1

Advanced RISC Machine
ARM
Cortex _ M

① مقارنة بين MicroController

AVR (Atmega32) — 1.5$

* Processor 8 bit — يعني أكبر عملية بعملها تكون 8 bit
* F. Max 16 MHZ → يعني يقدر يعمل 16 مليون عملية في الثانية
* (Flash Memory) 32k
* RAM 2k
* I/O 32 Pin
* Timers 3
* 8 ADC Channels
* 1 USART
* 1 SPI
* 1 I2C
* 5v

oP1 oP2 → ALU

* بسط يستخدم للتعليم وبالتفاصيل في الـ datasheet

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

STM32F103C8T6 — 0.7$

↳ ARM Cortex M3

* Processor 32 bit
* Max Freq. 72 MHZ (up to)
* Flash M 64k
* Ram 20k
* I/O 35 Pin
* Timers 9 (16bit) أصغر واحد
* 10 ADC Channels
* 3 USART
* 2 SPI
* 2 I2C
* DMA , CAN , RTC
* 3.3v

وشوية Prefairals أثناء الكورس.

Embedded System Challenges

↳ Size
↳ Cost
↳ Performance
↳ Power consumption

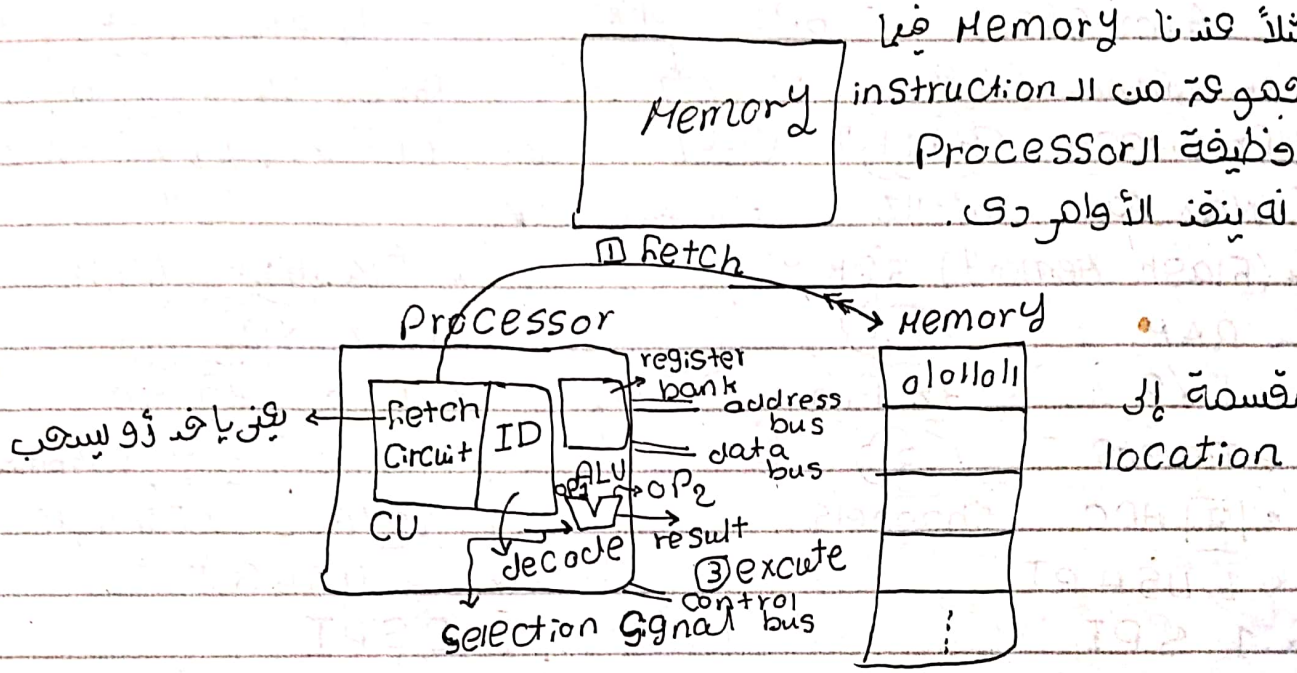
ARM Arch

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.



كدا الـ Processor دوره يمر على كل location وينفذ كل instruction ويدخل على الـ 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 P instruction by Struction

ودى هى عملية الـ fetch ودى أول مرحلة الـ cycle

بتاعة الـ Processor إنه يروح يعمل الـ fetch.

ولكن جه لـ Processor البايزى الـ في الـ Memory دا هيعمل ايه؟

فيه بقى الجزء التانى في 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 وعشان يفهمها
بيحتاج معلومتين

① Instruction set

عبارة عن جدول

Instruction	OP code (operational code)
ADD	010
SUB	100

مثلاً

كل عملية ليها OP code خاص بيها مبتكرش 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 جالك ؟

Ex:

3 bit	3 bit	2 bit
OP code	operand 1	operand 2
010	110	11

بعدين هياخد ال 010 ويروح على Instruction set
ويقول دا معناه اية ؟
هيلاقى معناه ADD ، وال بعده 110 دا كدا رقم 5
وال operan 2 معناه 11 3 ، يبقى ال instruction
دى معناها ان عايزة أجمع 3 + 5 وبكرر حصل
هو هيفهم بس للكن مش هينفذ ، هيقوم بعتها على
ال ALU (Arithmetic logic unit) وهى ال unit
ال فيها كل الدوايس الخاصة بال instruction ال بيعملها ال
Processor زى Add , Muti , Sub .....

علشان ال ALU تشتغل بتحتاج ٣ حاجات :

① operand 1 ، operand 2 ، selection signal

مسئولة عن فتح وقفل دايرة معينة من الدوايرالموجودة داخل ALU

وبكدا ال ALU هي هتطلع ال result

وده بقى عملية Excute

* طيب سؤال ؟

طيب ال بيعرف ازاى أن location على الدور 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

* معلومة : ال Memory و register :

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

① أشهر هذه ال register هو PC (Program counter)

وهو R بيشاