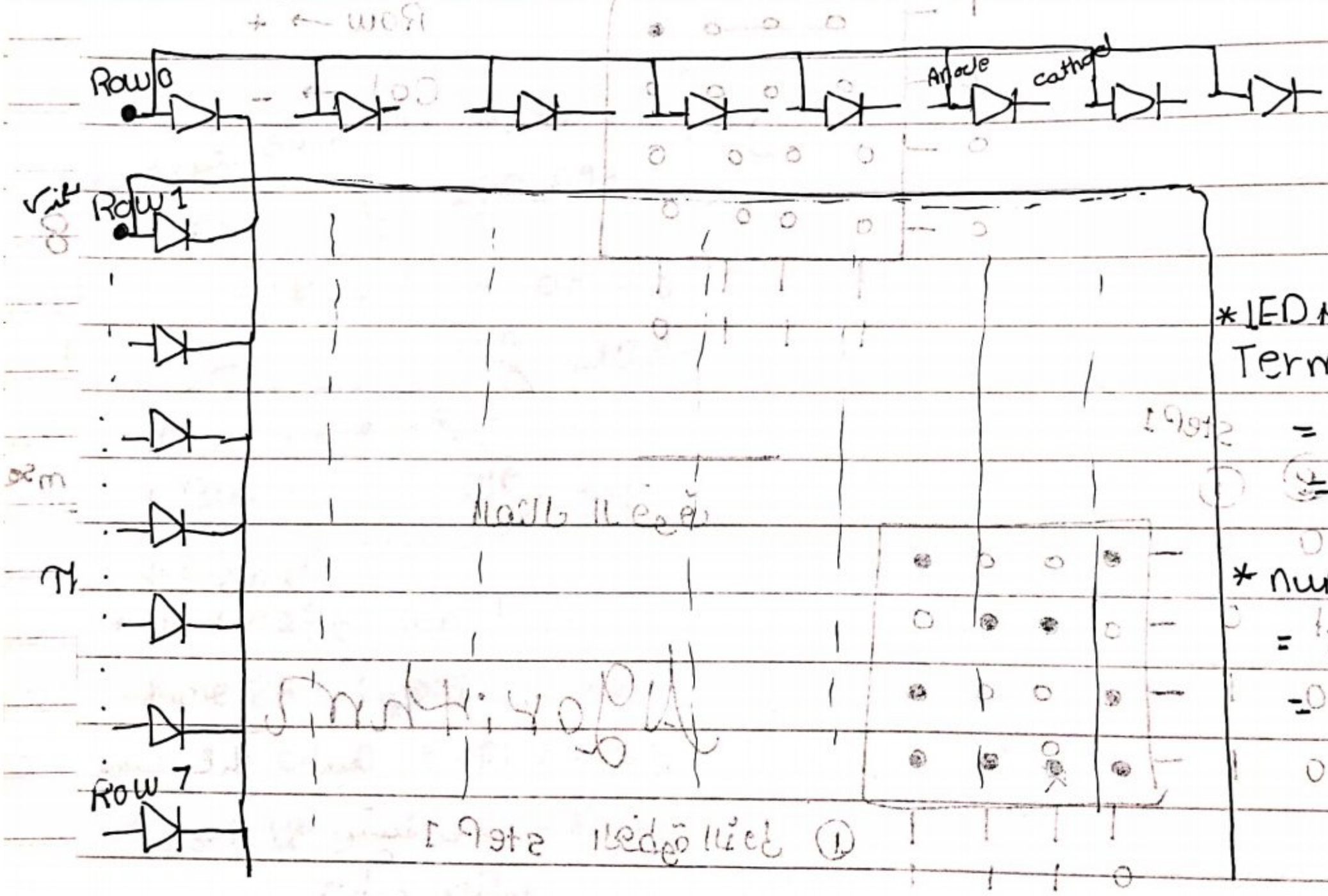
الموجود 8*8 led Matrix

Total number of led
= 8 * 8 = 64 leds

علشان أحسب ال Power ، اعرف بتاع الليد الواحدة .



LED Matrix Connection



* LED Matrix Terminals = x + y = 16

* number = x * y = 64

وهنا ممكن العكس في التوصيل آموجب وسالب

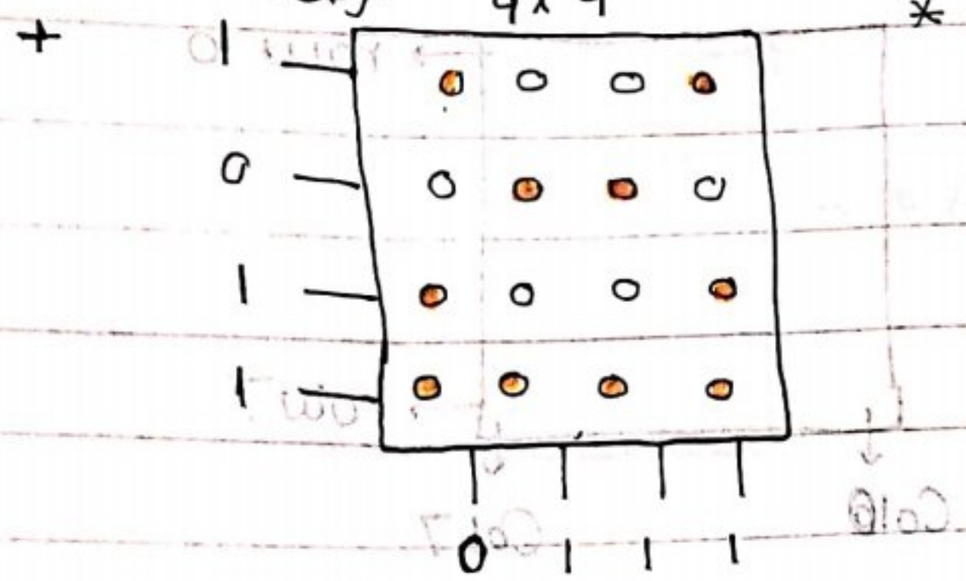
ROW → +

Col → −

بتاعنا دلوقتي لكن ممكن بقا يلك غير كدا في السوق

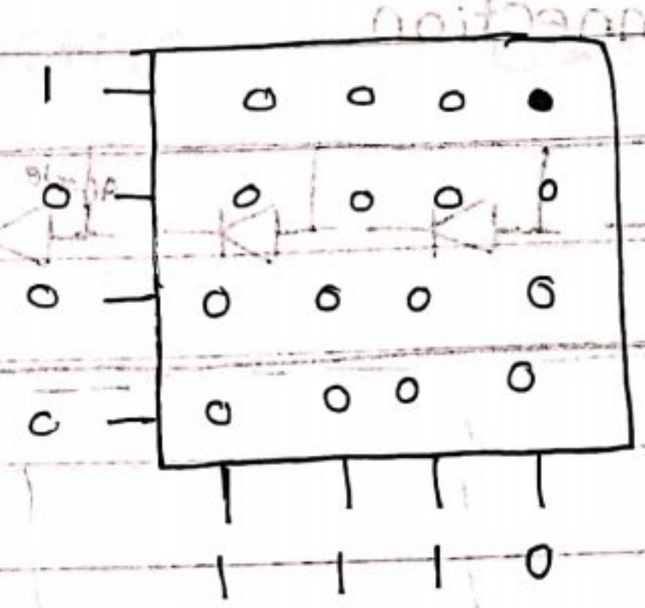
* Concept of operation

"فرضاً" 4x4



* operation Mode
  - ↳ column control
  - ↳ Row control

active column at Time

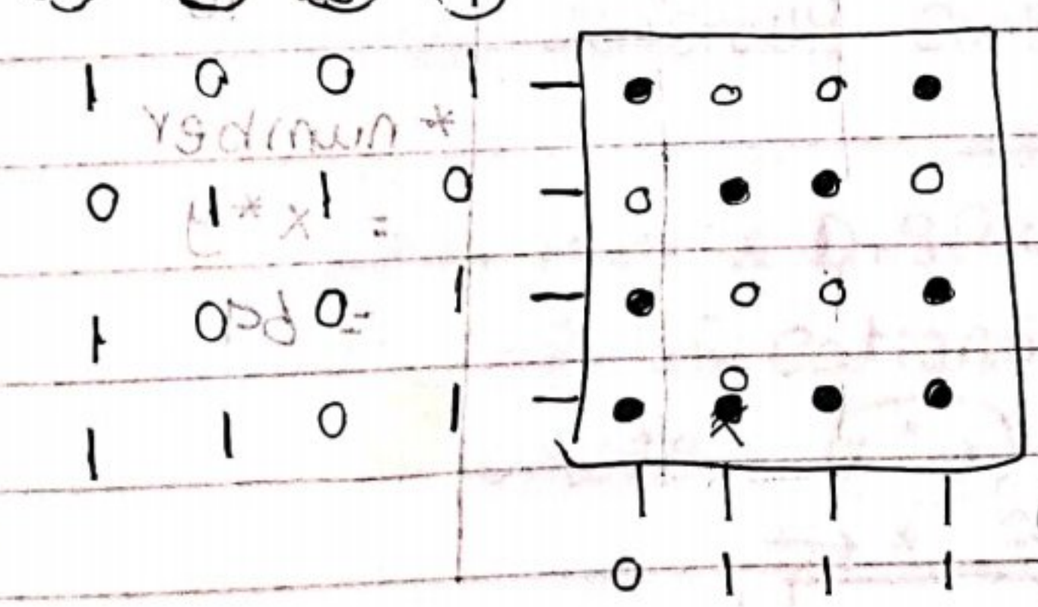


Row → +

Col → -

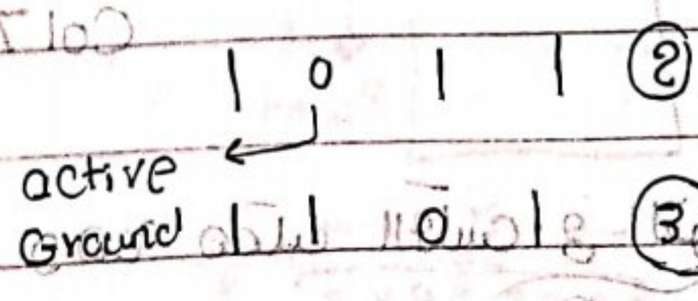
المثال الفوق

step 1



Algorithm

① الخطوة الأولى step 1



بس كدا column واحد بس الاشغال at time
وكدا مش هشوف الشكل مرة واحدة. !!
الحقيقة إنك هتشوف داكلر رسمة واحدة عن طريق
concept عظيم جداً ، اسمه

Presistance of vision — كسور الرؤية

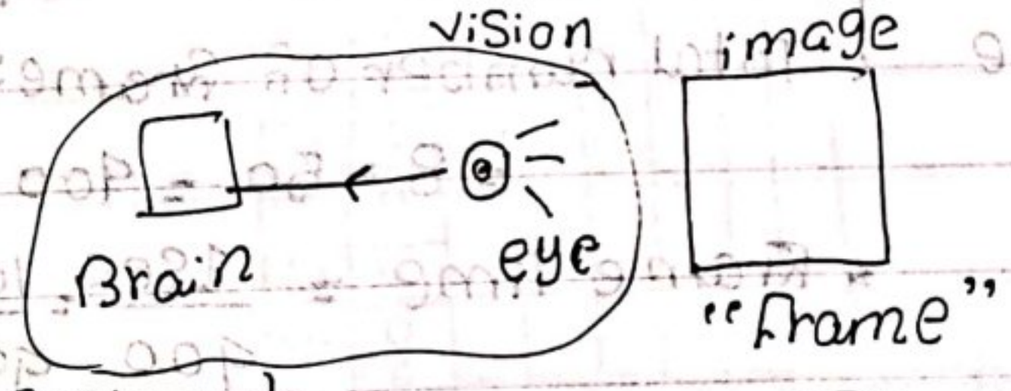
POV

ودى خدعة

* علشان نشوف الصورة مرة واحدة.

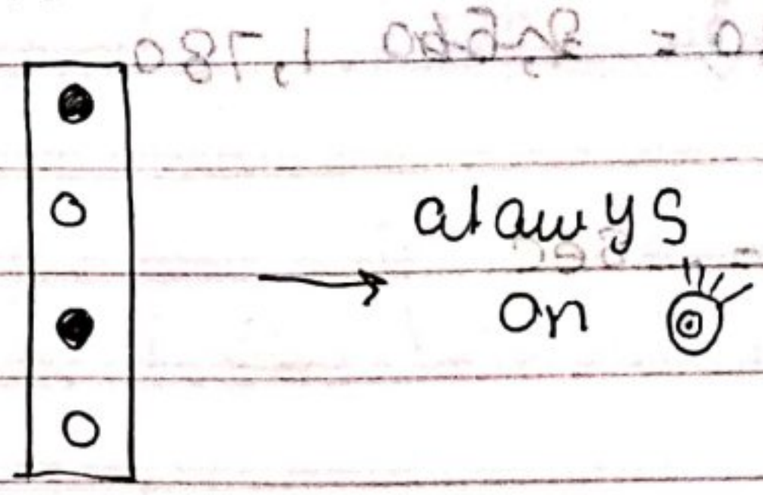
POV concept ← مرعبة جداً والقصة بتاعة الفيلم الباث مخدر تالما
← وهى معرضة دولياً

Presistance of vision (PoV)



This system ↓
* Average limit of (25) FPS (frame Per Second)

يعنى العين والمخ بيقدروا يعالجوا 25 frame
فى الثانية ، لو اتعرض أكتر من 25FPS هيركب كله فى بعض
فلو احنا اشتغلنا على 50 FPS هش هنشوف فيها صور ورا
بعض لا ، هنشوفها زى الفيديو كدا بشكل مركب.



↓ on → 50 frame in second
off → 50 frame in second

طيب ليه شوفت ال on وال off لأ مع انهم متساويين
لأن العين ~~resistive~~ resebtors للضوء أكتر

Algorithm

Enable column 0
* Set rows value
delay (2.5 ms)
Enable Column 1
Set rows value
delay (2.5 ms)
⋮
enable Column 7
Set rows value

"delay"

→ number of Columns = 8
→ FPS = 50
Total number of frames
= 8 * 50 = 400

* Frame Time = $\frac{1 sec}{400} = \frac{1000 ms}{400}$
= 2.5 ms

(ex) 16×16 led Matrix
80 Hz → delay ...?
16
to 8/2 * 80 = ~~2,560~~ 1,780

delay ∴ $\frac{1}{1,780}$ sec

CRT TV Hz 110

normal range 50 : 60 Hz POV

lower

هيسبب مشكلة

Flickering

هزة بترعش

higher

Ghosting → شبح

هتلاقي المفروض ليات

المفروض متنورش هتنور

خفيف أوى

*** _____ ***

Step 1

	C0	C1	C2	C3	C4	C5	C6	C7
R0	0							
R1	0							
R2	0							
R3	0							
R4	0							
R5	0							
R7	0	1	1	1	1	1	1	1

*array 8byte

8byte

arr[ ] = 0 129 18 18 129 0

0 129 18 18 129 0 0

Row 0

arr[0] 0 0 0 0 0 0 0 0

[2] Disable Col (function DisableAllCol)

أفضل من كتابة الـ 8 كل مرة مرة

CamScanner

Infra Red Communication
IR

هو Communication Protocal وهو سهل جداً
هنقدر نـ Commu بيه أي حاجه على ال kit بتاعنا
وإنه رخيص جداً.

IR
↳ light can not be detected by human eye

decrease ← | visible light | → increase
IR

لو أقل بيعمل حاجه
gamma ray | X-ray | UV
اسمها
ultra vailote

مميزات مقدرش أشوفه
① فلو عمل بيه commu مش هشوفه زي لمبه الريموت، فدي ميزة مش هشوفه.
② safe to humans (IR)
مش زي مثلا ال X-ray لو جت على الإنسان ممكن تأذيه بالذات مع الوقت

* أساساً الأضواء الطبيعية زي الشمس فيه جزء منها IR
③ سرعته هي سرعة الضوء $3*10^8$ m/s (أسرع حاجة)
يعني دي ميزة قويه جداً
④ Relatively long distance
up to 10 meter بتتكلم بالأمتار
ودا كله.

⑤ very low cost
وهو أرخص Communication Protocal

لعبة البيت بتطلع IR . وهى Safe .

أكبر Challenge هتقابلنا

IR challenge :

سؤال هل ال IR الطالعة من remote control
هتضايق من ال IR الموجودة بردو ؟ زى مثلا لعبة
أيوا أكيد هيدخل فيه و المستقبل مش هيبقى عارف
دا من ايه ودا من ايه ، أكبر challenge بقى بتقابلنا

↳ Interference with The ambient light

ازاى أخلى ال IR الطالع من الريموت مثلاً يبقى
unique ومتأثرش بال IR الحوالية ؟

* دا إنى أبعت ال IR بفركوانسى freq مش موجود
فال ambient light
يعنى أى light فى الدنيا بيتحرك ب freq وهو عبارة
عن electric magnetic wave

light freq

أى ضوء

وهخلى ال receiver يستقبل ال freq دا بس

أشهر ال freqs هى 38 KHz

freq

يعنى هعمل IR بتتحرك ب 38KHz وهعمل

Rec.

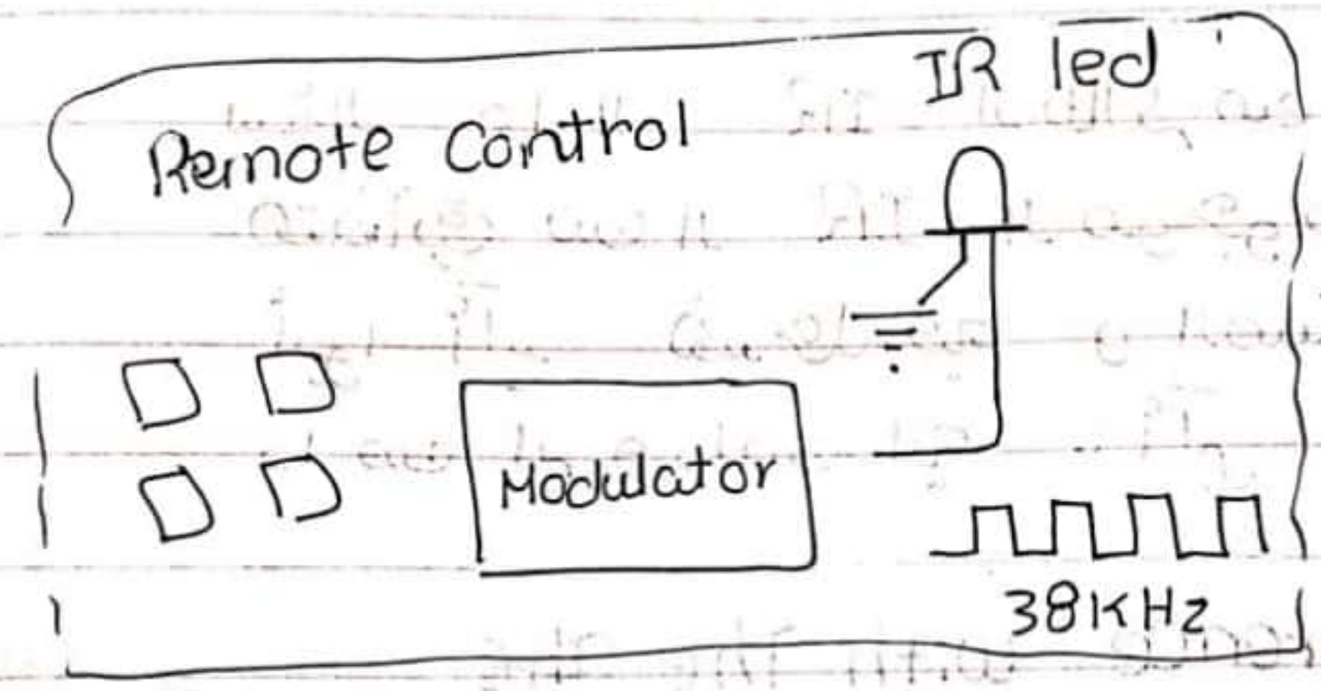
بس 38Hz يستقبل ال 38H بس

مفيش، ضوء الشمس مش بيطلع ال freq 38 دا لأنه نادر جداً ، فأي حد تاني بيبعت، ال reciver مش هيفهمها.

InfraRed System :

بيتكون من led عادية بس بيطلع IR

طرف على Gnd وتاني على modulator



ودا ميكروكنترولر دوره يطلع Signal على ال IR يخليها تنور ب freq مش 1 أو 0 لأ ب freq 38KH وفي بقى شويه

زرار بتطلع ب freq مختلف هنعرف ازاي دلوقتي

ال System الفات دا اسمه Transmiter

وهو ال remote control

لما بنضغط على زرار معين بيطلع Signals ب 38 KHz بس

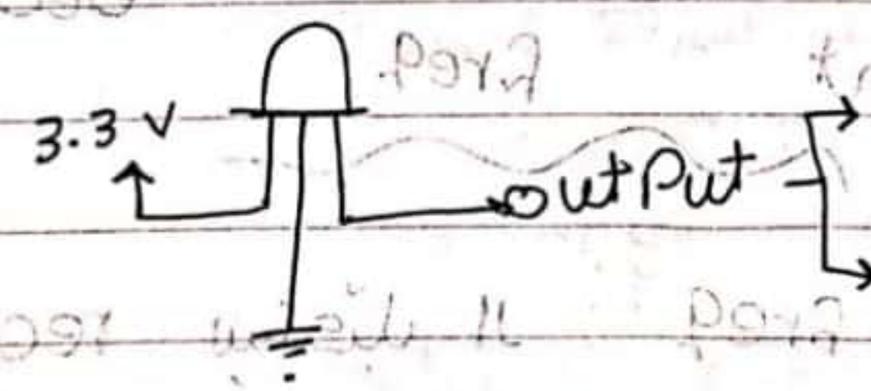
هنشوف الفرق دلوقتي، الناحيه التانية فيه

IR receiver

اسمه Photo diode

أكتر زرار

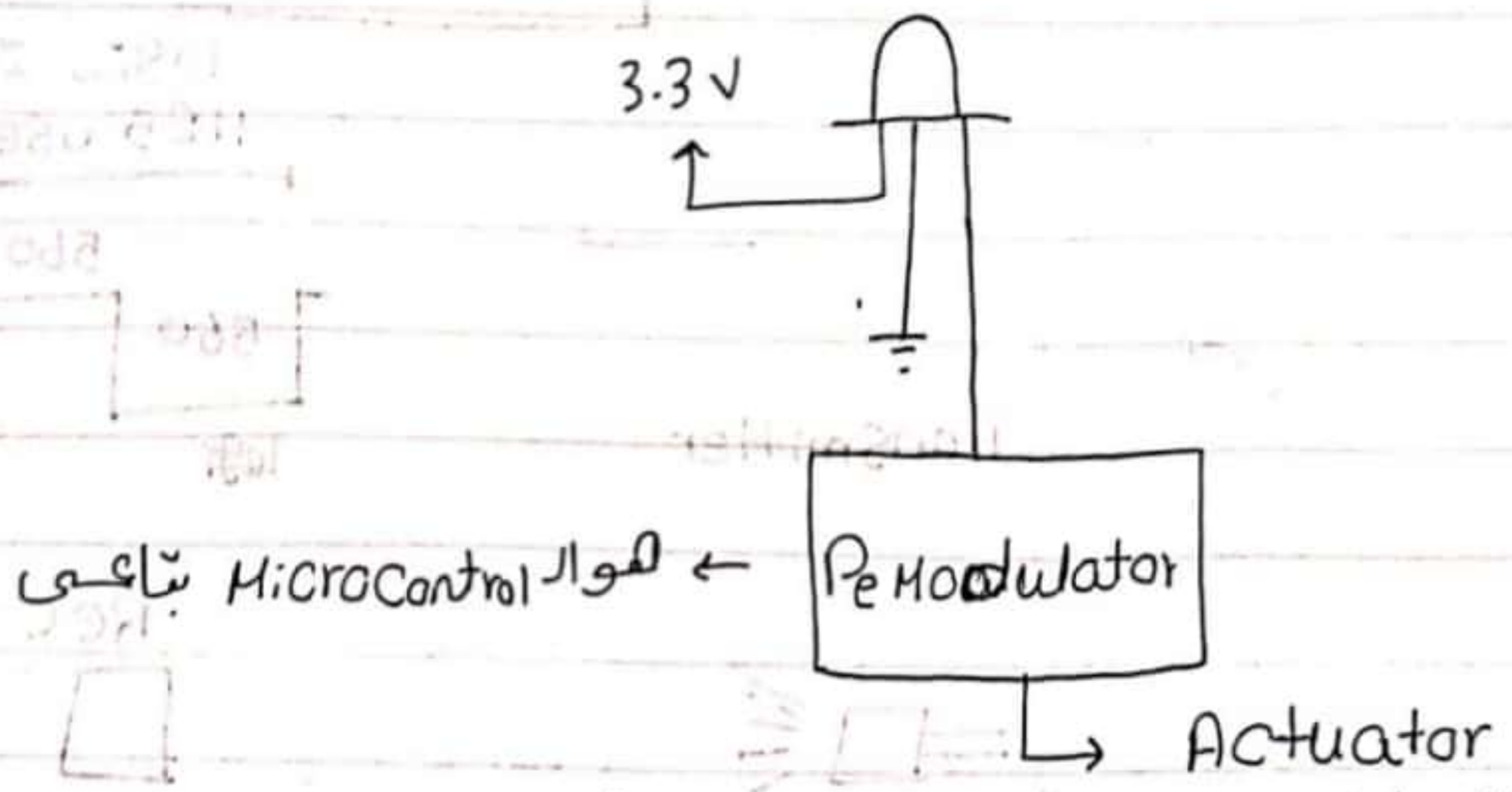
بيكون ٣ أطراف



يعني لو مش مضغوط قدام حاجه هيبقى منور (one) ومتوصل عليه كهربا ، لكن لو جت مثلا IR هيبقى ب zero one بردو لحد ما يبقى IR دا بيتحرك ب 38KHz ،

بيطلع zero لما يبقى حاسس بـ IR بـ 38KHz

نبدأ بقى ناخد ال output دا ونوصله على Modulatore



← هوا MicroControl بتاعى

ال عايز آخد بيه action زى مثلا (leds ، مروحة ، موتير --)

هنا بقى النص الأول ready الهو remote

من أشهر ال Protocols هو من شركة NEC

NEC Protocol

↳ Japanes company

هو Protocol عملاه شركة يابانية عملت IR

باسمها N.E.C Protocol ، سهل ، والشركة دى بتصنع حاجات كتير

IR Protocol ، IR signal with freq. 38 kH

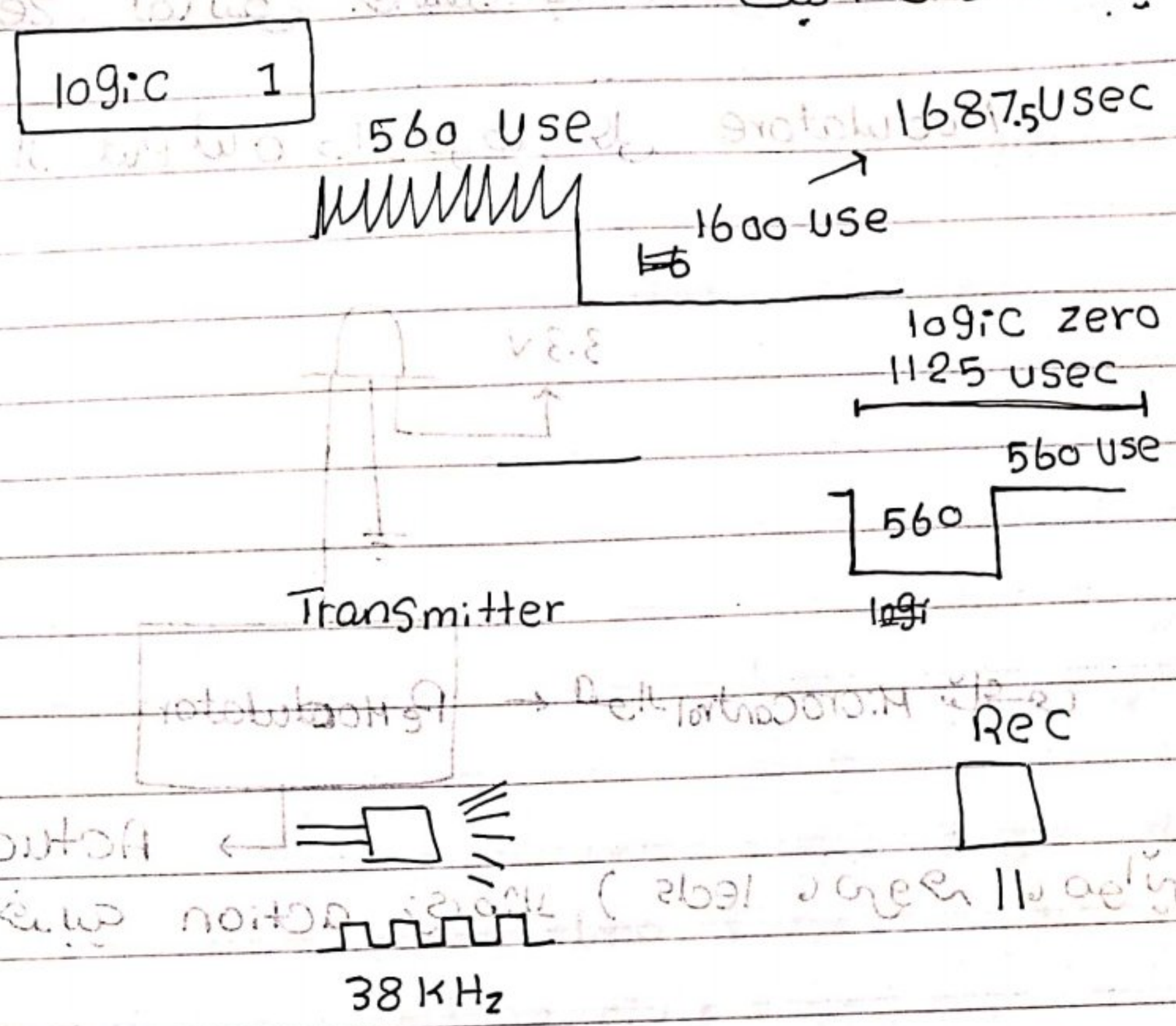
logical 0 — 560 usec — 560 usec

اسمها Pulse train هبعت Pulse train يعنى IR بـ freq

ال receiver يقدر يستخدمه ، طولها 560 use وأقف

نفس الفترة

طيب علشان أبعت



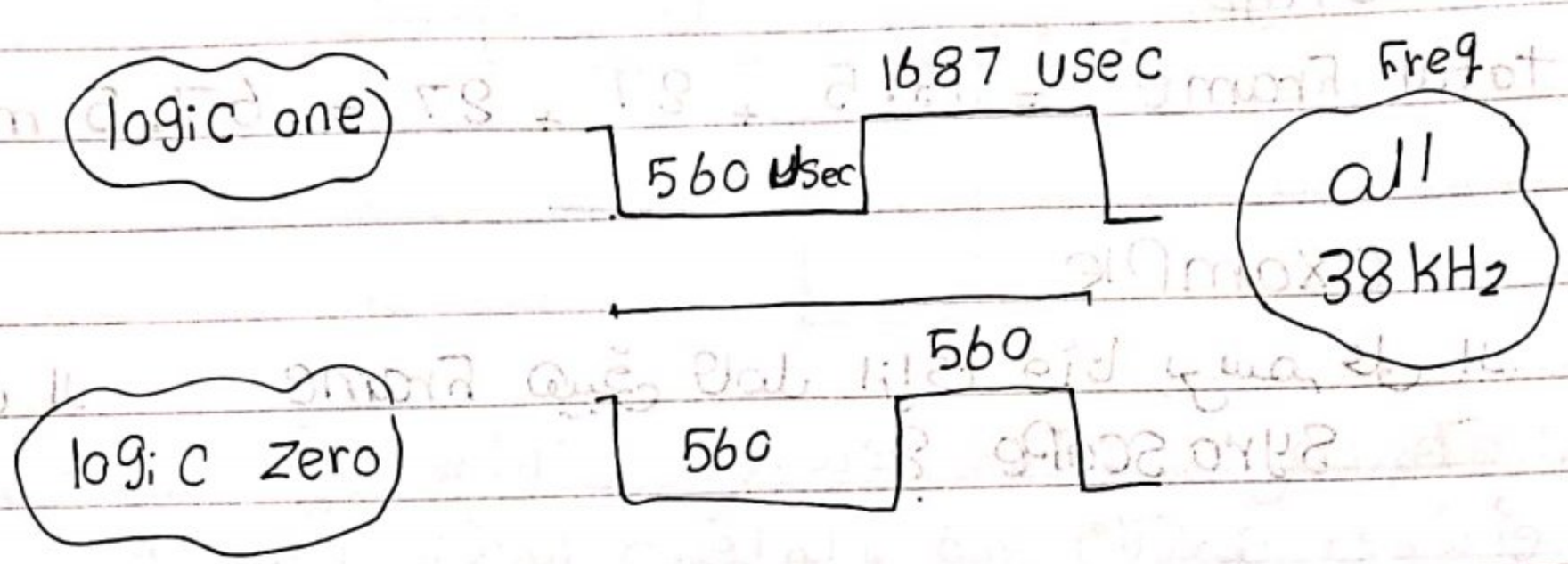
هنا ال Trans هيطلع IR بـ freq 38KHz ولسه ال Rec مستقبلها ش ، طيب ازاى هيفرق أنا باعت zero ولا one ؟ عن طريق الطول ، شوف ال IR طولها كام ، هتبقى مسلط الضوء فترة قد ايه ، لو عايز تبعت zero هتبعت الشكل الى فى الصفحة الورا 560 usec هتنور وتطفى لمدة (زمن) 560 usec ولكن بـ 38 KHz (سرعة) ، يعنى هشغل ال Transmitter هسلط عليه PWM بسرعة 38 KHz واسيبها شغالة عليه لمدة 560 usec كده ال Rec هيحس بـ Signal جاية من ال يوفها وال Signal دى هتستمر قد امه 560 usec فعند ال Rec هيبقى low وبعدين ال Tran هيوقف لمدة 560 usec كان ضهر وقفيش كدا light طالع معنى كدا ال Rec هيبقى high ففهم كدا إن دى bit = 0

0 → 1125 usec

196 1 → 2250 usec

طيب لو عايز أبعت high بقى !

ابعت برده نفس ال Pulse Train لمدة 560 usec يعني
شغل ال led للمدة دى ، بس هتطفيها لمدة 1687 usec



الفكرة فى المدة ال ابعت فيها ال light .

ال frame بتاع ال NEC فيه 32 bit 1 bit Start يبقى كدا كلها 33 bit ، بيعملها ايه ؟ خلى بالك ال frame عند ال Trans غير ال Rec هو نفس الترتيب لكن مش نفس الشكل

NEC Protocol

remote Control
Time ???

Start Bit	8bit Address	8 bit address inverted	8bit data	8bit data inverted

ليه ؟

بتاع ال device ال هكلمه
ودا بيختلف من جهاز ده
لجهاز ، الاتنين هيستقبلوا
respond تلت واحد هيـ
وواحد لا علشان الأدرس

why to send inverted bits ?

↳ 1 Verification للتأكيد ومفيش error حصل فى النص

الأهم

↳ 2 علشان يحافظ على ال Time بتاع ال frame ثابت لأن 0 بياخد 1125 (أقل من 1)
وال 1 بياخد 2250 .

2 Constant Frame Time

Start bit

9msec

4.5msec

frame طول

total frame = 13.5 + 27 + 27 = 67.5 msec

example

شكل ال Frame هيبقى كامل ازاى وأنا برسمه على ال Syroscope ?

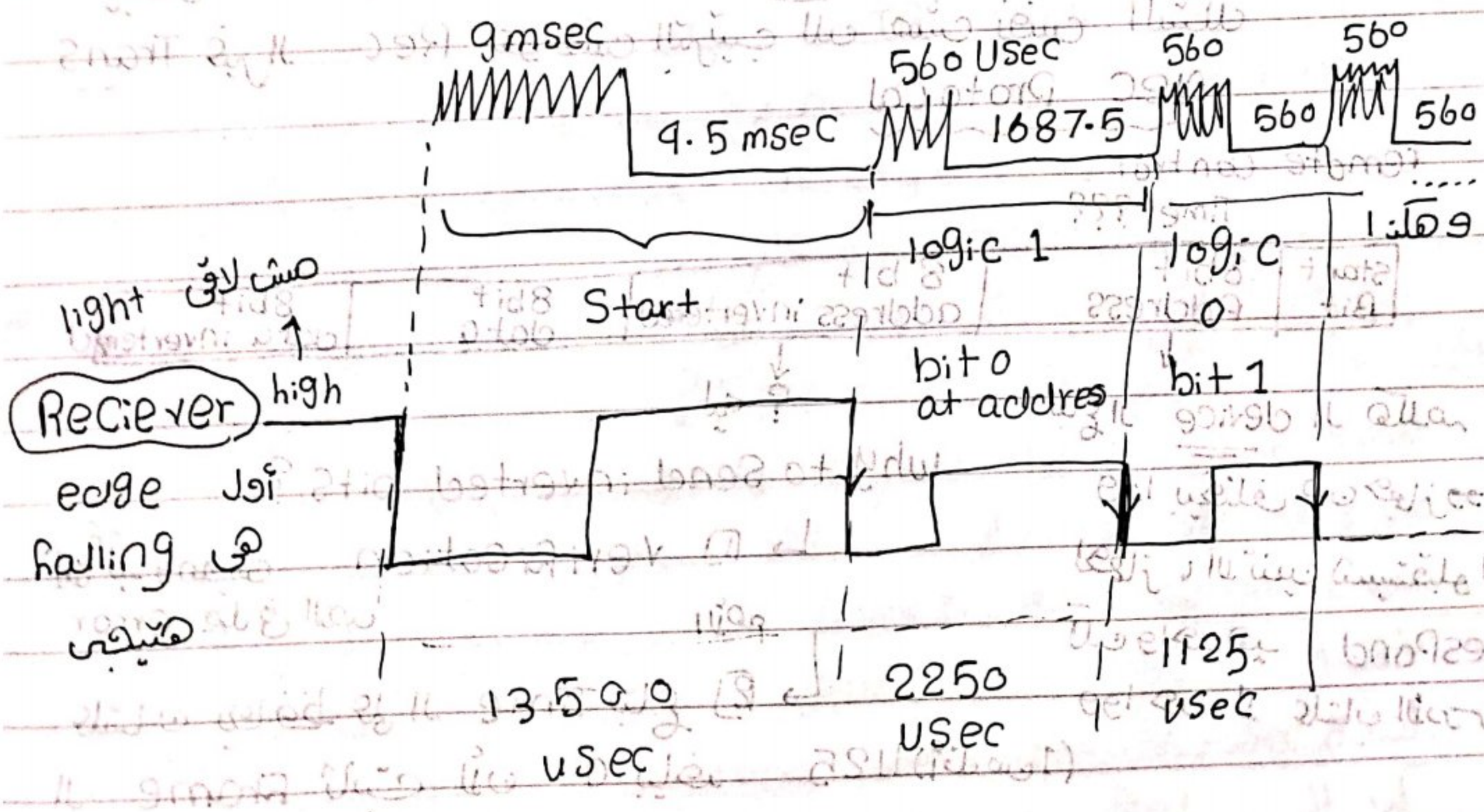
جهاز address = 1 decimal

هبعت data = 10 decimal

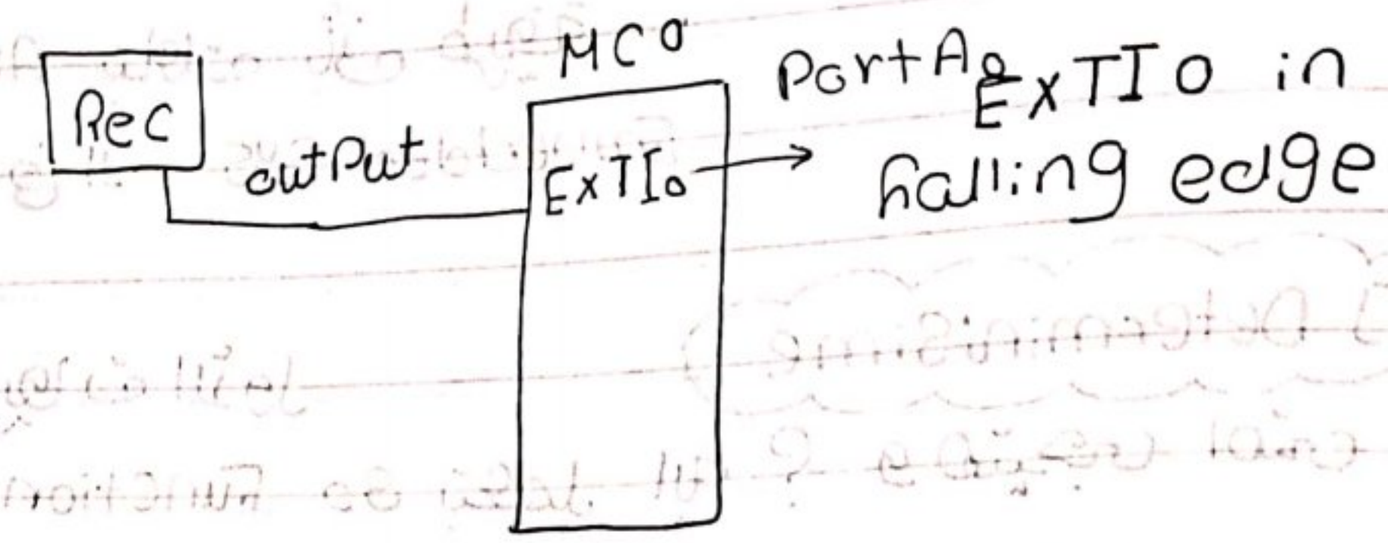
هنرسمه كده من ناحيه ال Transmiter هيبقى كامل ازاى

address

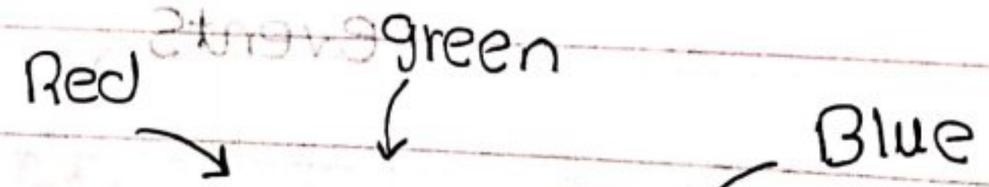
Transmitter 0 0 0 0 0 0 0 1 ← LSB هى الأول



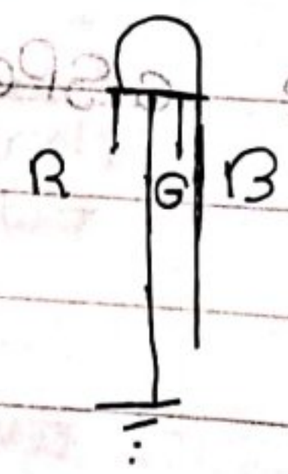
Simple Code :



هنا هكون فجر الـ EXTI إن ايدين int مع كل falling edge
وأشوف هيعدكام عدة ، لو لقيت الوقت العدى في
ريجيستر 13 500use يبقى دى Start ، ولو 2250
يبقى دى bit قيمتها one ، ولو في ريجيستر
1125 يبقى bit قيمتها zero



جواها
٣ ليدات
ممكن أحط واحد على أى لون
فيهم علشان ينور أو الاثنين
مع بعض



ASS + Homework + Technical Support

Software Architecture :-

هتبني الـ Architecher بتاعك بأي طريقة :
في ٣ طرق في الـ Embedded SYS

* نشو نشو يه تعريفات الأول

1] Determinisime
معناه الـ Function دي بتعمل ايه ؟ و هتيجي امتى ؟
كل Function

To know what is happening @ every Point

2] responsivness :
(السرعة)
مدى استجابة الـ Software بتاعي للـ events الخارجية

How fast I can response to a specific events

3] Periodicity :-
بقيس الـ Function بتاعتي هتتكرر كام مرة

How often I handle a specific job

S.W Architecture

1 ↳ Super loop Arch... Syst "while(1)"

⇐ استخدمناه كتير

```
int main(void)
{ ≡
  while(1)   or for( ; ; )
  { ≡
  }
}
```

loop

مميزات

* very simple
* Minimal Hardware (MC) وخلاص (بيحتاج شوية صغيرين)
* high Portable ***

يعني تقدر تاخد اللود دا وتحطه على أي Micro Controller

* mean: can be Port to different Controllers just change the registers name.

⇒ عيوب

* Responsiveness ضعيف جداً
* Timing مش مظبوط
* high Power consumption لأن while(1) شغالة على طول فالـ Processor هيشتغل كده كده

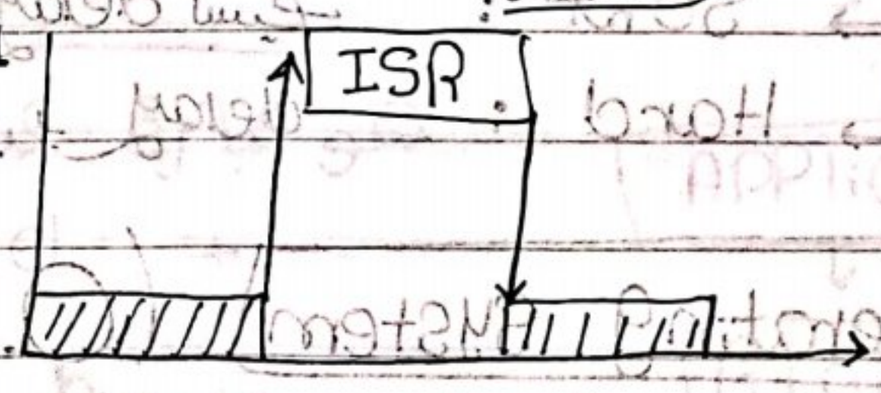
2 ↳ foreground | background

"Super loop + interrupt"

(ممكن يكون أكتر من واحد) foreground 1

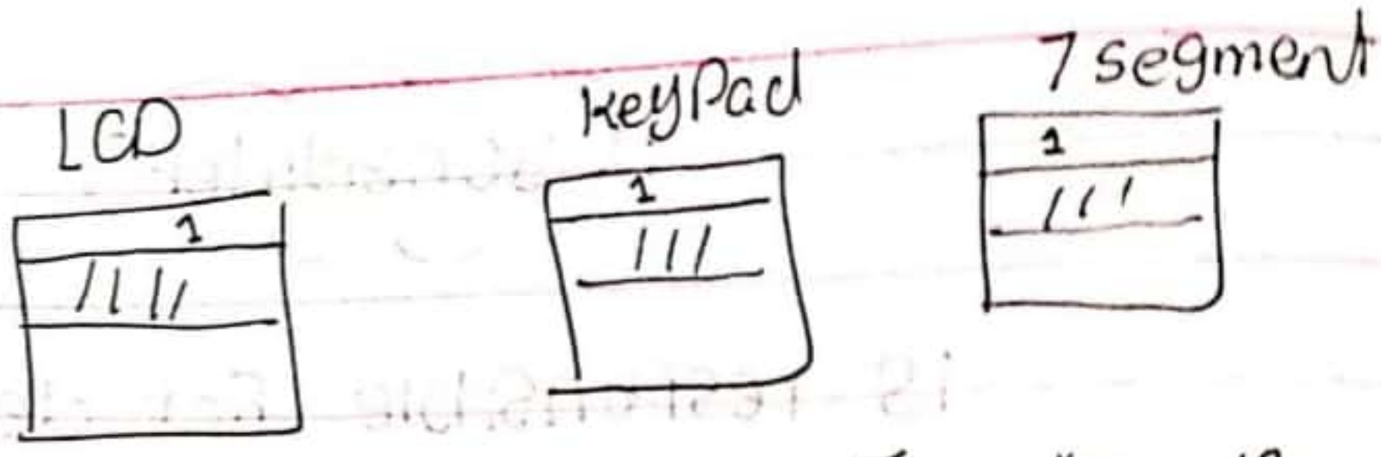
وهنا بال (INT) يه

ومن هنا أقدر أستجيب لـ event خارجية أثناء ما أنا شغال



شغال فيها الـ main:

"event trigger": يعني بالاستجابة لـ events



يعني حتة من كل حاجة ، ومنا بقى ال OS هوال بيعمل ال Multitasking

---

طيب ال RTOS موجود فين ؟
هو مرحلة موجودة بين ال Software وال Application
طيب هو بالنسبة لل Two Archuture التانين ؟

* RTOS :
  - higher Priority than Super loop
  - less Priority compared To The interrupts "Foreground"

يعني أعلى من Super loop وأقل من event trigger

طيب ايه الهدف من RTOS ؟ سؤال مهم
هو
It controlls The order of execution of The "Background"

طيب

RTOS : هو عبارة عن ايه

kernel + ...

- objects
- Services
- Scheduler

أول أهم مكونات في ال kernel

* Datatypes used by programmer as Task
  - Create Task Kill Task

ال هي ال funcs ال بنطبقها على ال objects

Application
↓
RTOS
↓
Hardware

طيب إيه هى ال Scheduler :

is responsible for determing
The order of execution Tasks

يعنى هو أساس ال RTOS هو المسئول عن ترتيب
ال Tasks وميت ال Tasks ال عليها الدور تروحله
هو دا عصب ال RTOS

* Scheduler → Algorithm (مخ)
↳ Dispatcher (عضلات)

* dispatcher:

بيشوف ايه ال ~~discussion~~ decision ال أخده ال Algothim
ويروح ينفذها

mean : executs the decision Taken by the
Algorithm

زى مثلا "Context Switching"

Algorithm : : which "Tasks" should be run

أى تاسك ال هتشتغل ، أى تاسك ال عليها الدور فى التنفيذ .

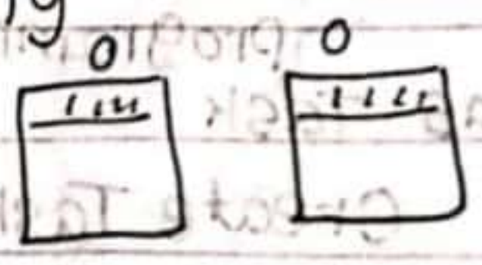
كل Algo 1) Shortest Task first come. (أقل execution Time)

الجوريزم تانى 2) Priority based

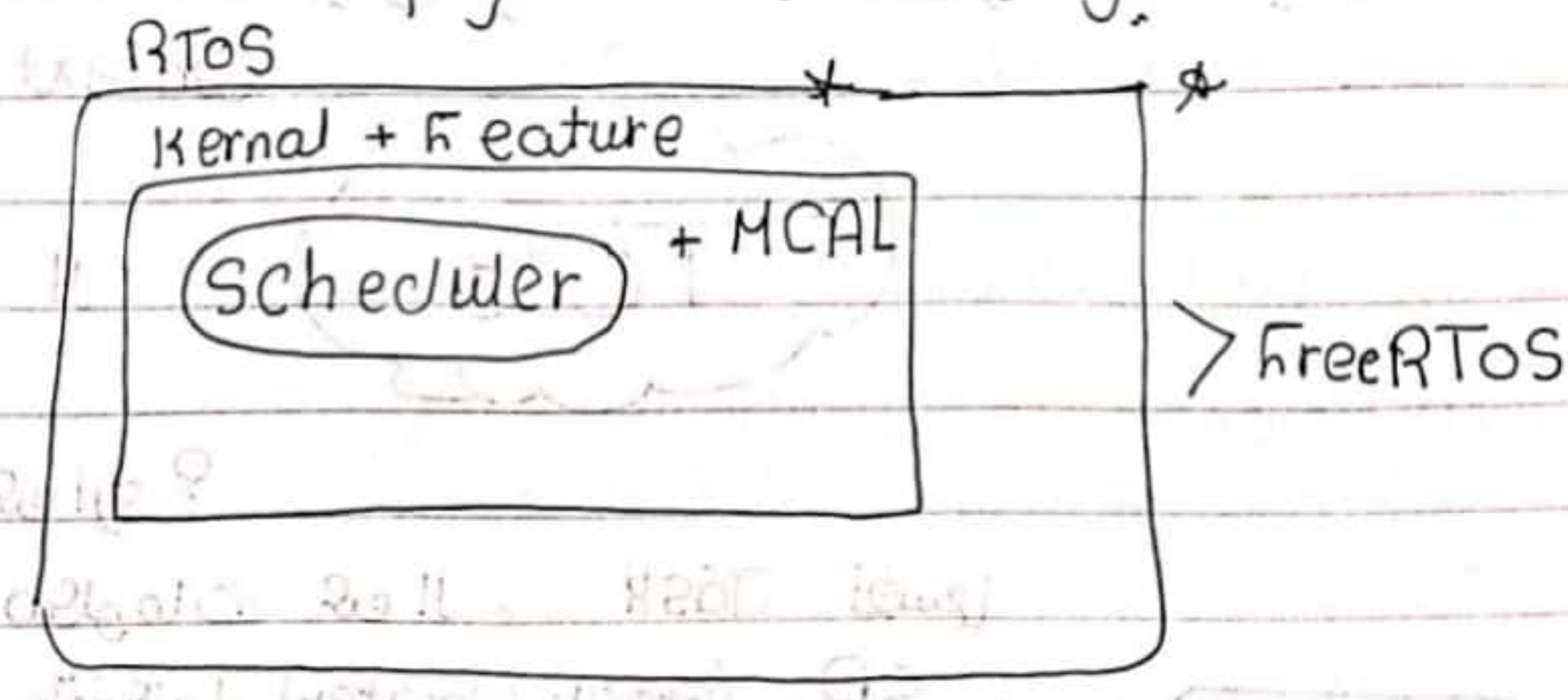
3) Round robin "Time slicing"

نفس ال Priority

هيا Timing من دى وينفذها وياخد من التانية
وينفذها وهلما



في Algorithms كتير، كلك دول أشهرهم.



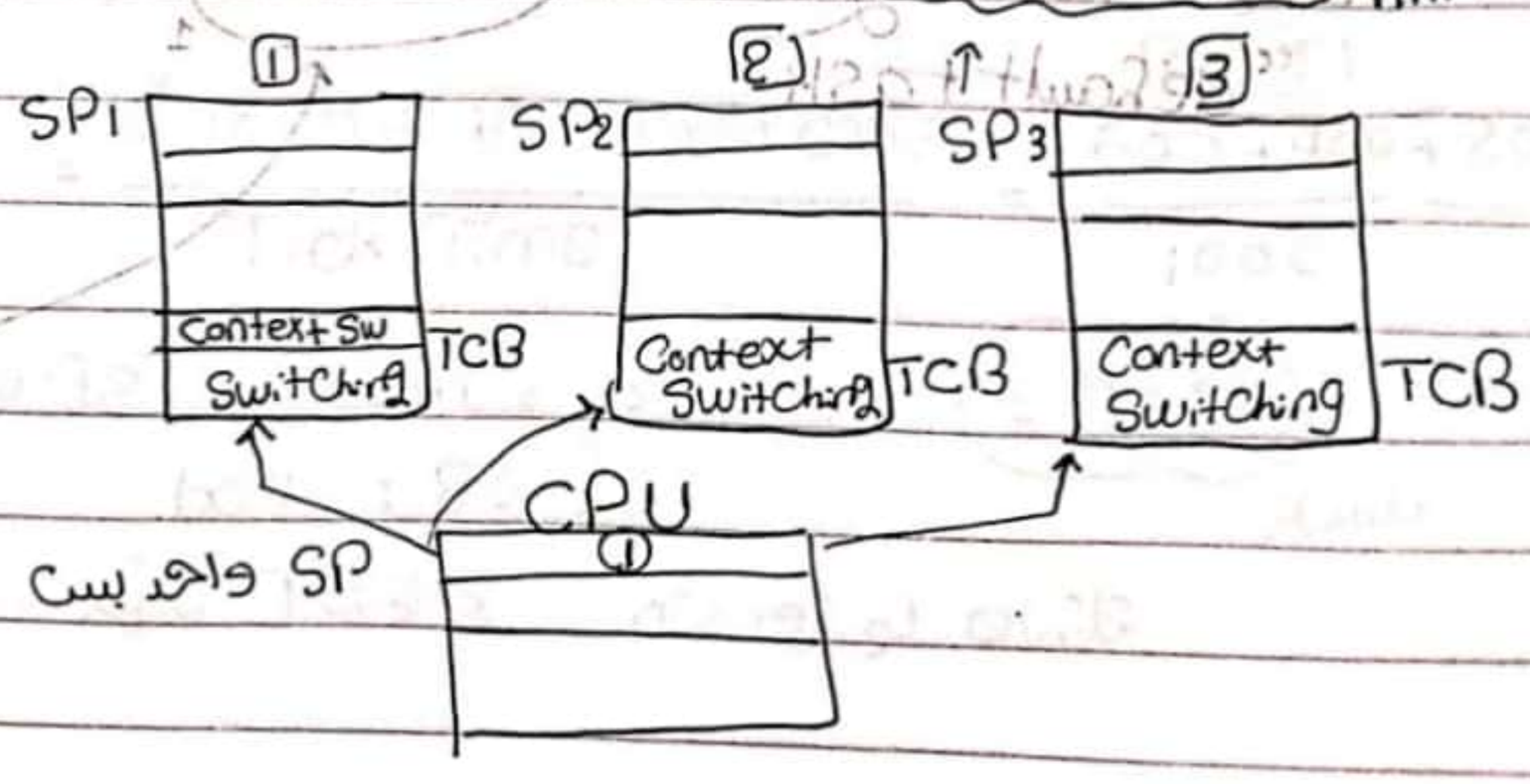
⇒ what is The difference between function and Task ?

Task ⇒ C function + Timing + Context Switching + Storage

↳ TCB (Task Control Block)

كل Task ماكرة نفسها حاجزة ال CPU لوحدها وداماش حقيقي والسبب في كدا ال Structure بتاعة ال Task وظيفتها تحفظلك ال Structure دا في حاجة هتنظملنا ال Structure دي وايه ايه ؟

كل Task ليها ال TCB بتاعها



كل Task بيبقى ليه TCB بيتحفظ فيه ال Context Switching

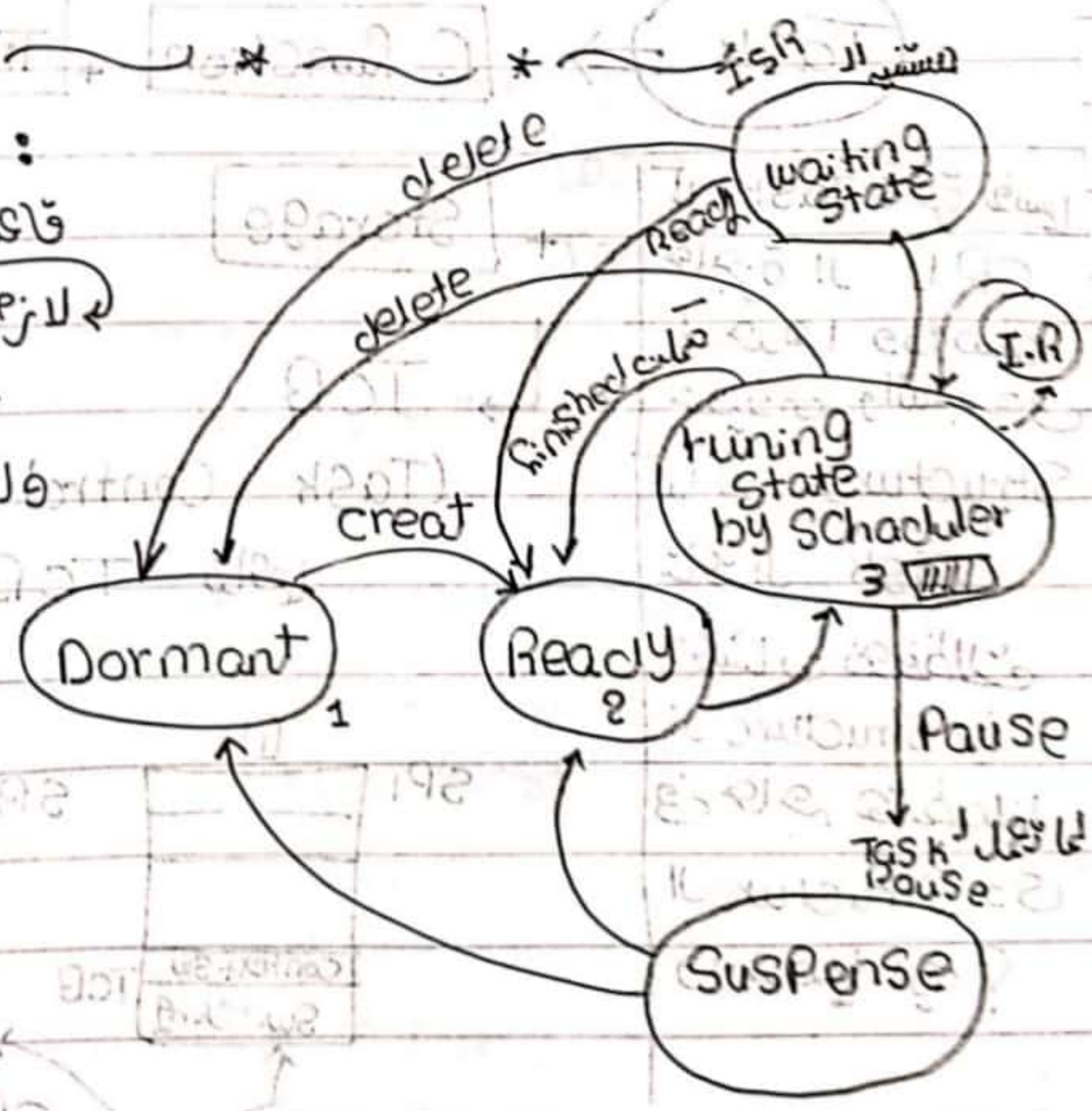
طيب ال TCB عبارة عن ايه ؟

فيه معلومات عن ال Task نفسها

TCB
- SP (Stack Pointer) بيبدأ منين أو واقف عند ايه
- Priodicity ⇒ بتكرر قد ايه
- Context Switching of Task ⇐
- first delay

Task states :

قاعدة :
لازم ال running يكون فيها حد مينفعش تبقى فاضية .
ولو مفيش بنعمل Task كدا ترن على الفاضى علشان نشغل ال running ال هى "default task"



Schedulr :→

تعبّره function بتناده لل Tick Time

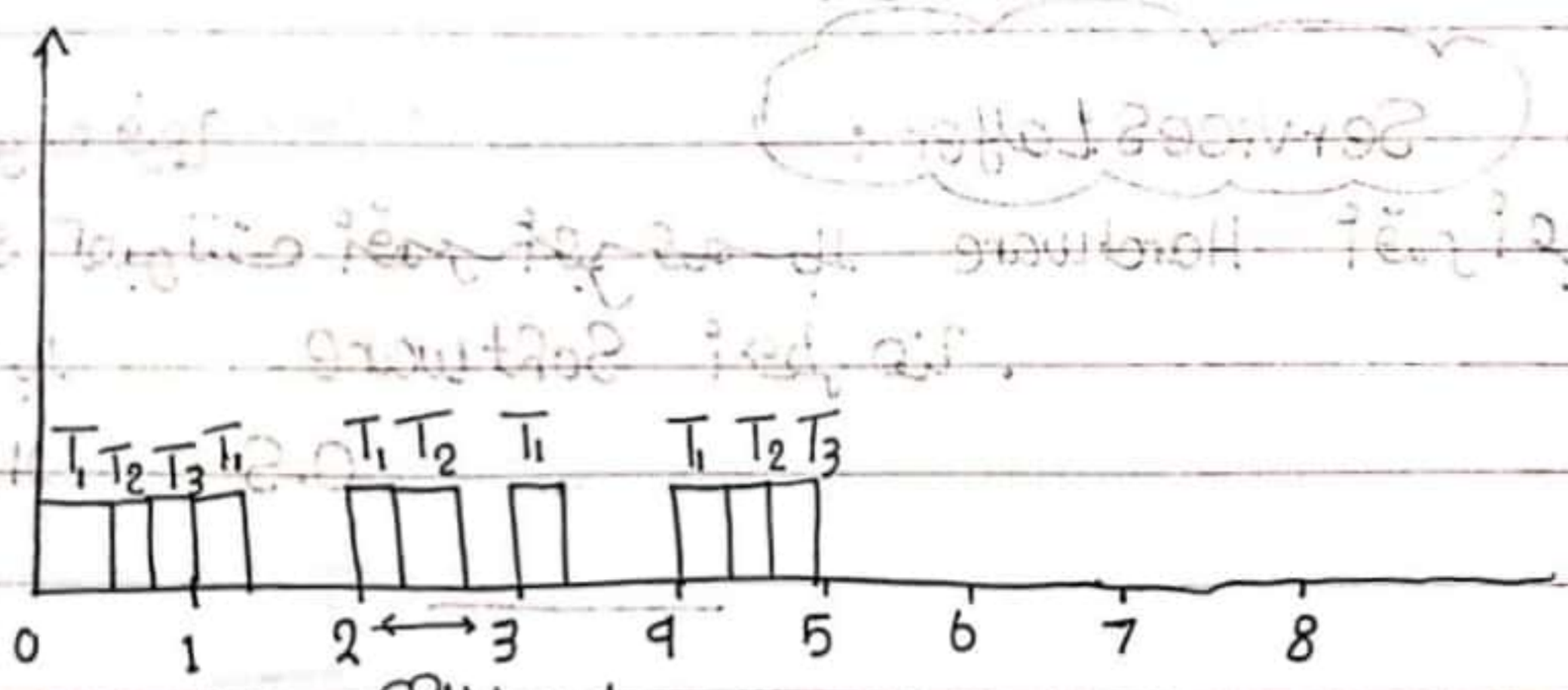
Schedulr : → Tick → ordering = 1 mSec or 0.5 mSec

ex;

	(Priodicity) أنا دي عليها	executionTime
Task 1	→ 1 mSec	→ 500 MSec
Task 2	→ 2 mSec	→ 400 MSec
Task 3	→ 4 mSec	→ 200 MSec

أرقام صحيحة

Timing diagram عايزين نعمل design لل Tick



$$\text{CPU load} = \frac{500 + 400}{1000} = 90\%$$

* CPU load

يعني ال CPU مستخدم بنسبه قد ايه؟

$$\text{CPU load} = \frac{\sum \text{Tasks in worst case}}{\text{Tick Time}} = \frac{500 + 400 + 200}{1000} = 110\%$$

$\boxed{0 : 1}$

مشكلة

طيب أنسب Range لل CPU

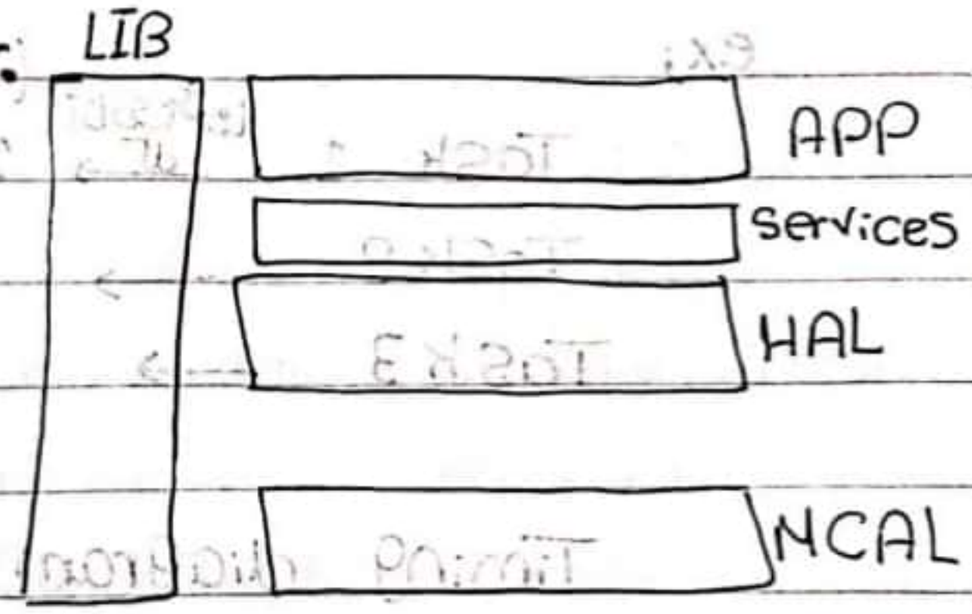
60% : 80%

علشان لو زودت Tasks تانيه

First delay : note مشروحة كويس في ال

* ~ * ~ * ~

→ layerd Architecture :



Services Layer : بيكون فيها

① أي كمبونتس أقدر أعبر عنه بال Hardware أقدر أعبر عنه بال Software أحط هنا .

② وال O.S

* CPU load

CPU load = $\frac{\text{Ticks in workload}}{\text{Tick Time}}$ = $\frac{500 + 400 + 100}{1000}$

CPU load = $\frac{500 + 400}{1000}$ = 90%

CPU load = 90%

CPU L 11 Range جميل جدا

لو : 80 %

= 110 %

مثال Ticks جديدة جاية

W37

Lec 16

Audio Signal :-

Basics :

في عالم الـ embedded
Real world — Embedded Systems

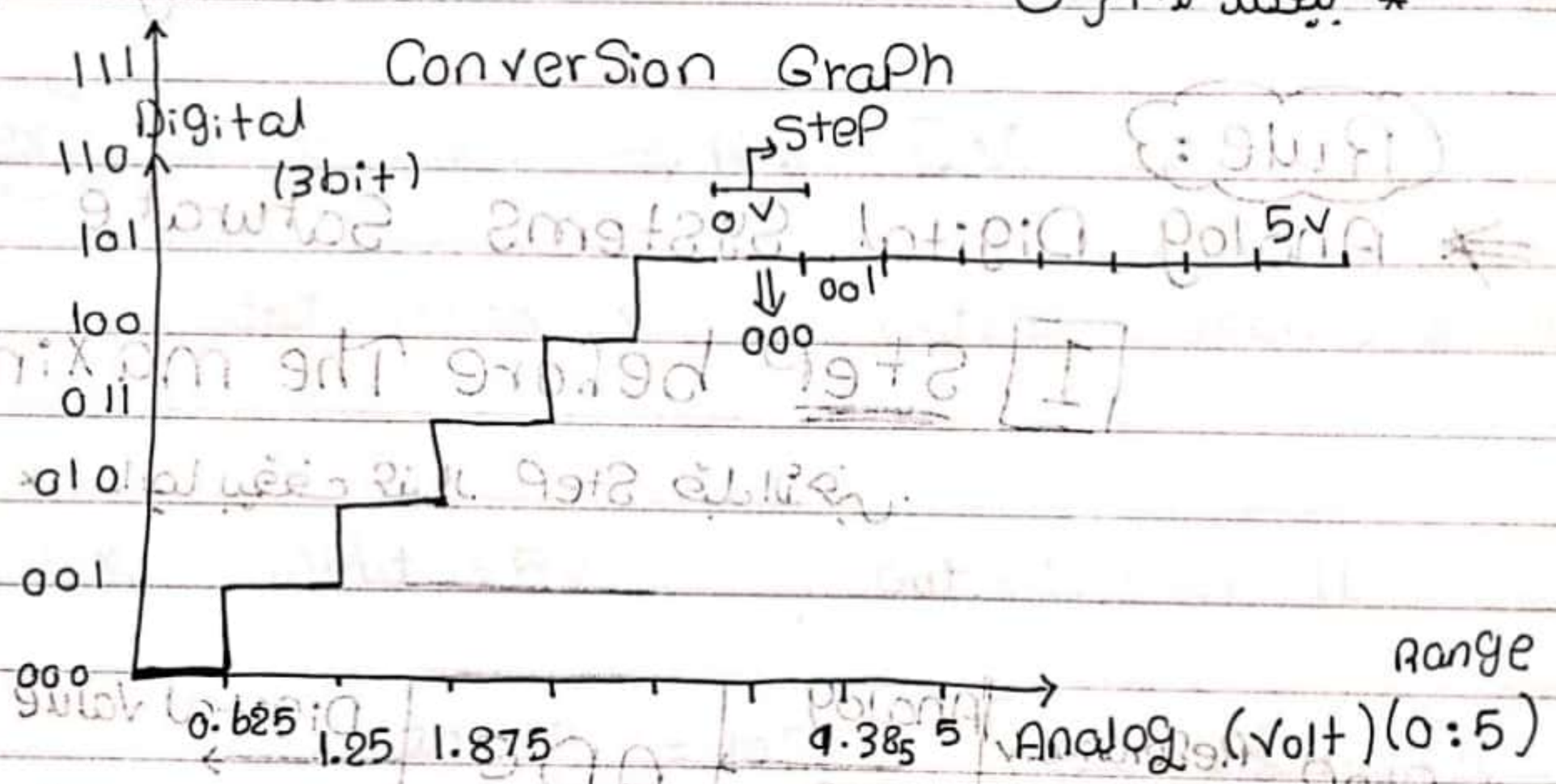
Analog → Analog To Digital ADC converter → Digital

Analog عايز يدخل عالم الـ digital عالم الـ

Resolution يعني هتشيل الداتا في كام bit (n)

* بيعمل كدا ازاي

Conversion Graph



هو هيشوف قيمة من الـ Analog ويحطها على القيم الـ digital عند الـ

step :

معناها القيمة الـ Analog الـ هتحركني one step في الـ digital 8 احتمالات لأن دا الـ عندي

* كدا هنمسك من 0 : 5 ونجزأها

$$\text{Step} = \frac{\text{Maximum Analog voltage}}{2^n}$$

$$\text{ex; } \frac{5}{2^3} = 0.625$$

تحذير:

بعض الناس بيعملو ال ح وده خطأ فاضح. *

معناه إن كل ما يزيد ال resolution هتزيد الدقة أكتر لأن
مثلا هتقسم على رقم أكبر بالتالي هيمثله رقم digital وهكذا
بس مش هنوصل لدقة ال analog.

ex: Digital → Analog *
↳ 000 0

هو في الحقيقة range
بس هنقرب للأصغر

Rule:

* Analog Digital Systems Saturate

$\boxed{1}$ Step before The maximum

* دايما بيقف عند ال Step قبل الأخير.

* $Step = \frac{Reference_V}{2^{Resolution}}$

Analog value → ADC → Digital value

ADC:
- → Resolution
- → Reference voltage (Max_V)

* Analog value = digital value * Step
001 * 0.625

Analog value = Digital value * $\frac{Reference}{2^{Resolution}}$

ex; 10 bit ADC , Ref voltage 3.3

* Digital = 1010 1111 00 (= 700)

$$\text{Analog Voltag} = 700 * \frac{3.3}{2^{10}} = 2.25 \text{ Volt}$$

$$\underset{\text{(الأصغر)}}{2.25} \xrightarrow{:} 2.25 + \text{step}$$

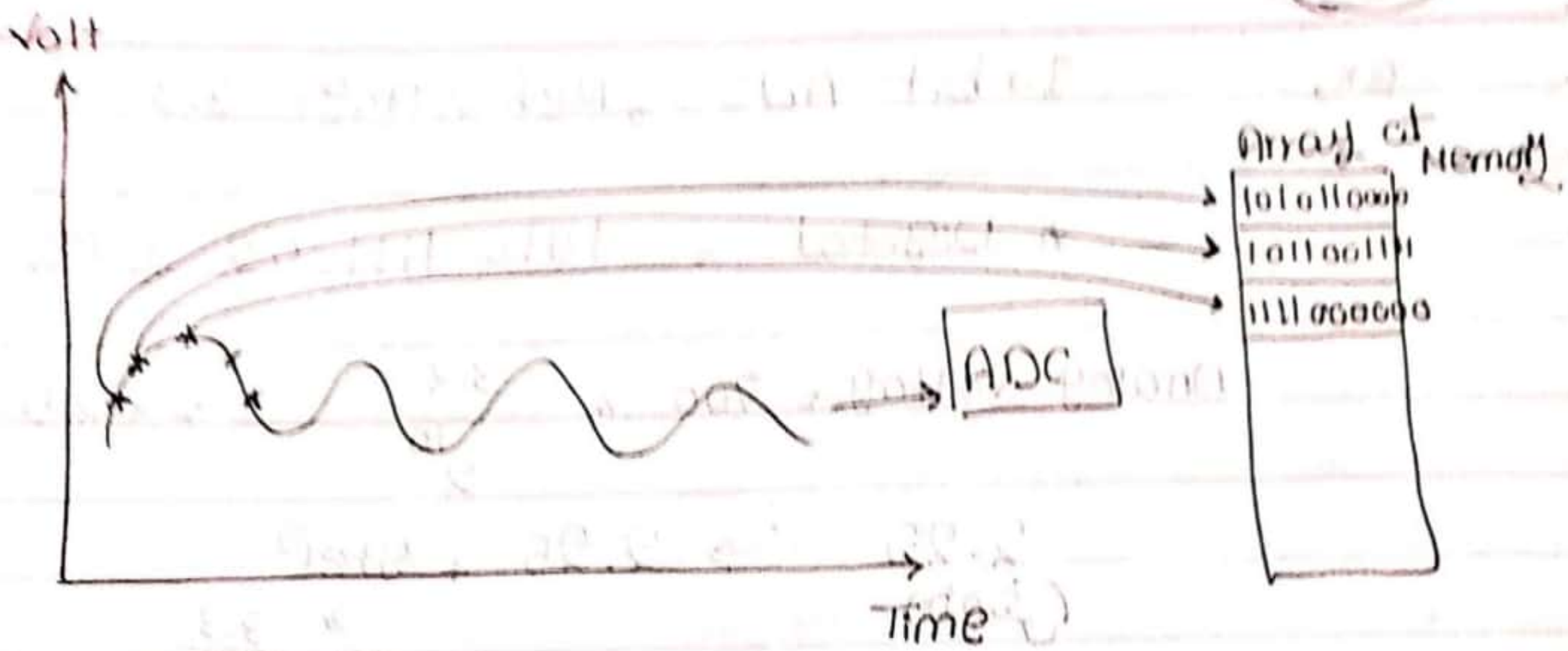
$$\text{step} \downarrow \frac{3.3}{1024}$$

ex; 10 bit ⇒ Ref 5V

out in ⇒ 0V output = 0000 0000 00

ex; input = 5V out = 1111 1111 11

$$\text{analog} = 1023 * \frac{5}{1024} = 4.995$$



1. Sampling

يعني هاخد Samples عندك وامسك كل Sample
فيهم وأوديها لـ ADC

2. Quentization التحويل

يعني أشوفها في أي range

كلها في error لكل نوع 1، 2
فهنا ظهر عالم اسمه Nyqueot
قال

$$\text{Sampling freq} \geq 2 * \text{Max freq}$$

* The more Samples. The better accuracy, The higher memory consumption

Audiable Signals

* هي Signals المسموعة ×
  20 Hz to 20 kHz الإنسان بيسمع ال
  في ال range دا بس

دا ال بنسمعه

* human voice : دا ال بنتكلم بيه 150 : 9 kHz
  ⇒ male 150 Hz → 3.5 kHz
  ⇒ Female 250 Hz → 9 kHz كل ما يكبر بيقل حاد.

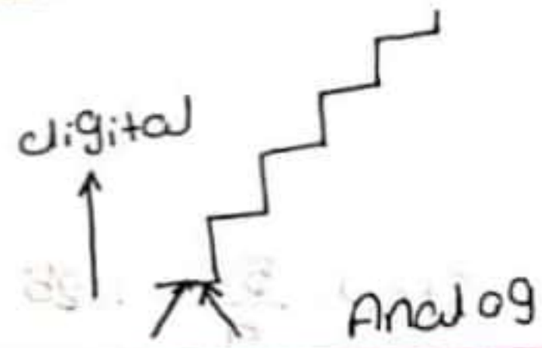
Highest freq in human voice = 90 kHz
Based on
Nyqueot → Sampling freq >= 2*Max

* Sampling freq >= 8 kHz
  for human
  voice

يعني علشان أحول صوت إنسان لصيغة لازم آخد
Sample ← 8000 Hz في الثانية
ودا أقل حاجة في الدقة.

"فرضا" assume
↳ 1 Sample
= 1 byte
1 Sec ⟹ 8 k byte 69 k byte

169



17 ⇒ what types of errors would be exist in the output signal compared to input signals?

1. Sampling error.
2. Quantization error.
3.

Q 3 DAC output my be equal to the reference voltage ....?

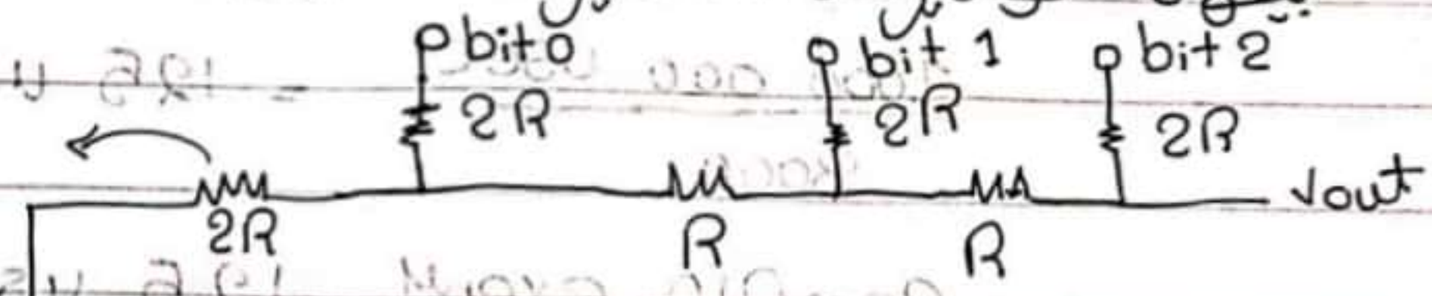
Never

معلومة

ال ADC نفسه بيحتوي على DAC وهو أعقد من ال DAC

R2R DAC

بيقول لو عايز تعمل دايرة DAC هات مقاومة "2R دايما"



كدا 3 bits ← DAC وكلما تعوز تزود bits، زود R نايمة و 2R واقفة

$Step = \frac{5V}{4} = 1.25$

2 Bit DAC

علشان نثبت الكلام دا ، على هنثبت على

inPut	outPut
00	0.0
01	1.25
10	2.5
11	3.75

Home work:

↳ using 8 bit R2R DAC

write C code to generate

Sign wave

rate = 8000 Sample / sec

$\frac{1000\,000 \text{ usec}}{8000} = 125$ usec

* Sample every 125 usec

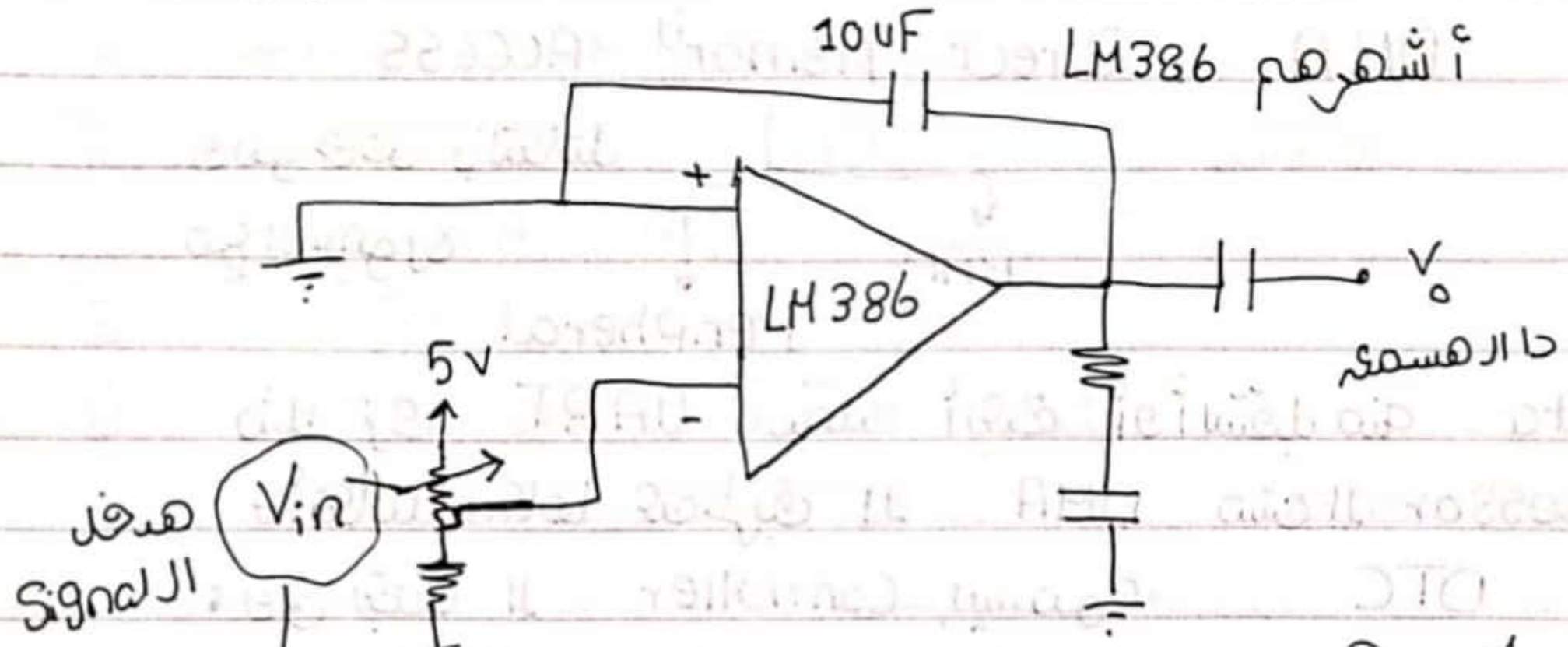
171

$Max = 3.3 - Step$

$- \frac{3.3}{2^8} = \boxed{}$

Amplifier

Audio Amplifier :



⇒ fixed Gain 200 x 200 %

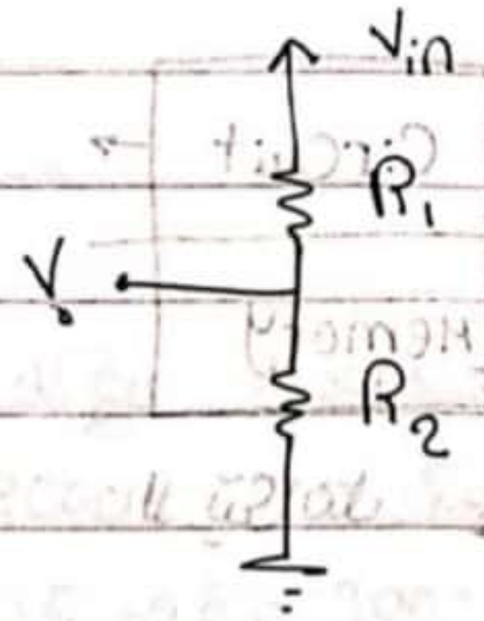
أى فولت هيضربه فى 200

هناخدها (Signal)

نصغرها الأول وبعدين ندخل تكبير

هنصغرها بال المقاومة المتغيرة

، المقاومة المتغيرة عبارة عن



$$V_o = V_{in} \frac{R_2}{R_1 + R_2}$$

Lec17

DMA

مقامه من مقام ال Processor

DMA : Direct Memory Access

وهو خاص بالتكامل مع الميموري

RAM

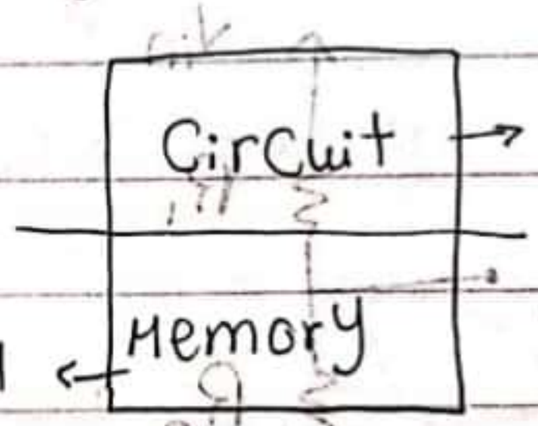
Peripheral

مثلا زي UART بتمكن أبعت أو أستقبل منه data وأتكامل معاها عن طريق ال DMA مش ال Processor، وفي بعض ال Controller بيسموه DTC Direct Tranfer Controller.

ايه الهدف من DMA أو DTC

Mission: Copy data from Source Memory To a destination Memory

* أساسا ال Peripheral عبارة عن ميموري لأن دايما بيبقى فيه Buffer بيحفظ الداتا (لازم) أي Peripheral عبارة عن جزئيت.



Try step Buffer

ال registers

وال Processor لا يتكامل إلا مع Memory

For ex;

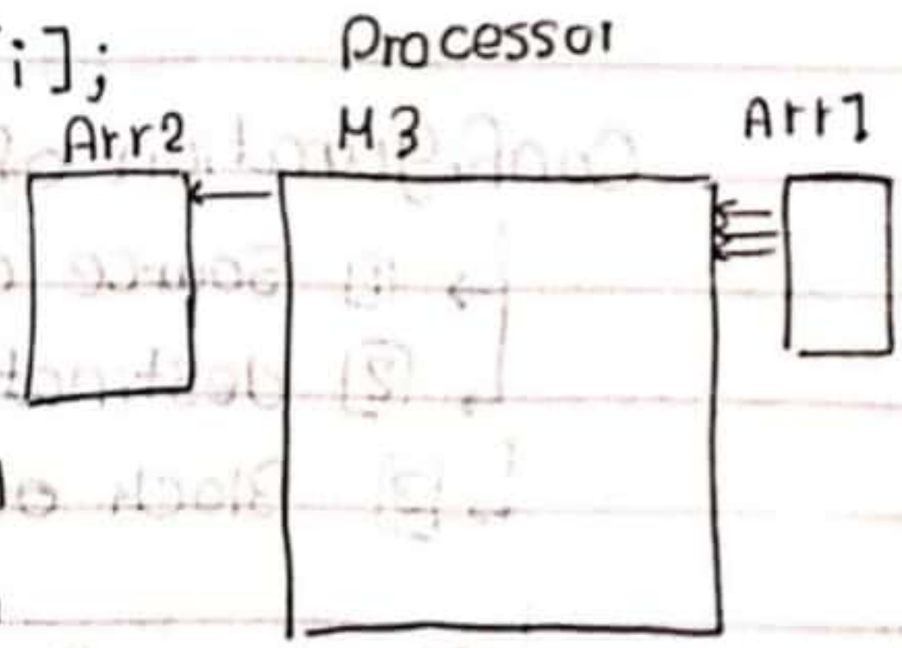
DMA :

1. Peripheral يفرغها على RAM — ممكن ينقل من ال
(Sent ~~from~~ to UART) RAM → UART — مثلا من
2. Peripheral → RAM — والعكس
(receive from UART)
3. RAM → RAM
4. Peripheral → Peripheral
as ; UART → SPI — ودي مهمة جداً

* ~~~~~~~~~~ *

ex;

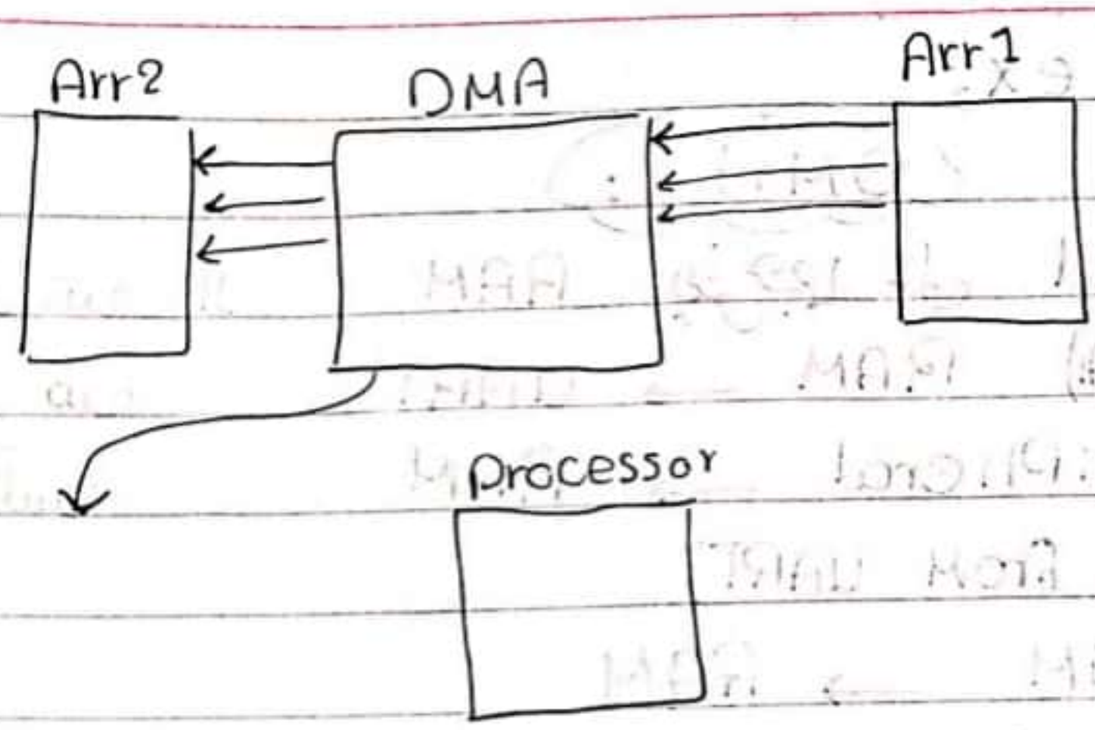
```
~~void~~ int main (void)
{
    u8 Arr1 [1000] = {0};
    u8 Arr2 [1000];

    u16 i = 0;
    For (i=0 ; i< 1000 ; i++)
    {   Arr2 [i] = Arr1 [i];
    }
}
```



هيعمل iteration لكل ال Processor
fetch و decode و excute لكل مرة
(3 cyde) clock cycle يعني 1000 مرة في عدد ال
وهنا بقى عايزين نريح ال Processor شوية

$1000 * 3 \approx 3000$ clock cycle

ال DMA



ايه ال هيحصل ؟

ال DMA هيقرأ arr1 وبعد مايقرأها هيعمل Save في arr2 ، وال Pro مش هيعمل أي حاجه لنا ، ال Processor بيعرف منين إن DMA خلص نقل

* ال DMA هتديك interrupt

* After finished will generate interrupt

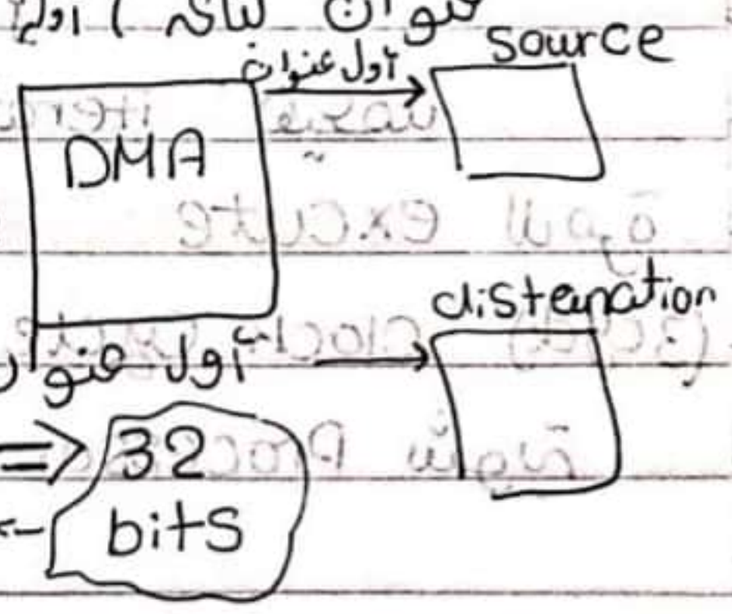
وبكدا هو حل محل ال Processor .

DMA

Core Peripheral تبع ST

Configuration of DMA :

- [1] Source address (المكان اللي عايز أنقل منه أديله ال عنوان بتاعه) أول
- [2] destination address
- [3] Block of length كام byte



- [4] Size of element :-
  - dependent on data bus ⇒ 32 bits ← بتاعه STM32

ممكن ننقل (1byte , 2B , 4B)
آخرنا

STINC	DINC	Case
1	1	Array To Array
0	1	① Array initialization ② read data from Peripheral (UART) To Array
1	0	 ① Array To Peripheral (UART)
0	0	 ① Bridge between 2 Peripheral "Communication Protocols" M.C (SPI) ⇄ M.C ⇄ M.C (UART) ونعمل دا بدل الرسم الفوق دى DMA

* DMA :

* Can be Configured To Trigger Source anew Data based on Peri Pheral event
→ Free running (don't wait any Thing)

6 DMA :

can repeat → "(Circuler)"
يعني ينقل الداتا أكتر من مرة

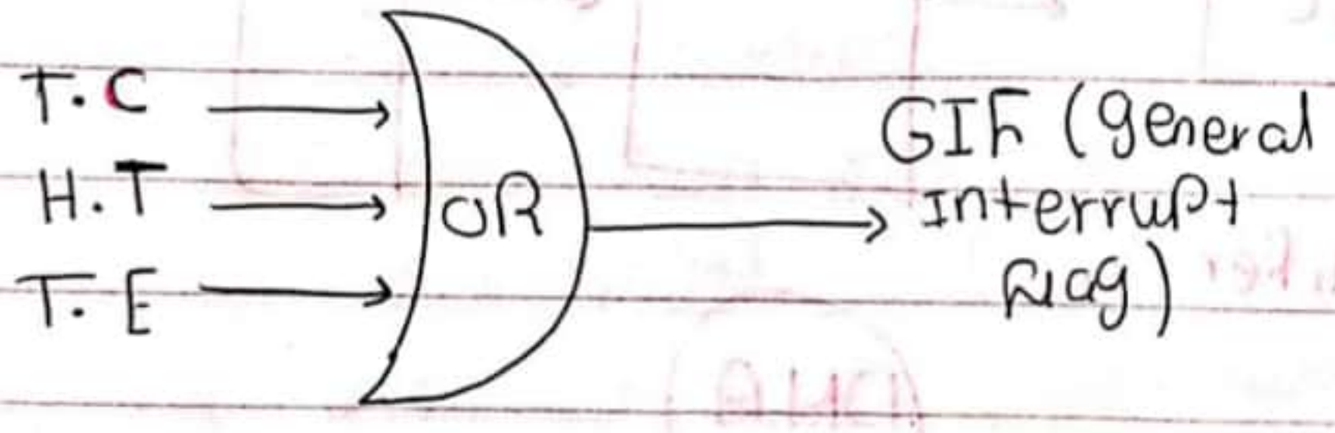
ايه ال interrupt ال ممكن تولدها ال DMA

DMA Interrupts :

1) Transfere Complete :
لما يخلص نقل داتا يدى *interrupt

2) half Transfere complete
لما يخلص نقل نص الداتا

3) Transfere error (Memory Consumed) أشهر ال error
لو حصل error أثناء نقل الداتا



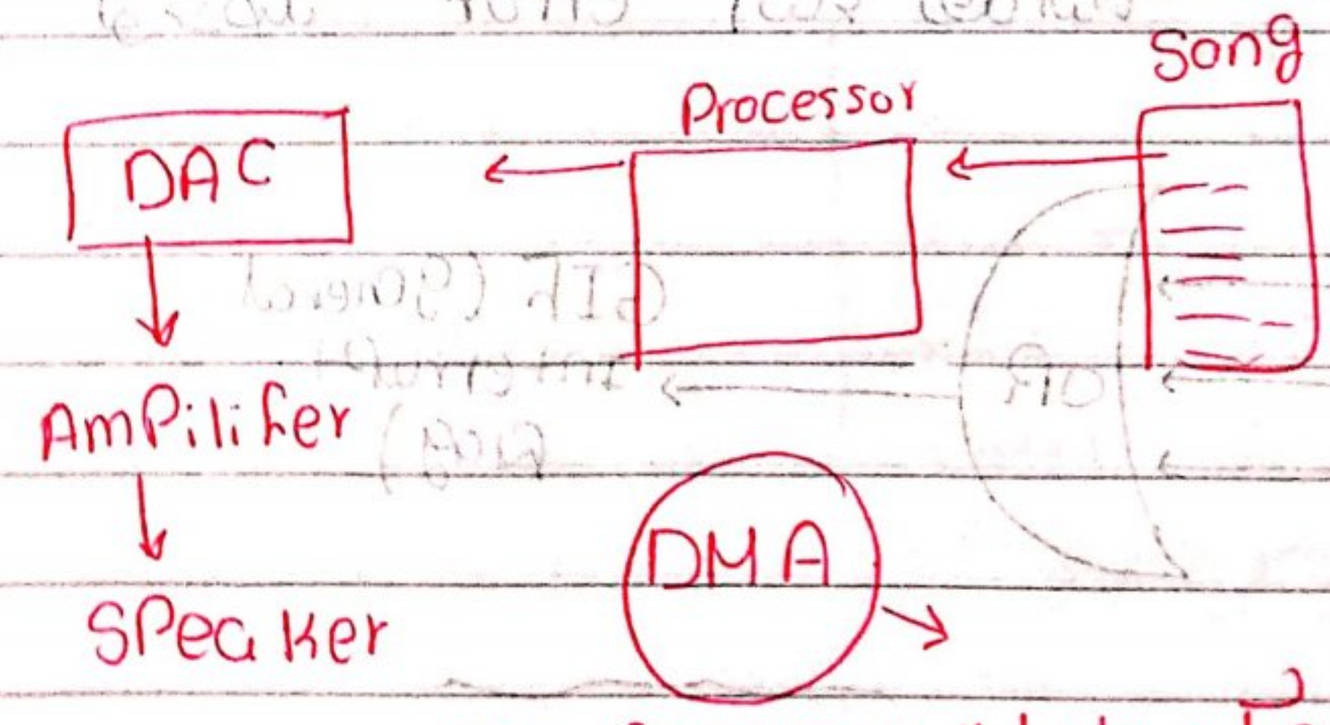
DMA → AHB

→ The Same clock for Processor and DMA ?
which is faster Than?
DMA is faster Than Processor ?
(fetch, d, excute) Processor لأن مش بيعمل زي ال
فهي أسرع .

(2) DMA , Processor access The Same location
(Trying To Access)
أنا ليه الأولوية
* DMA higher Priority Than The Processor
لأنه أسرع من ال Processor

So; DMA is Considered another Processor.

أهمية ال DMA:



ممكن دا يدل ال Processor هنا
وممكن تتحكم في سرعة ال DMA

3) DMA Channels :- 7 Channel

1	2	3	4
UART	SPI	I2C	CAN

بيشوف ال Channel ال high Priority ويروح يشغلها
وعندنا نوعين من ال Priority

- S.W → SW over H.W
- H.W → H.W Connection (1 2 3 ---- 7)

S.W Priority :- (has higher Than Priority)

1) very high
2) high
3) medium
4) low

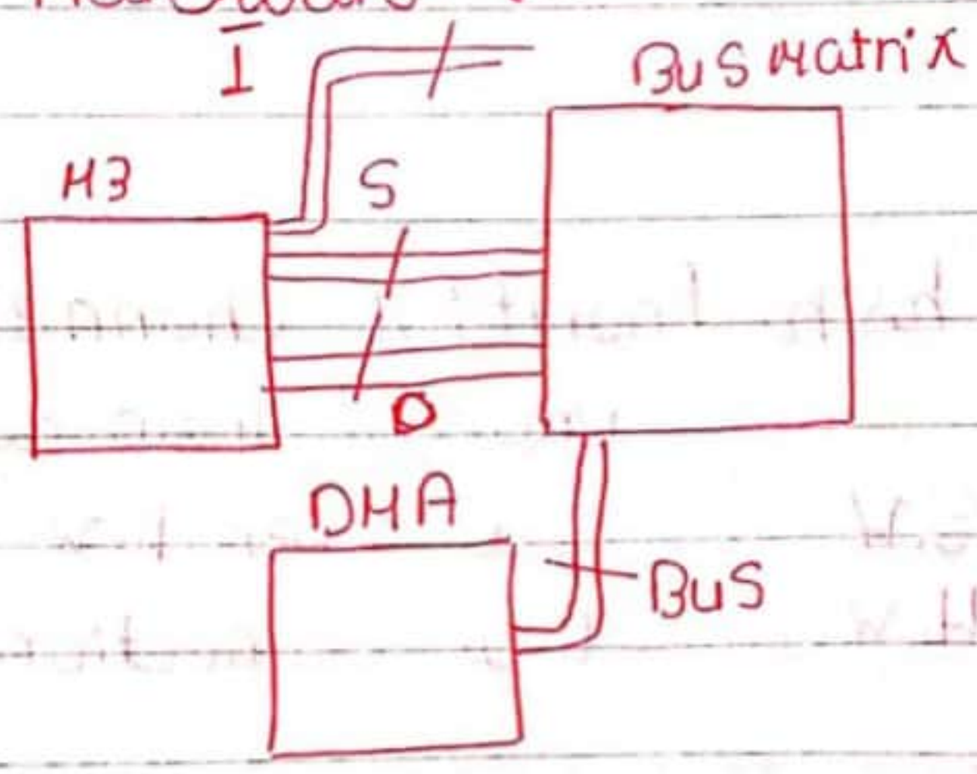
DMA Configuration at file (Config)

1. Source address.
2. destination address.
3. block length (num byte).
4. element Size (~~word size~~).
5. SINC , DINC.
6. Trigger Source.
7. repeat (circuler).
8. Interrupt.

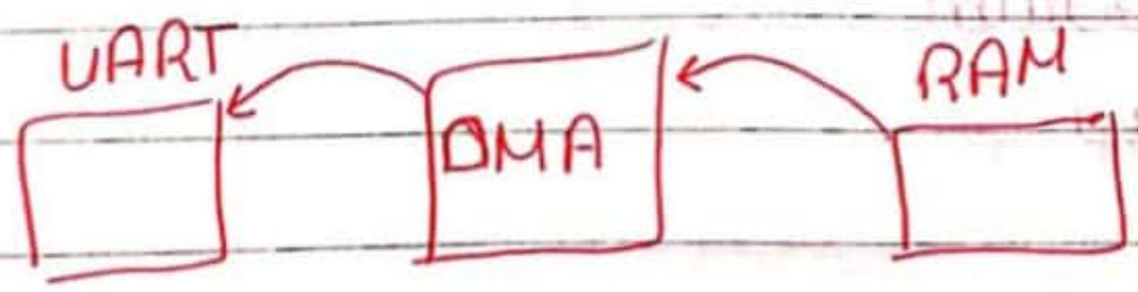
DMA Features :-

① → Save Time for The Processor
② → Faster in Copy

DMA "Hardware" :-



DMA Read Memory and writes another Memory in The Same Clock Cycle



at Same Clock Cycle.

Memory Size :-

Memory size :-

Source size	destination size	State
same	same	normal
16bit (2B)	8bit (1byte)	Data alignment (LSB)
8bit (1B)	16bit (2B) (LSB)	

→ Coding driver

At data sheet

* Register description.Pdf

[13] DMA → registers

⇒ 7 Channels

DMA interrupt status فال [3] flags ليها channel كل register

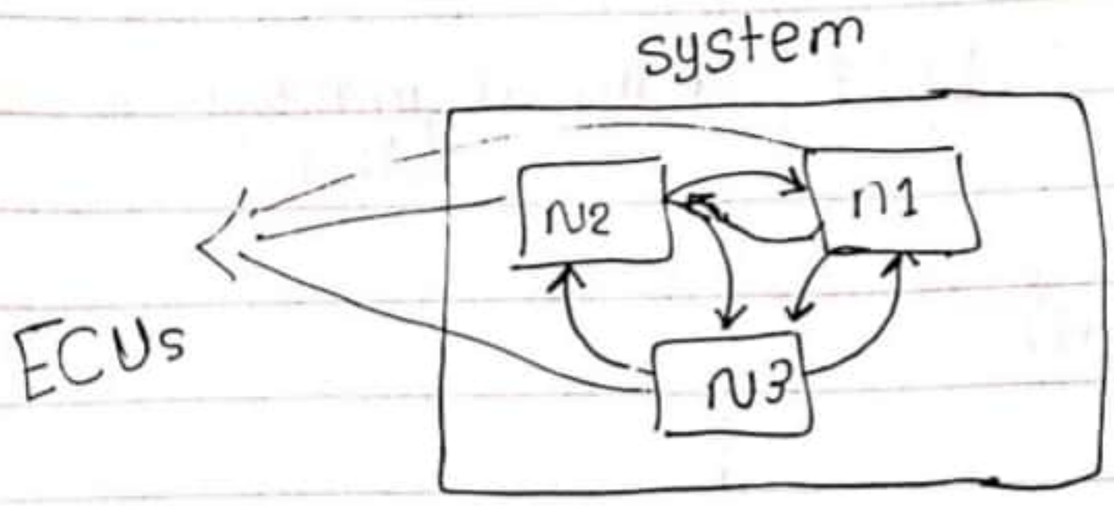
GIF, GTIF, GCIF, ~~GIF~~

وليها flag بيعمل (ناتج) OR بينهم

الـ flags دي Read only

Communication :

يعني عندنا system واحد قررنا نقسمه على كذ MCs



ECU (Node) : Electronic Control Unit

يعني board عليها Micro Controller وفي السوق اسمها كارته (ECU)

ماكل جزء هيعمل جزء من المطلوب فهيحتاجه يتواصلوا مع بعض

⇐ ايه الهدف من التقسيم ؟ ايه السبب ؟

1) Complexity ↓

هيبقى سهل في الـ debugging وبكدا هتقل الـ Complex

2) Reliability ↑

معناه إن الـ system بتاعي يـ behave زي ما مطلوب لفترة أطول

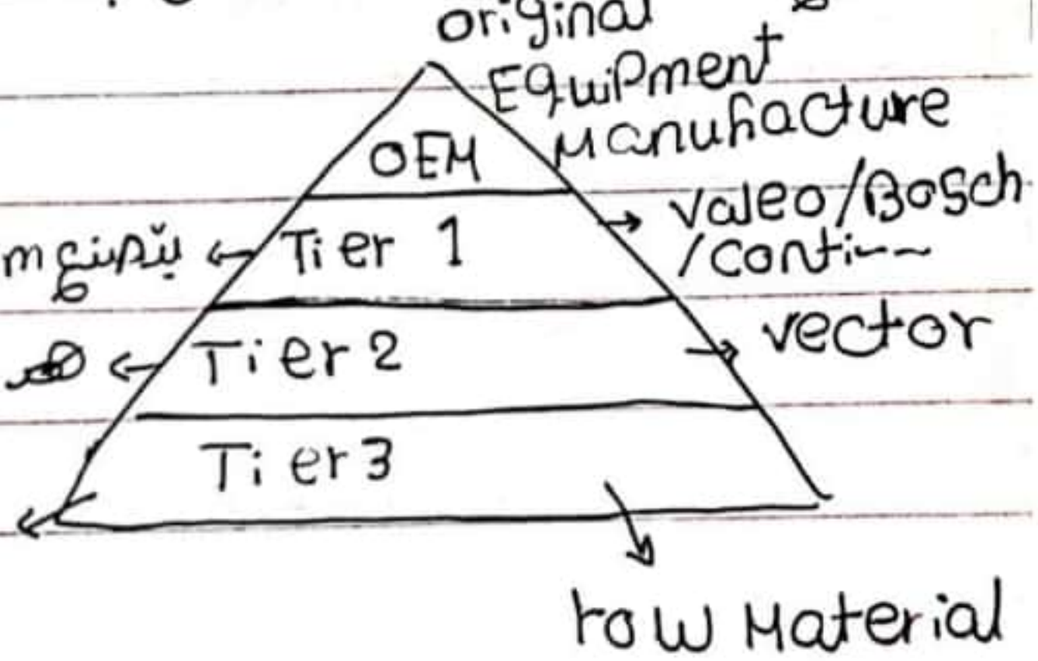
3) Dependancy ↓

بيصنع بالكامل system زي الكشافات مثلاً" والموايش

هو بيصنع حتة من الـ system بيديها لـ T.1 يكمل بيها الـ system

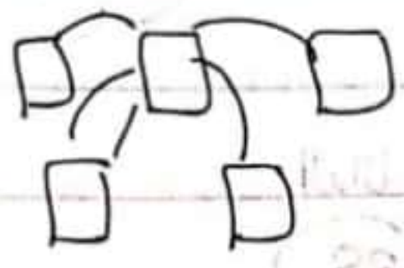
زي الـ Compilers والـ debugger

بيصنعوا مثلا المايكروكنترولر، زي البلاستيك أو الفتيل أو الـ PCB



Conveitional network

كل node تكلم التانيه



فوصلو فالأفضل هى طريقة ال

* Communication Protocol

وهى طريقة توصيل معينة

لأى Protocol //

- Features (سرعه ، عدد النود ، المسافات ، ...)
- Hardware interface (شكل ال Conect بين كل نود والتانية)
- Data frame format

Standard VS Protocol

Standard	Protocol
يعنى Protocol كل الناس بتفهمه بنفس الطريقة فى العالم	يعنى طريقة معينة ومحدش يعرفه غير اللى الروم اللى (تخيل أسهل بروتوكول :..)
مثال: I2C, CAN, LIN	مثال: UART

* Communication Protocols Specifications:

⇐ 7 حاجات

1. Communication Medium

الداتا هتتبعت إما بشكل (من نود إلى نود)

	wired	wireless
1. Speed	~	~
2. Cost Maintenace & IAS (صيانة)	↑ غالي	↓ ① Point مش بياخد وقت في تصليحه
3. Mobility (حركة)	↓	↑ ②
4. Safety (صحة الانسان)	↑ ①	↓
5. Security (يعني ضد ال Hack)	↑ ①	↓

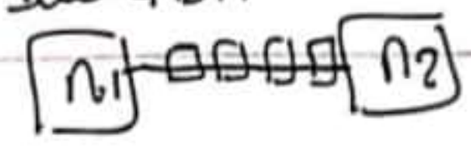
⇒ Security (بالنسبة للداتا)

يعني هو System بيكون Secure إن حد يعمل hack عليه ، يعني ال data ماشية بأمان

⇒ Safety (يعني أمان بالنسبة للإنسان)

2. Data Transfer mode

يعني الداتا هتتبعت ازاي.

Serial Communication	Parallel
مثلا 4 bit	
	
	at The Same Time

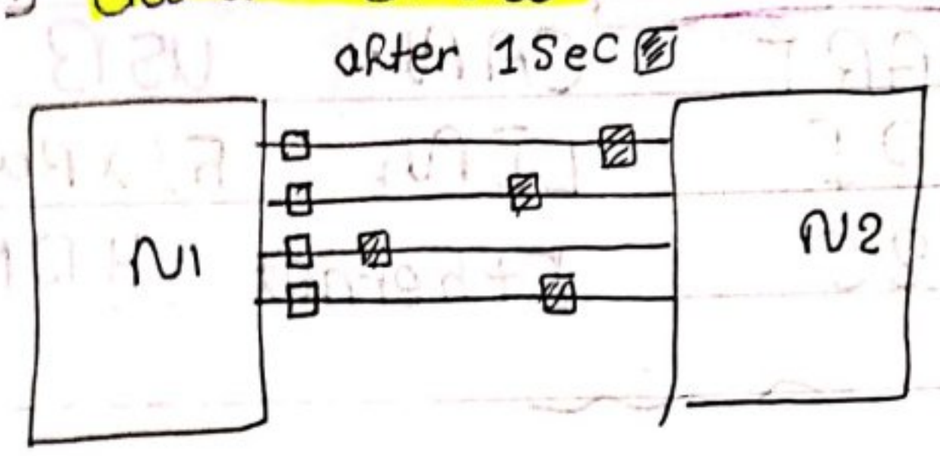
ex; 1 bit ⇒ 1sec

Serial
* 9sec

Parallel
* 1 sec

* Parallel com. Problems

1 data skew

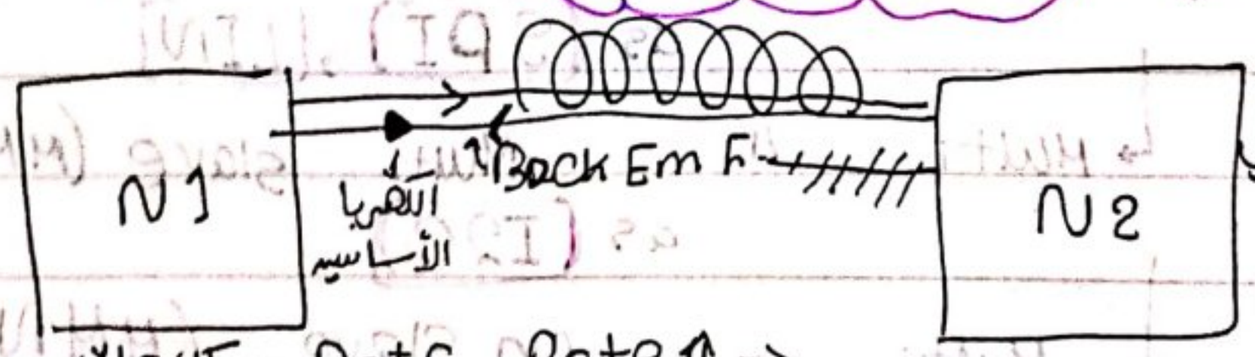


skew

حصلت المشكلة دى لأن الأسلاك دى مقلب مش فى نفس
ال connectivity بسبب عوامل طبيعيه مثلاً
فمش هينفع بعد 1sec أقرأ لأن الداتا كلها مش
هتكون وصلت فمحتاج أستنى وقت أبطء واحده
وده مش كويس (مش rate).

* طيب هنحط ايه limit لـ data Rate

2 Problem ⇒ Interference



أى سلك ماشى فيها كهربا بينشأ عنها مجال مغناطيسي وأى سلك جوه المجال بينشأ فيها تيار عكسى

كلما نزيد ← Data Rate ⇑ ⇒

Interferance ⇒ هتزيد

[3] Complex ↑

[4] higher Cost ↑

يعني في عالم ال embedded system ال Parallel اختبار فاشل

⇒ **Serial**

- UART
- SPI
- I2C

CAN ✓ USB
LIN ✓ F/XRay
Ethernet HDMI

[3] N2N Relation (Node & Node) (Separation)

علاقة ال Node بال Node تانية

[1] Peer To Peer أي حد بتكلم في أي وقت

عندهم نفس القوة في ال Power

[2] Master Slave

يعني ال Master هو ال بيتكلم وال Slave لا

إلا لما ال Master معين يقوله بتكلم

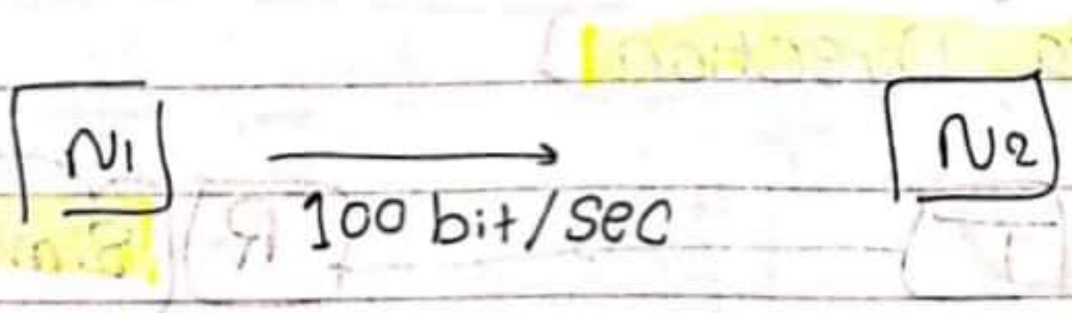
- Single Master Single Slave (SMSS)
- single " Multi " (SMMS) as (SPI), (LIN) — واحد بتكلم والتاني بعدين
- Multi " Multi slave (MMMS) as (I2C) — يعني شرط ال base يكون فاضي
- Multi " No Slave (MMNS) as (CAN)

هامة جداً

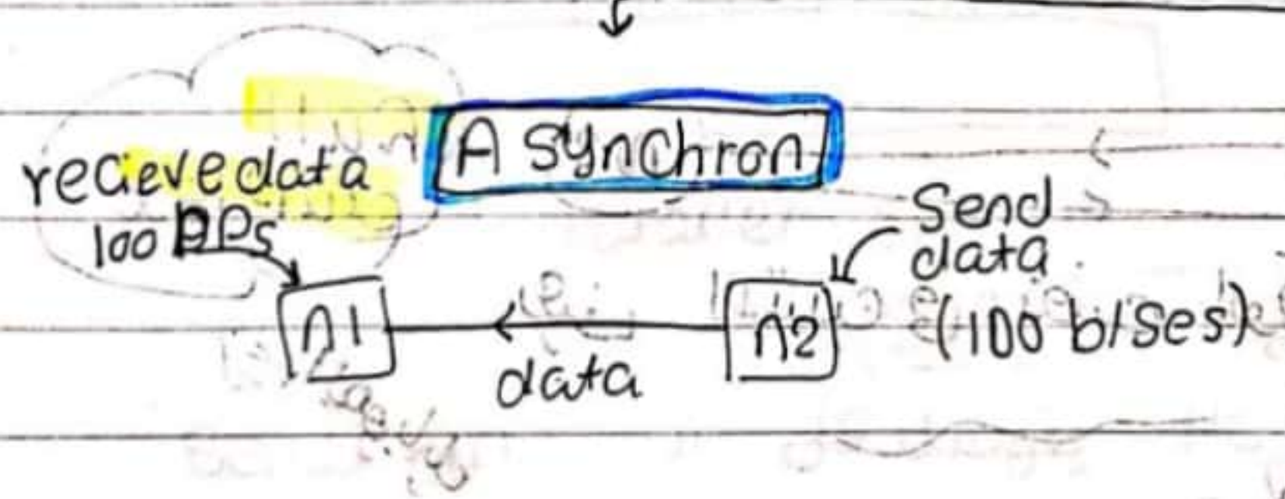
9 Synchronization

يعني معناها تزامن (يعني احنا الاتنين عندنا نفس الزمن)
يعني لو قولنا Two nodes بيكونوا SYNC
يعني هما الاثنين شايفين نفس الزمن

*

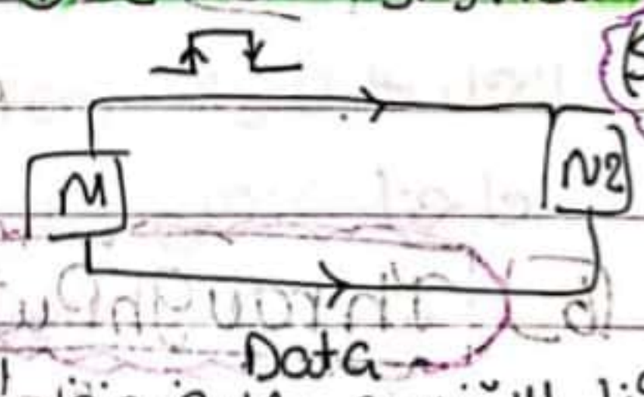


لازم N2 تستلم في نفس الزمن (نفس السرعة بتاع N1)
علشان يفهموا بعض صح
وده شرط في ال Communication.
* طيب دا عملت نعمله من خلال حاجة بطريقتين



Synchron

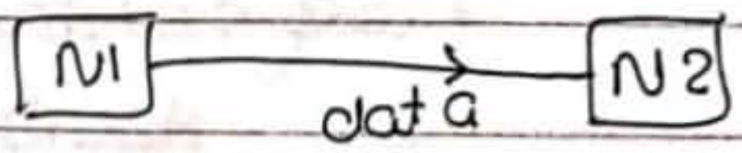
① clock signal



(SPI , I2C)

(هنا احنا الاتنين شايفين نفس ال clock.)
send → raising edge
recieve → falling edge
فكدا كدا هنفهم بعض

② Frame Synchroization



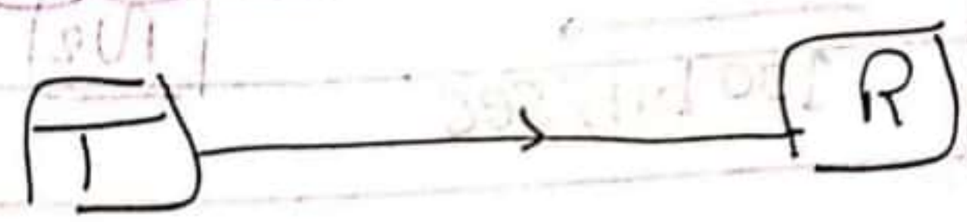
هيبعت 2 Pulse قبل
ما يبعت ال data
as (LIN)
سلك واحدة و sync

ال Sync فيه *

* Dynamic Data Rate ....!

لأن N1 ممكن أبعت بسرعة مختلفة

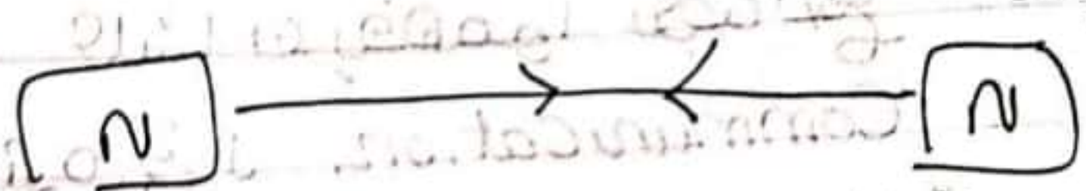
5 Data Direction



Simplex

يعني data مش رايحة في اتجاه واحد بس

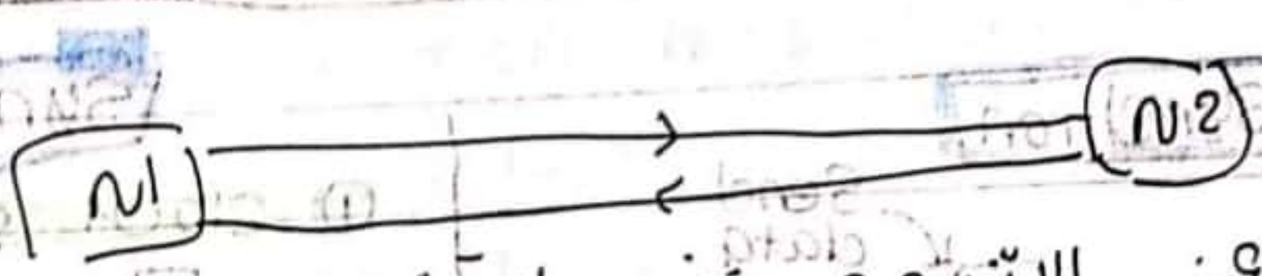
or



Half duplex

الاتنين بيتكلموا و بيسمعوا بس مش في نفس الوقت.

or



Full duplex

يعني الاتنين في نفس الوقت

زي الموبايل

6 Throughput

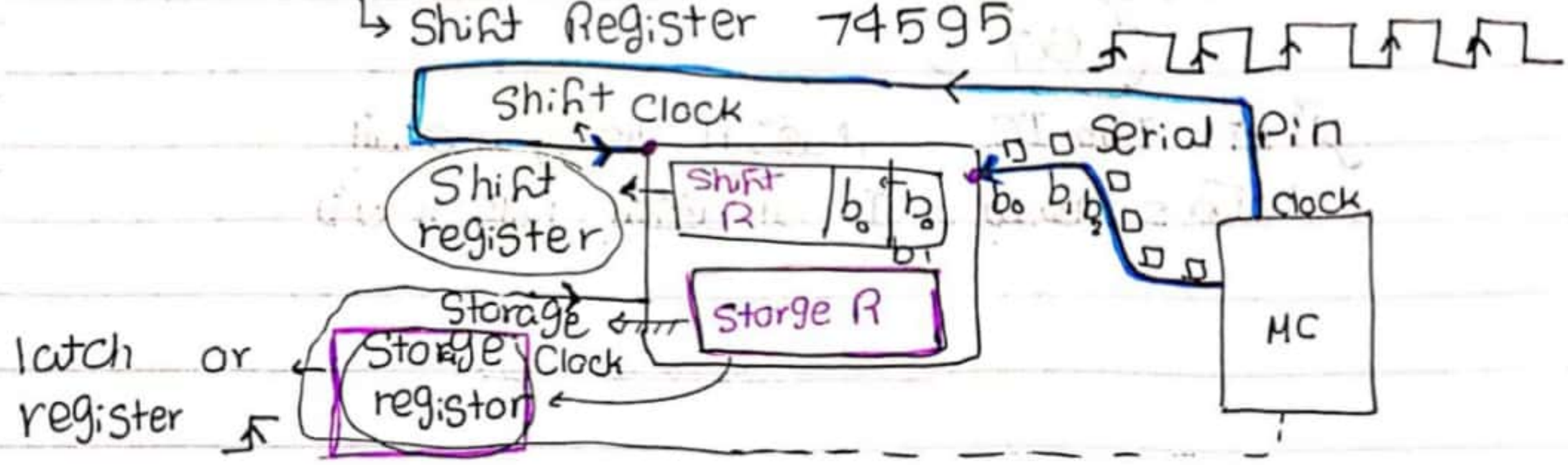
نسبة مئوية بتعبر عن قد ايه من الداتا مهم

$$\frac{\text{Data bits}}{\text{Frame Size}} \%$$

ازاى لو عندك System ← Parallel أخليه Serial

Parallel To Serial Conversion

↳ Shift Register 74595



وعندنا MC عايز يكلم حاجه ، الحاجه دى Parallel

زى مثلا ال Leds

Parallel system

الحل ده هو ال Shift register دا .

فاحنا هندخل ال MC يتوصل على ال Serial Pin

بتاع ال Register دا ويبعت bit by bit

وسلك ال Clock بتاع MC راحه لل Shift clock

بتاع ال register

ومع كل raising edge هيبعتها ال MC لل Shift Clock

هيبدأ ال IC دا يشوف ايه ال موجود على Pin ليه *

ويدخلها عنده فى ال Shift register

Slave

، بمجرد ما يتملى ، هيقوم ال MC يبعت one

ل Storage Clock (واحدة بس)

SS then Pulse recieved on Storage Clock :

رأى ① Data Copied from Shiftregister

To Storage Reg.

② if outPut,Enable , data on storage register Transfer to Physical Pins.

Q7

بنستخدم في حالة استخدام Two IC أو أكثر
لأن الـ enable بينتقل لـ IC الـ بعدها وهكذا

lec 19

Communications

SPI

* TFT
* SD card

1) Serial Peripheral Interface :

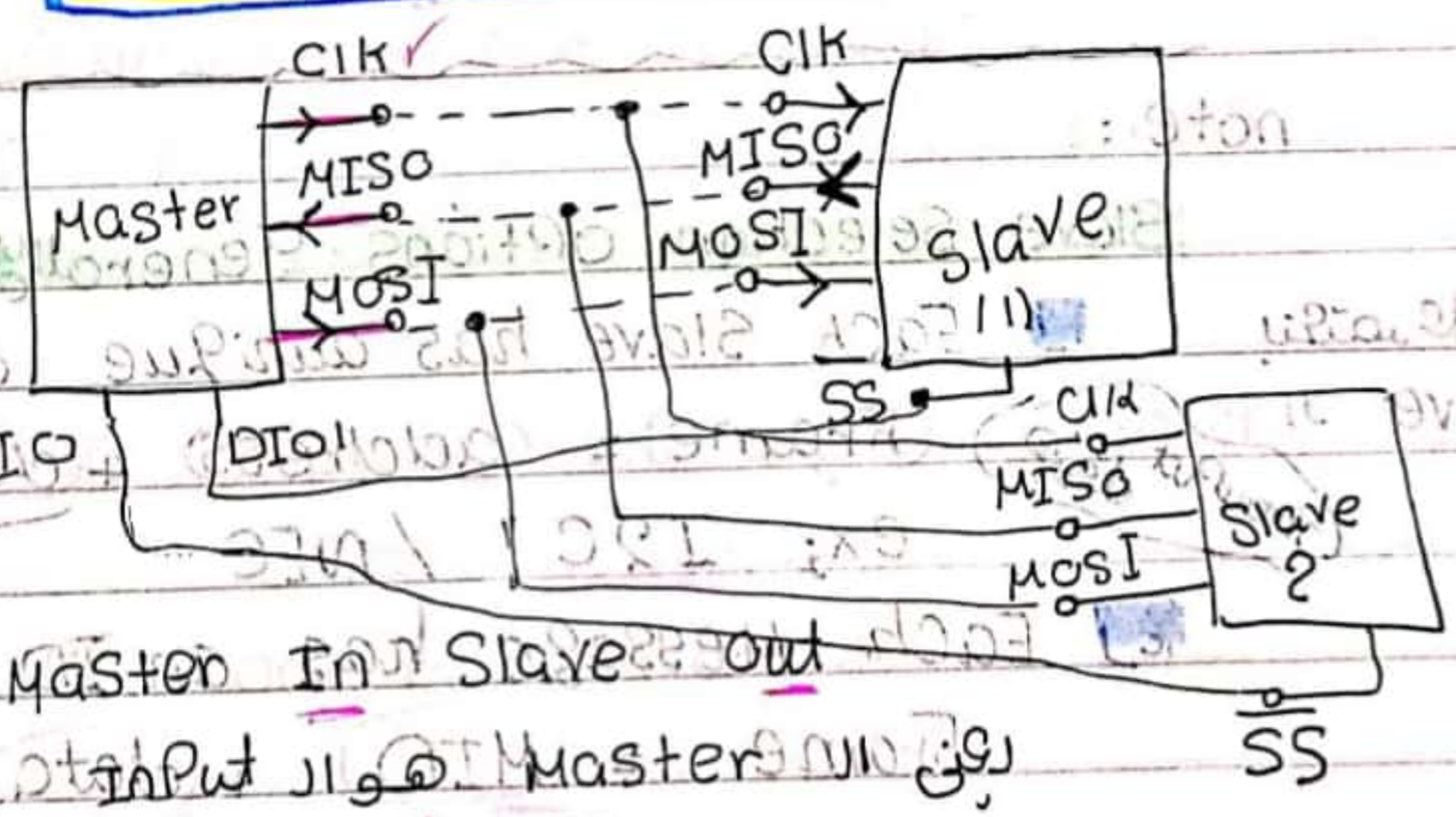
→ 1) Specifications :

1) Serial
2) wired
3) Synchronus (Shared clock wire)
4) SMMS
5) Full duplex
6) Throughput %

2) Hardware Connection شكل ال nodes

في طريقتين

① * **Independent Slaves**



* MISO : Master In Slave Out

يعني ال Master هو ال input

* MOSI : Master Out Slave In

وده السلك اللي بيبعت عليها data لل Slave

⇐ ال Master في ال SPI بيحتاج ٣ سلكات

SS : Slave Select (active low)

ومن خلالها ال Master بيختار أنى Slave هيكلمه

* مينفعش في ال SPI ال Master يكلم كذا Slave في نفس الوقت ، لأنه Full duplex يعني بيكلم ال Slave وبيرد عليه في نفس الوقت فكذا سلك MISO متقابلها مشكلة لأن وارد يكون السلكين المتوصلين بيها بيبعتوا 1,0 .

Advan...
1) Simple
2) Direct

Dis ...
1) No broadcasting
2) Number of slaves (Max) dependent on number of DIO Pins @ master.

note :

Slave Selection options generally :

بنعتمد على إن ال Slave ده

1) Each Slave has unique address
Frame = address + data → توفير hardware
ex; I2C / NEC
Throut هيقل

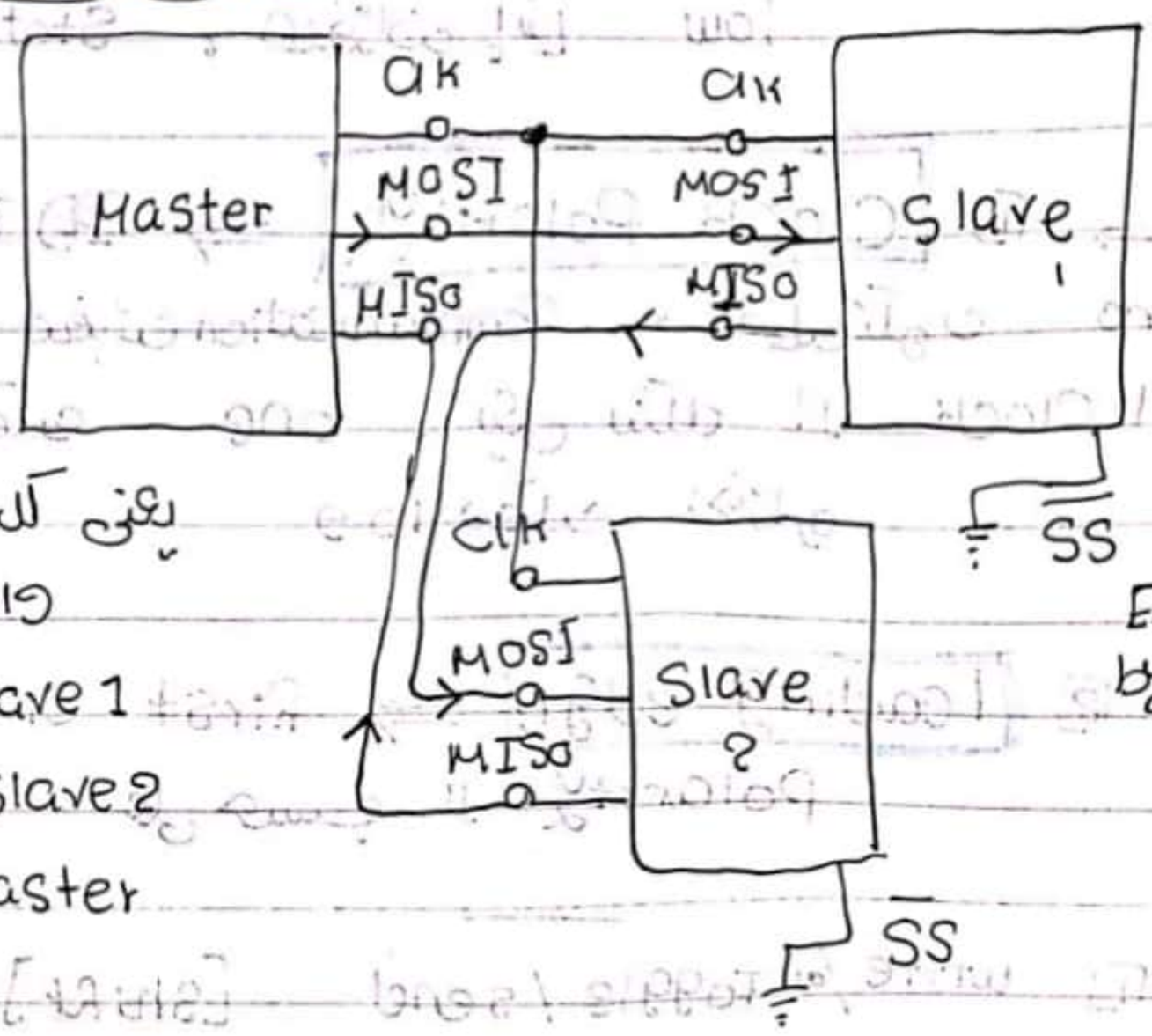
2) Each Message has an ID :
Frame = MID + data
ex; CAN / LIN

3) Hardware selection → Chip select
→ Slave select
Hardware Pin for each slave

Throut أعلى وسرعة أعلى

no broadcasting
Hard ware استخدام

[2] * Daisy Chain — الطريقة التانية لـ SPI



Enable by default

يعني كل واحد بيكلم node واحدة بس

Master → Slave 1

Slave1 → Slave2

Slave2 → Master

Adv

[1] unlimited num of Slaves

[2] Master Consume only 3 lines

Dis

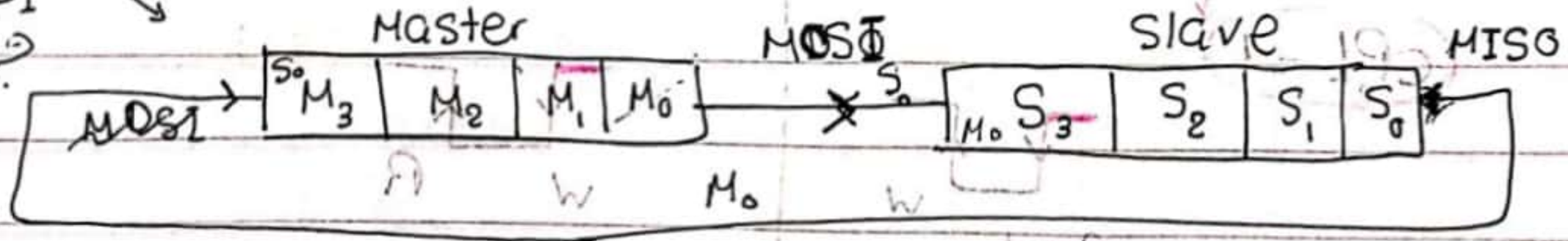
[1] Transfer Time depends on the Slave Position

[2] one Point failire

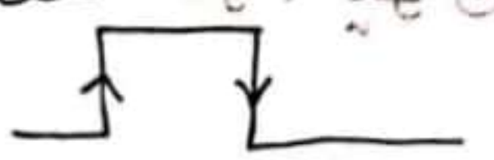
[3] Frame Format

ريجستر بتاع SPI

Master



Clock (output of Master)



Clock

في الأول ايه State ، هنفترض إنها low

→ ① **Clock Polarity** ⇒ ID late State of Clock

الحالة ال هو مفيش Communication ، ممكن تكون Zero و ممكن one ، يعني شكل ال Clock ايه وده ممكن نختاره

→ ② **leading edge** ⇒ first clock edge ↑ or ↓ على حسب ال Polarity

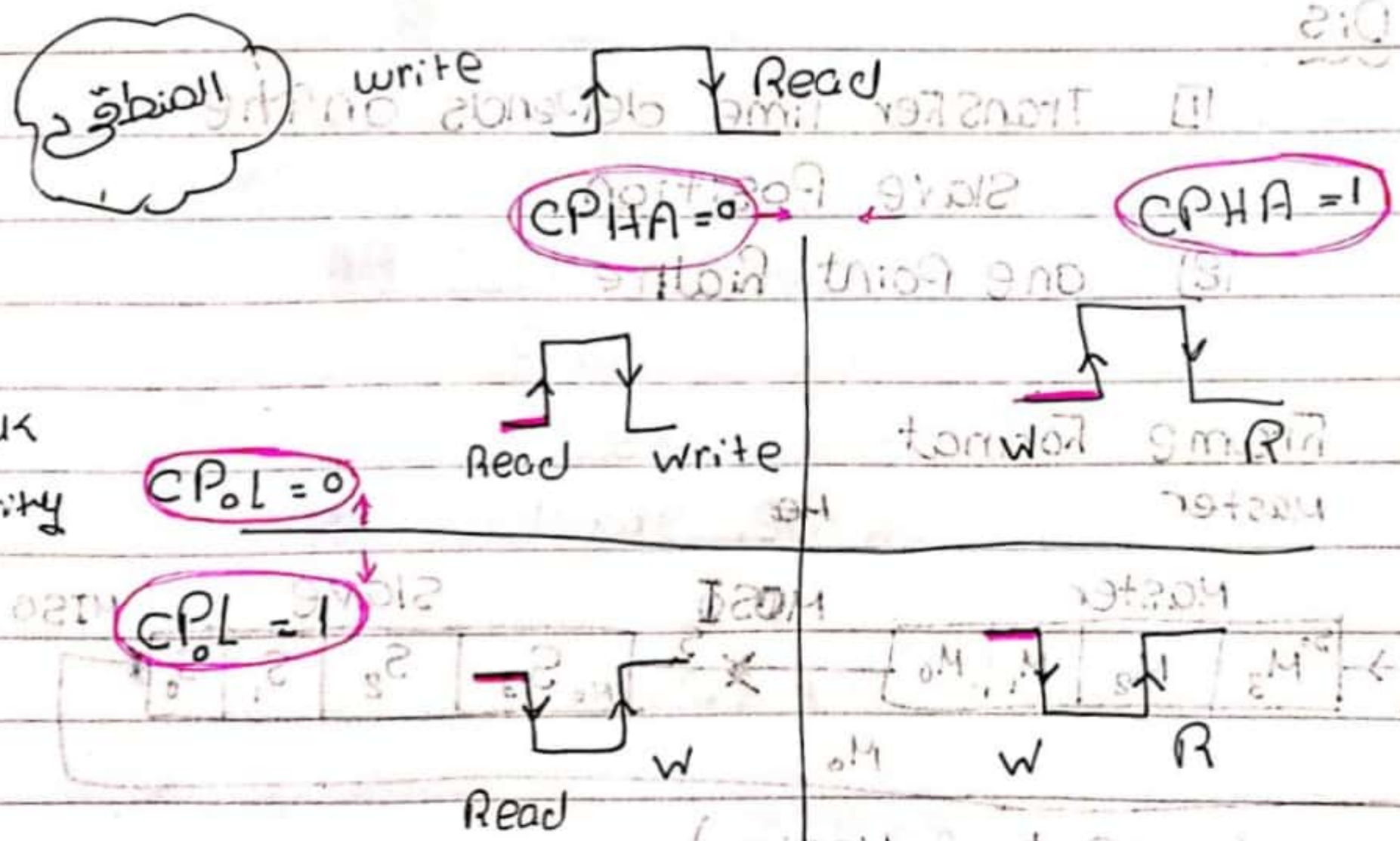
---

[1] write / or Toggle / send [Shift]

[2] Read [from Slave]

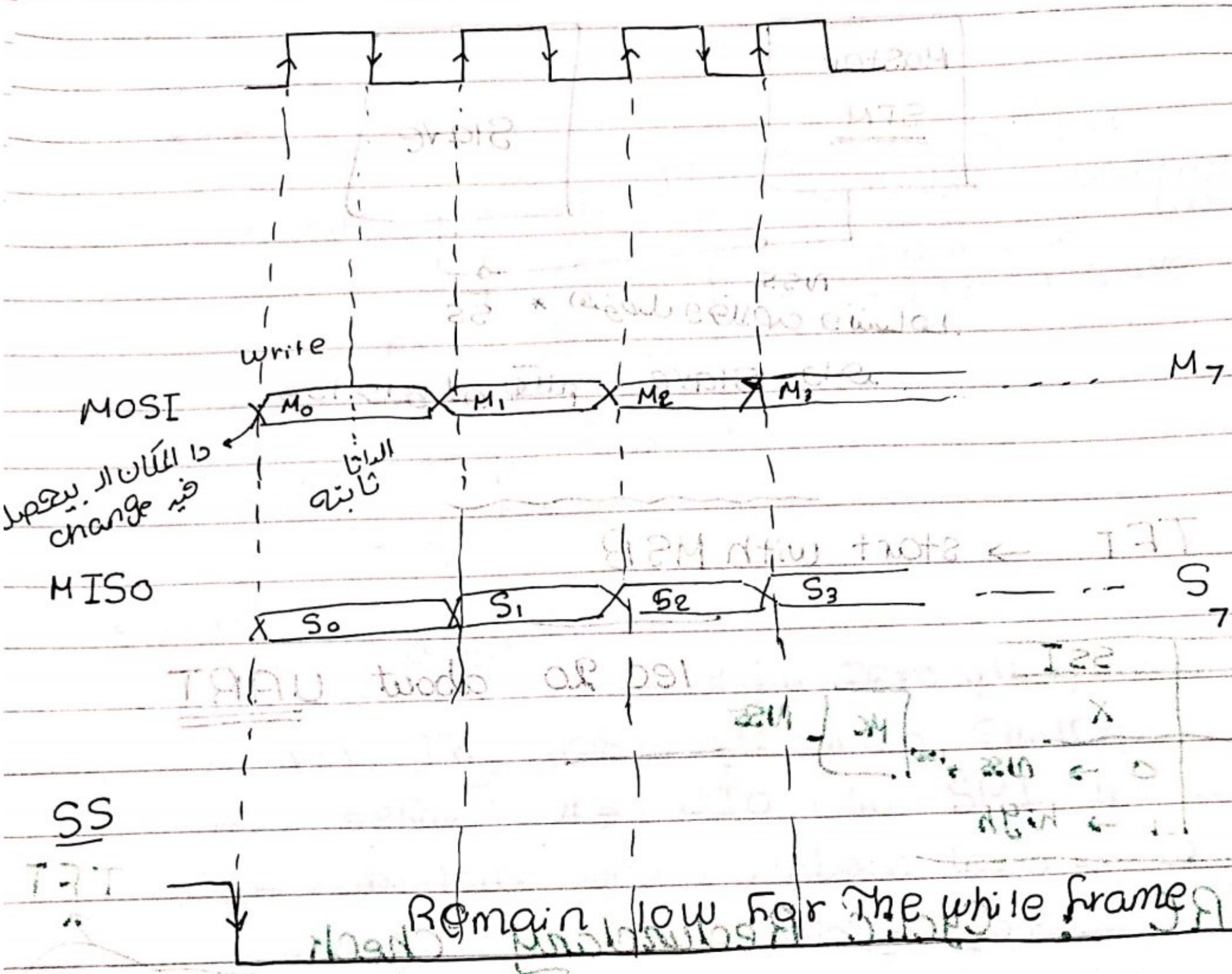
→ ③ **Clock Phase :**

what to do first @ leading edge.



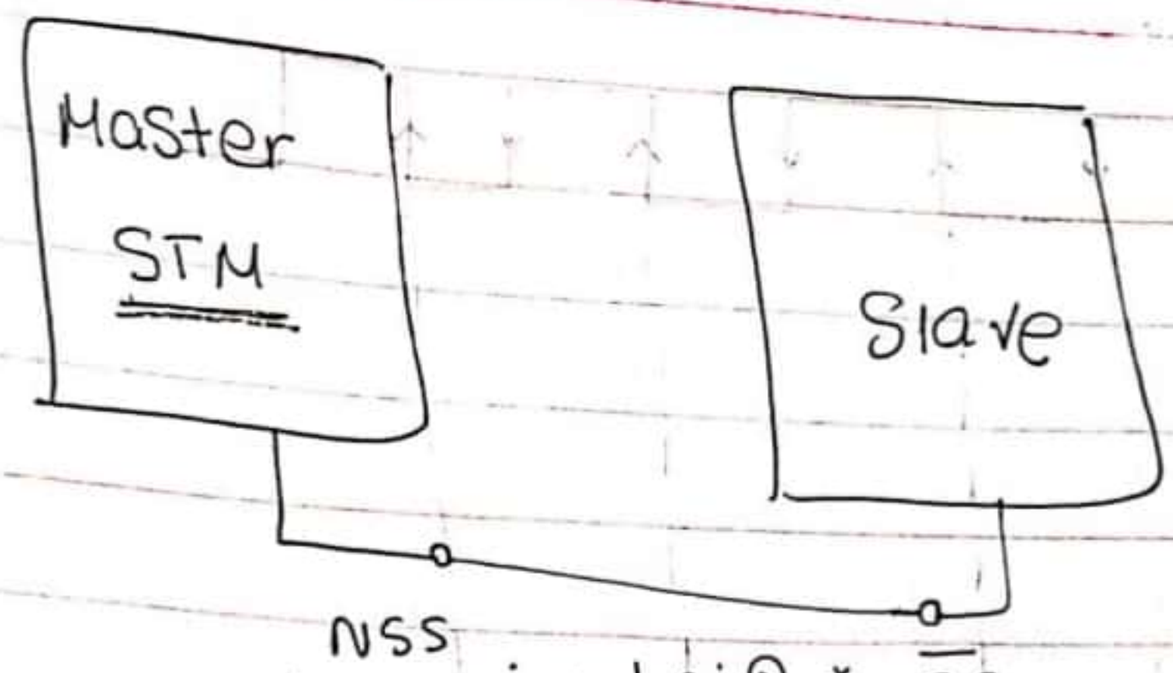
Slave وال Master لازم يكون ال

Phase نفس ال



Configurations:

1) Clock Polarity
2) Clock Phase
3) Frequancy
4) Master / Slave
5) Data Size (8 or 16)
6) Data order (MSB or LSB)



* هنوصل وخلاص وننساها.

وده حلو لو هتلم Slave واحد

TFT → start with MSB

	SSH	SSI
W	0	X
	1	0 → NSS → low 1 → high

lec 20 about UART

MC → NSS

TFT

CRC : Cyclic Redundancy Check

MSB

LSB : 0

1 LSB

SSM : Slave select Mode

0 → H.W

1 → SW Portable

or

Gnd or DIO Pins

SSI : هي قيمة ال Slave Select في حالة انها

Controling ~~H.W~~ SW

0 3 4 7

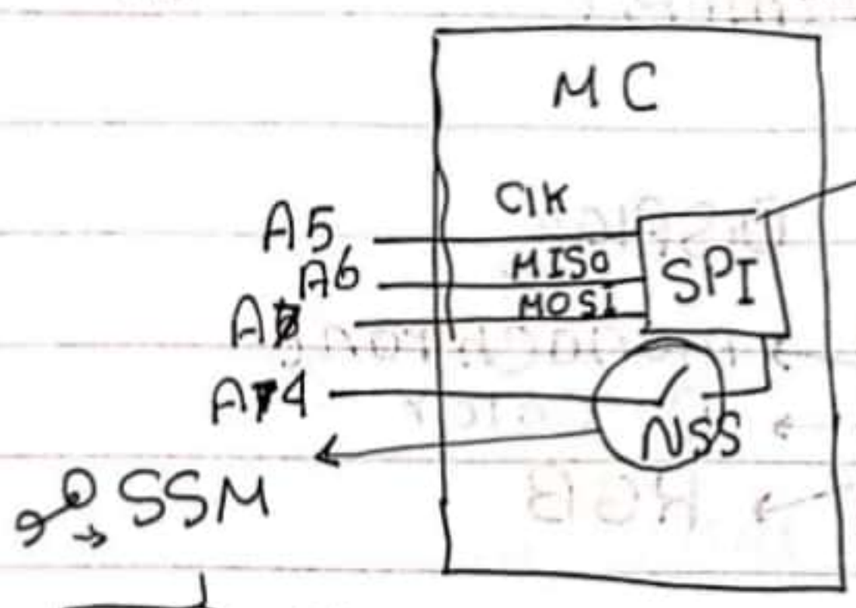
0000 0011 0100 0111

Lec 21

TFT

* SPI

↳ SSM / SSI → 0 ⇒ slave selected
→ 1 ⇒ slave not selected (master)



مملت يشتغل
Master or Slave

لو خليناها بـ zero يبقى الـ Switch مناسب عليه وطالعلي برو Pin وممجوزه للـ SPI مش DIO ، وال default بتاعها A9 فاحنا دايما هنحطها بـ one علشان مش عايزين نستخدمها في حاجة.

كدا ال NSS قيمتها مش طالعة برو ، وقيمتها هنا على حسب قيمة ال SSI فهي.

، لو احنا شغالين slave فهتبقى SSI بـ high
لازم Master

علشان (active Low) NSS

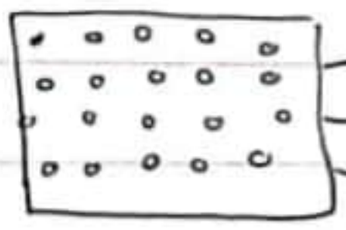
as Master

SSM	SSI	
0	0	NSS Pin is activated on A9
0	1	وراح على Slave
1	0	not working in Master
1	1	NSS not connected on A9 → Master Activated

Display

1) Segment Display.
   - ↳ Segment Display
   - ↳ Alpha numeric

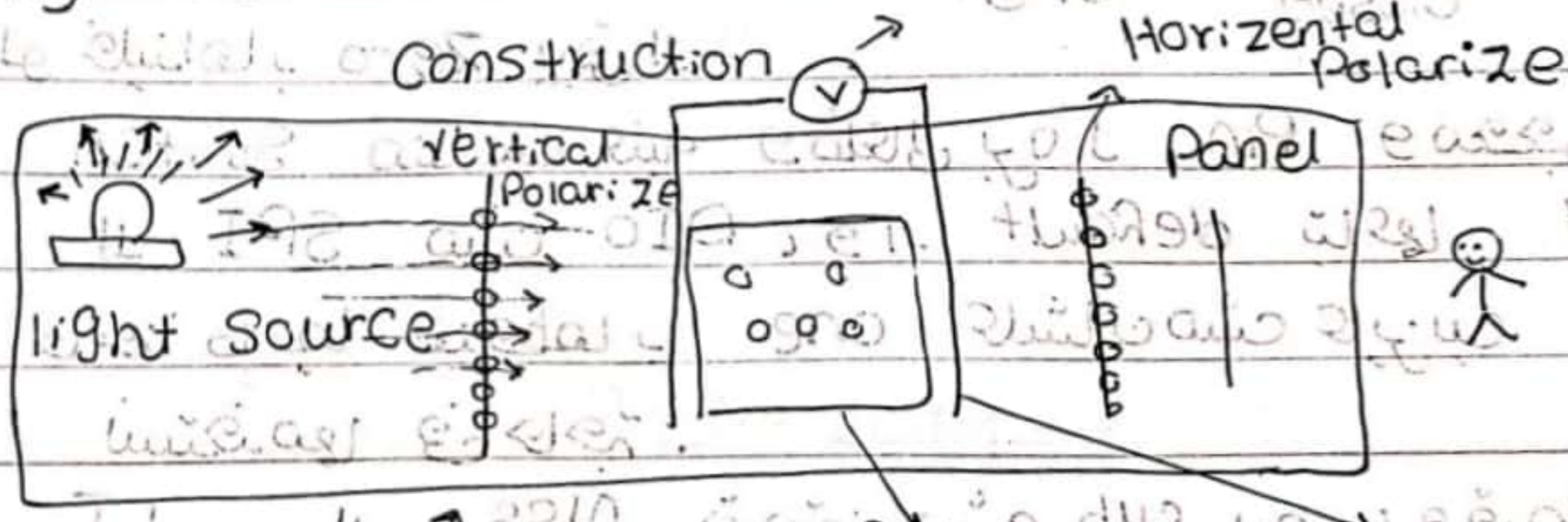
2) LED Matrix Display



→ monochrone
→ Bi Color
→ RGB

3) LCD

Construction



عبارة عن سائل فيه جزيئات لمادة اسمها crystal (طبيعية)،

عبارة عن System كل لون يوحد اتجاه اللون الخارج

لما يعملوا عليها apply volt

تبدأ تغير من وضعها يعني تبقى ▭ أو ▯

فممكن توجب الضوء أو يعدي من خلالها

الحقيقة إنها بتخلي الضوء يعدي من خلالها (زي ما هو)

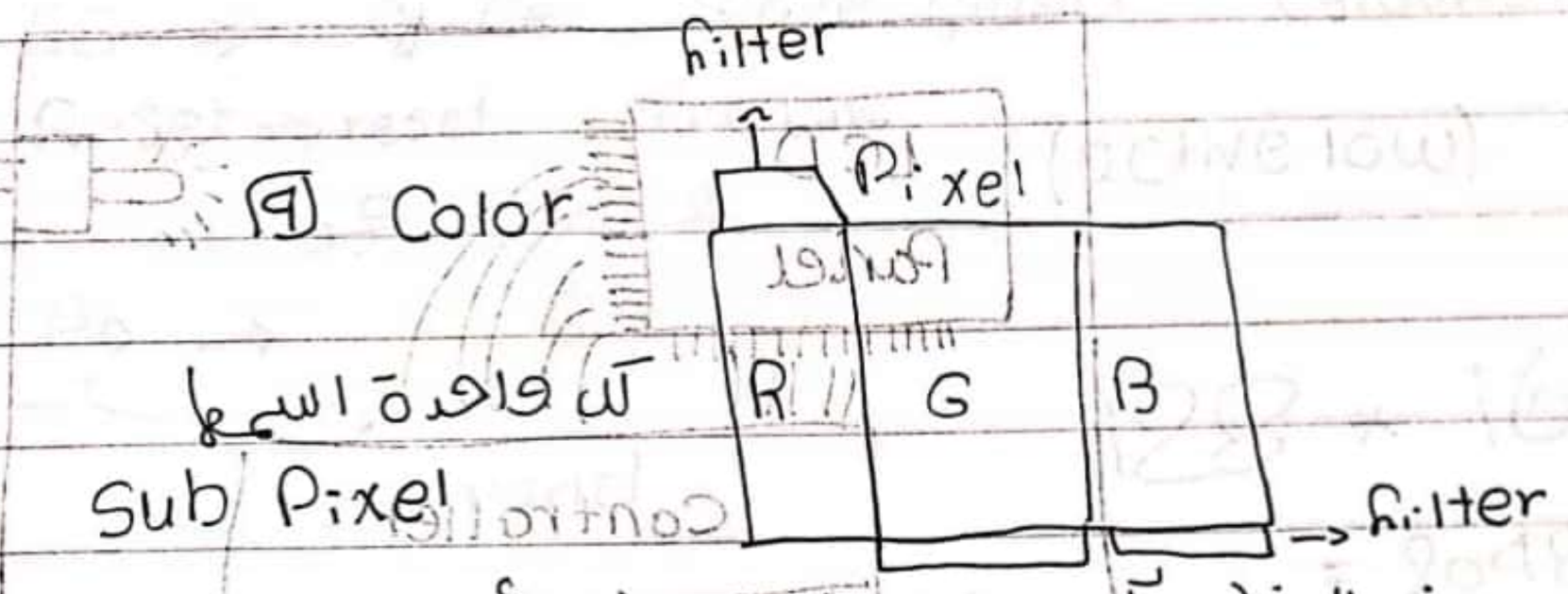
يا بتقلبه 90° درجة.

Light Source → Large
→ LED

② Contrast التباين (vs) Intensity

③ Thin film Transistor

④ Color



كل واحدة اسمها Sub Pixel

أي لون في الدنيا يمكن تحويله لـ ٣ ألوان دول بس بنسب مختلفة (Red, Green, Blue) لو كلهم صفر يبقى اسود لو كلهم 1 يبقى أبيض

⑨ Touch

علمياً مفيش حاجة اسمها Touch Screen (دا اسم تجاري)

الـ Touch هو عبارة عن Display أياً كان نوعها (LCD, TFT)

بنحط عليها حاجة اسمها Touch Panel وهو عبارة عن لوح شفاف مصمم فيزكلي بشكل معين

Display + Touch Panel → Resistive
→ Capactive

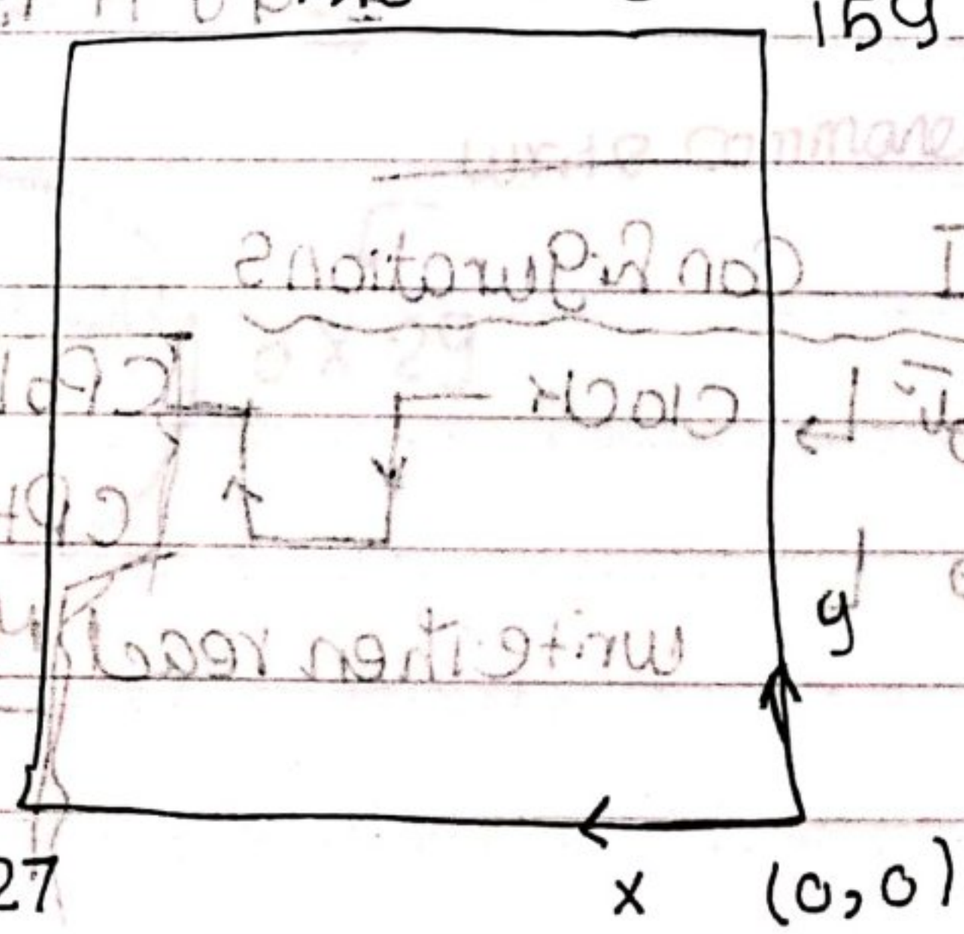
الشاشة تحتها microcontroller
من شركة SItronix
واسمه ST7735S

وال Pins الخارجة دى خاصة بـ ST وملهاش
علاقة بال panel

Pin → SCK ⇒ Serial Clock
خاص بـ SPI ، هنوصلها على A5
→ SDA ⇒ Serial Data (MOSI) (A7)
→ CS ⇒ ~~Ce~~ Slave Select (Ground)
→ Reset ⇒ reset علشان أعمل
للـ IC الرجوا (active low)
→ A0 ⇒ DIO Pin
Data — Command

128 * 160 = 20480

LCD = 1.8 inch

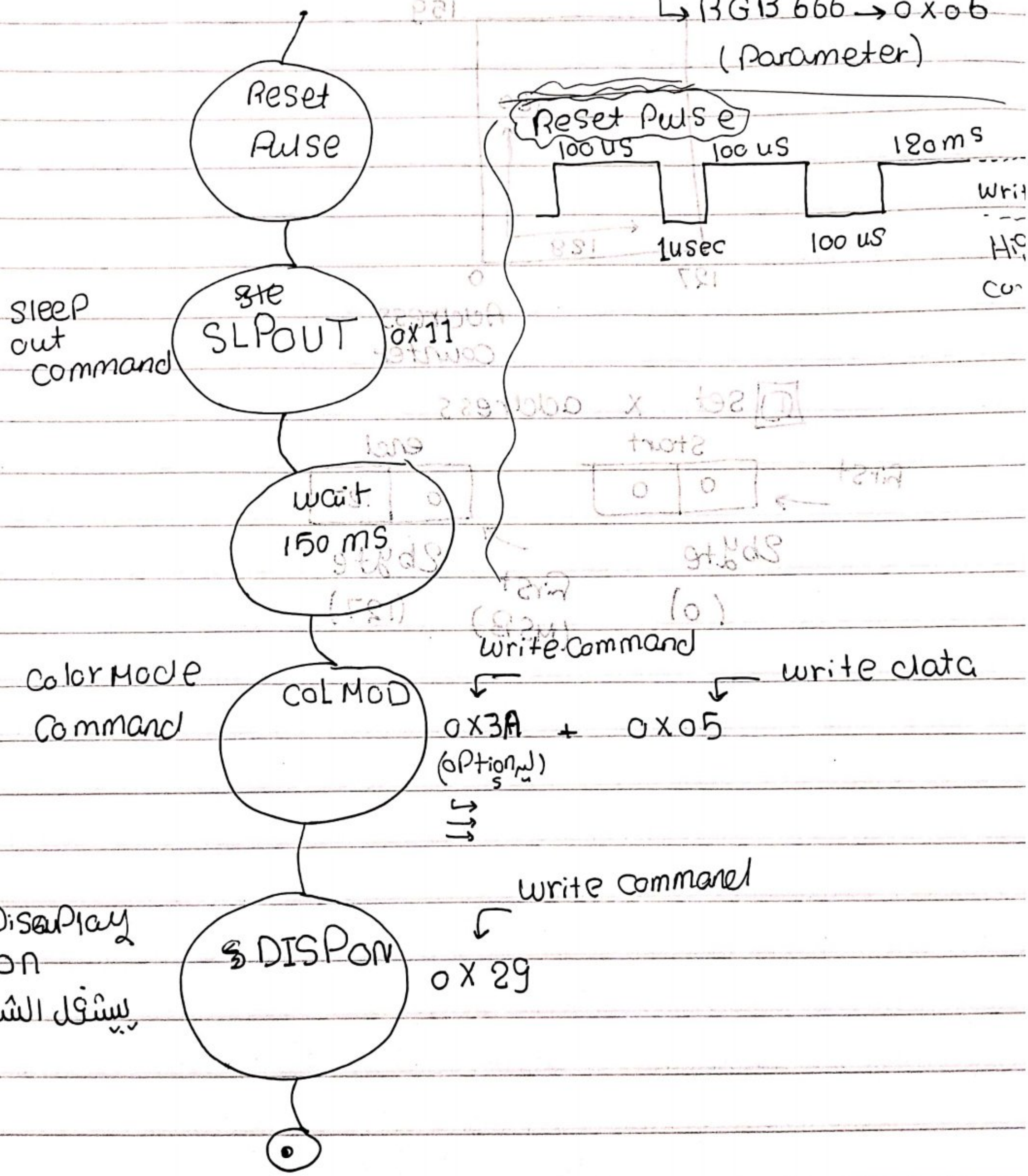


1 Pixel = R G

Color Mode
- ↳ RGB 444 → 0x03
- ↳ RGB 565 → 0x05
- ↳ RGB 666 → 0x06

(Parameter)

Init Sequence :



LeC 22

IOT

* web application

Networking

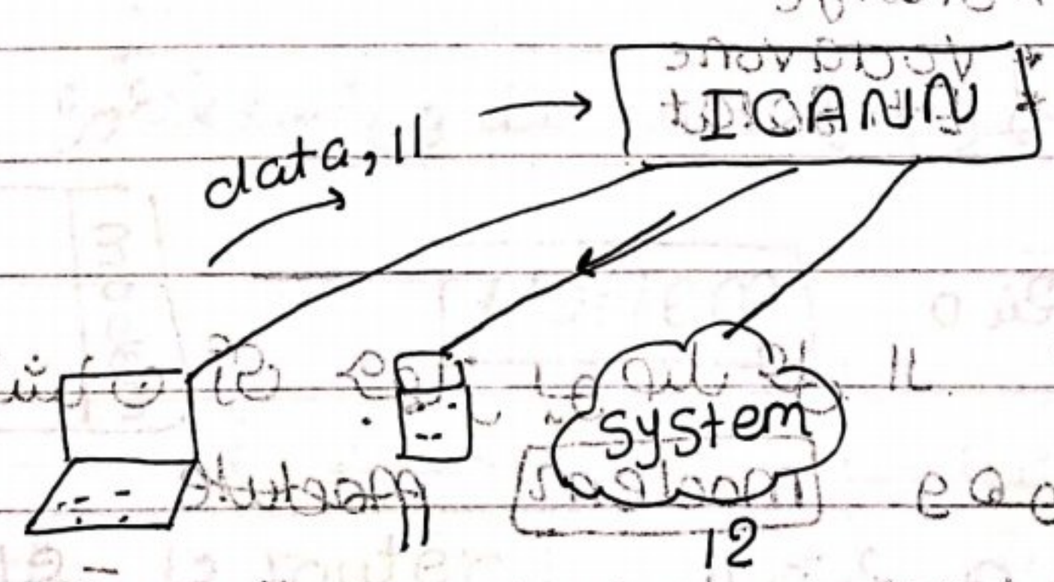
" Internet "

→ network

عبارة عن nodes مربوطة ببعضها وبتبعت ال data عن طريق ال Protocol معين

ICANN

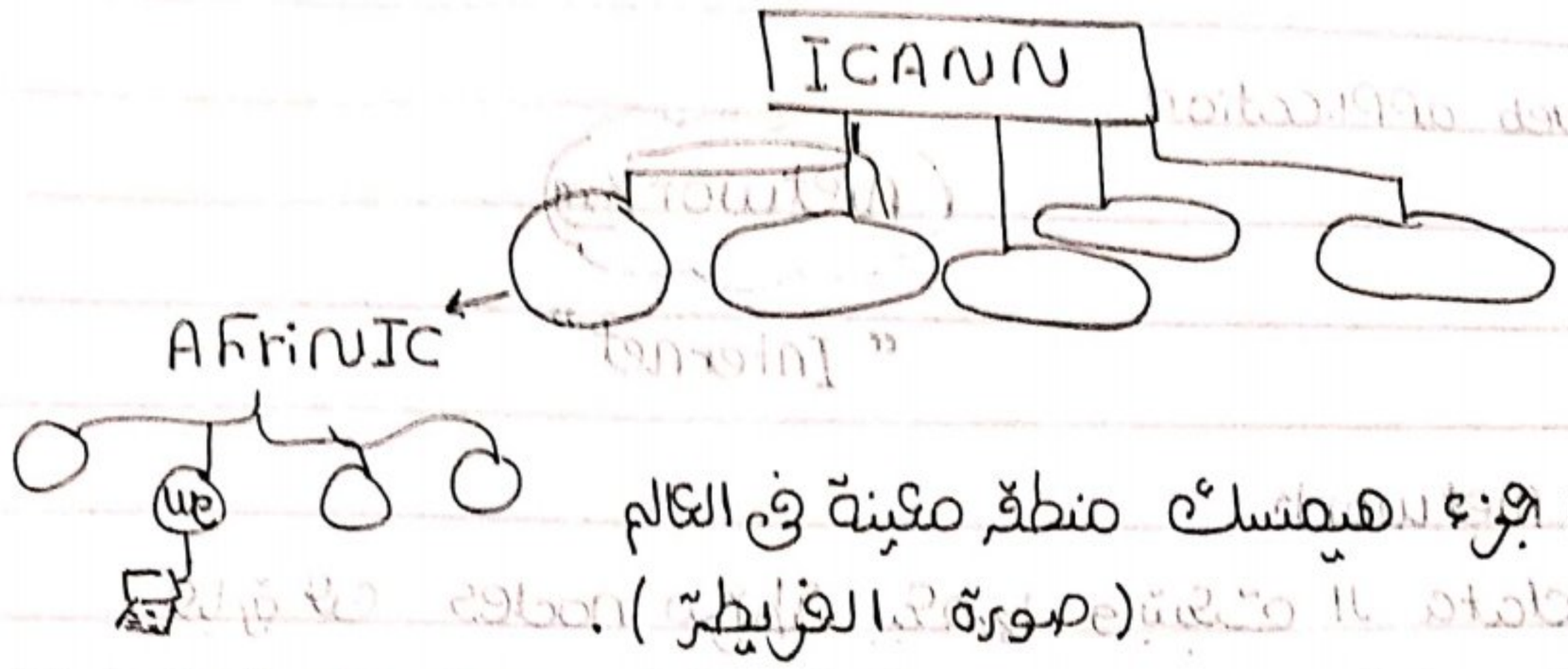
هي منظمة عالمية موجودة في أمريكا وهي المسئولة عن ال internet في العالم وأي جهاز واصل على الانترنت هو بأي طريقة متوصل على ال ICANN



المشكلة إن العالم كله متوصل على ال internet وكلهم المفروض صعب هنا بقى ال ICANN قالت إحنا هنجزء العالم بتاعنا لسبع أجزاء

ICANN ليها زي العالم كله IPs وده اللي بيوزع ال
وعددها 5

Regional ISP (Internet Service Provider)



كل جزء هيمسك منطقة معينة في العالم
(صورة الخريطة).

⇐ احنا مثلاً في مصر ال AfriNIC هي اللي بتتبع لـ ICANN

وبعدين كل جوا كل دولة فيه

local ISP (كل جوا بلد)
في شركات توفير خدمات ال internet

ex
* WE
* orange
* vodafone
* Etisalat

وعشان أي جهاز يوصل على ال internet ، لازم يكون
عنده modem وهو جهاز بياخد ال data
من الكمبيوتر ويرميها على Physical medium
عشان ينقلها على ISP بتاعك

modem
wired
wireless
(3G, 4G)

Telephone (بنستغل فكرة وجود التليفونات)

Each modem has IP → internet Protocol address

IP

هو عنوانك وهو unige ومحدش تاني ياخده

طيب ازاي هيكون ال IP بالعدد داكله وهو unige ؟

هو ال IP → 4 Byte

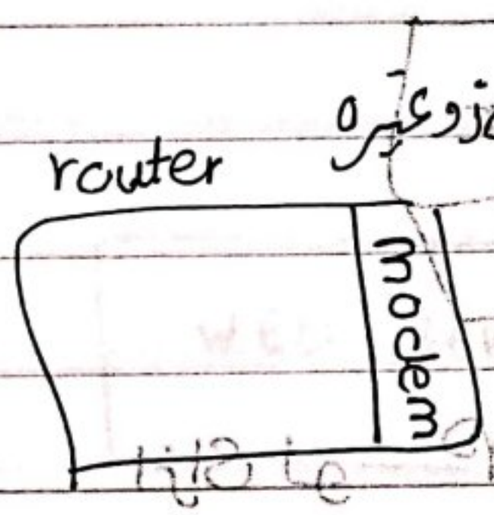
2^{32} = 4 مليار وشوية مثلا

والناس أساساً حوالي 7 مليار ؟

قالو هنعمل حاجة اسمها

router

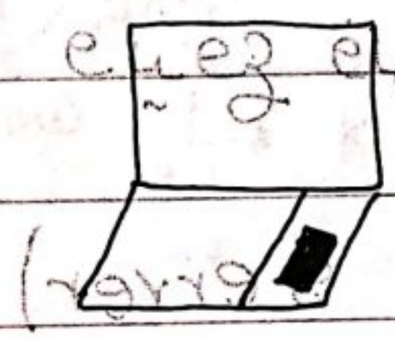
هو جهاز بيتحط في أي مكان مثلا في البيت زي غيره



Modem

ال router دا عنده

device هو هيبعت لل router فا الجهاز هيبقى فيه NIC



(Network Interface Controller)

wired — wireless (wifi)

ethernet كابل ال

LAN

B.B.B.B

0 : 255

ال router بقى بياخد IP وال device بياخد حاجه اسمها local IP محدش يعرفه غير ال متصل بنفس ال router

[2] حل تانى لتوفير ال IPs

Dynamic IP

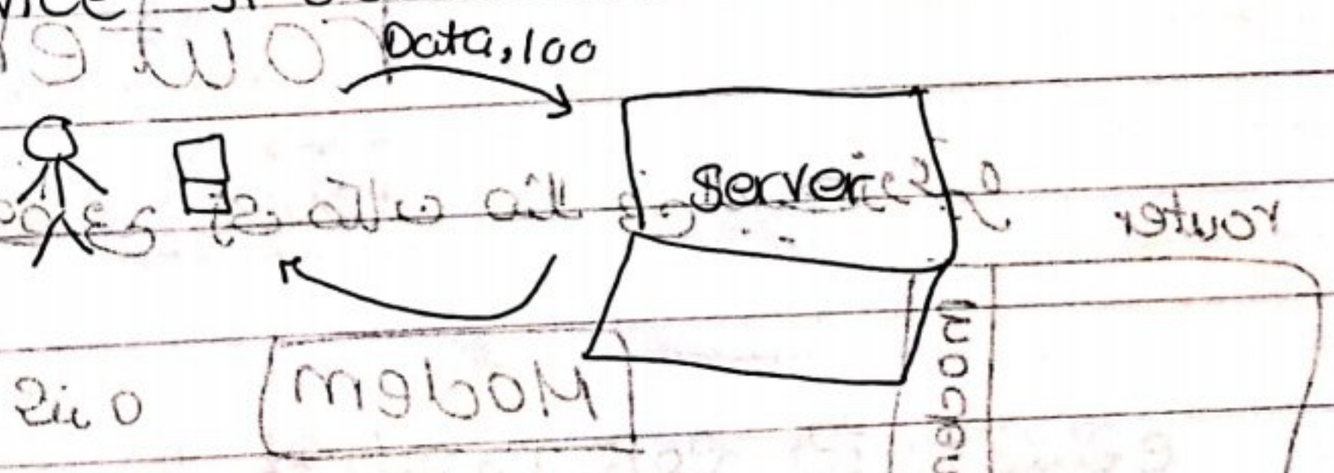
لو قفلنا ال router وفتحناه تانى هنلاقى ال router واخد IP Global تانى

Client Server

تشبه ال mod

ثم ال ~~websi~~ Server عبارة Combuter متوصل بال internet

و الانسان ال مكان ال device هو Client



ازاى لو IP دا Dynamic يعنى هيتغير هتواصل مكان ك Client

غيبه نلعب قالوا خلاص بعض ال nodes هنجيلهم Static IP

ويرفع فلوس ال ISP علشان ياخد Static دا.

DNS (Domain name Server)

Domain	IP
facebook	---

wired — wireless

(wifi) — Ethernet كابلات

Modem ثم ال router

أشهر الـ Protocols اللي هنتكلم بيها مع الـ Server

HTTP

وفي زي FTP

في عالم الـ web دا اسمه Front End الحاجة اللي بشوفها Front end

واللي على الـ Server هو Backend

Front end	Back end
HTML	PHP
CSS	Python
Js	Java
	MySQL

في Framework جاهزة

IoT

Internet of Things

web development + Embedded APP

(1) Framework جاهز

web من غير مكتب

↳ KAA

↳ ThingSpeak

حول بقلوس

↳ Microsoft Azure

↳ CISCO

↓

open Source

دا الهنشتغله

(2) Native

developer هنجيب ← website

يعملها

Connect

Embeded device

↳

simple APP

HTML & PHP

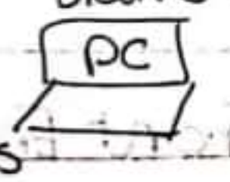
← هنعمل website بسيط علشان نتبسط بيها *

Website Implementation

علشان نعمل website

↳ Server
↳ hosting servies

Static IP



↳ ② Domain

Freewha.com

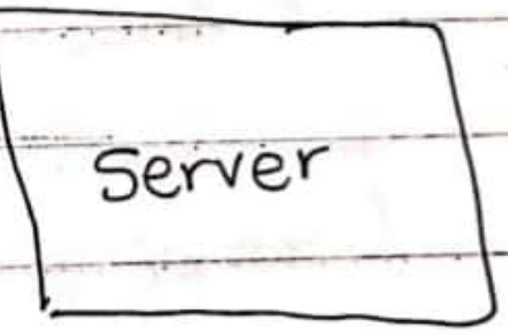
webseeka

ow * ayaramzy

www.webseeka.freevar.com

علشان أستعرض الداتا ال على server ← ودا علشان HTTP →

ببعته من ال Browser هنستخدم (Chrome)



FTP

(filezila) وهنا هنستخدم

بستخدمه علشان أرفع ملفات من ال server أو أنزل

→ Entry Point

أول فايل بيتنادى واسمه index or hostindex (.html or .php)

PhP

هى العقل المفكر

دورها لما أدوس على اللينك مثلا هى تنادى على الـ Script الفلانى

```
<? PhP

echo " ";

?>
```

PhP variables

$x = ...

array

$arr ['x']

$arr ['y']

$_GET ['status']

element	association
x	0
y	10
z	30

high level language
يعني قريبة من لغة الانسان

Tool Chain
كلها بتركب على الكمبيوتر
(SDK)

ES machine langu...

Tool Chain :

يعني سلسلة من الـ Tool ، سلسلة ورا بعض كل واحد بيسلم للتاني ، بتحول الـ highlevel إلى machine lang...

Tool Chain	*
PreProcessor	1

الـ C بتتكون من على الأقل 4 من الـ Tool

أول ما نضغط على run الـ file.c بيروح على أول Tool وهي الـ PreProcessor

وهي Tool

* Search for PreProcessor Directive

* واللي فيها 11 ← PreProcessor بيبدأ بـ #

#include #define #undefine

#if #else ← #elif !! #ifdef

#ifndef #endif

#warning #error

بيدور عليهم ويعمل TEXT replacement ويحط محتواه

* Programa ← ملها ش علاقة بال PreProcessor

بيعمل Text replacement ويشيل ال # ، لو Pragma هيسيبها زي ما هي.

* Pre Processor is target Independent
لا يعتمد على المايكروكنترولر يعني بتاع ال ARM هو نفسه بتاع التانيين

* البريبروسيسور هيطلع file.i) intermediate file
مكتوب بالسي ، لكن مفيهوش #
والباقي موجود زي ما هو بالظبط

بعدكدا file.i هيدخل على ال Compiler
بياخد file.i بيحوله ل assemply code
ويطلع file.asm واسمه
Assembly file
ال Assembly دي عبارة عن instruction
ال بيفهمها ال Processor

، ال Compiler بيعمل دا ازاي ؟
بيمر بعدة مراحل = ٣ مراحل
Front end

[1] Tokenizer
/* int x = 10; */
كل حاجه مهمة في ال الكود بيسميها Token
والباقي هيمسح زي المسافات الفاضية والكومنت

[2] Syntax Analysis
بيشوف ال Syntax صح ولا لا
int x = 10;
Push

[3] Parsing
هو عملية تحويل سطر السي ال مكتوب لسطر ال Assemply ال بيقابله

ال Parsing هيقابله مشكلتين

1) Non defined objects

```
ex; extern void fun(void);
    fun();
extern int x;
void fun2 (void)
{
}
```

Symbol Table

object	Type	state
fun	void(void)	needed
x	int	needed
fun2		provided

global حاجة ← object

يعني أنا بقدمه لوحده جايزه

2) For (i = 0; i < 5; i++)

logical Address at stack

push 0

compare i,5

jump

وهكذا

لأنه إن هملت تعمل X += 5

هنا بقى الـ compiler هيستبدل الكود دا بدا

وهنا مرحلة الـ optimizer (وهو مراحل ودى صورة منه)

optional

يعني بيعمل بيحاول يعمل نفس الـ يتعمل

لكن بطريقة أفضل

(optimization levels)

ومع كل level فيه مجموعة Algorithm بيطبقها
وممكن أقفل ال optimizer ←

* عايز أشوف بقى file ← ال ASSemPly file. دا
* بالكومند ال في السي.

ال Compiler) ←
فيه حته Target dependant
(1) , (2)
(3) في ال independant ←

CamScanner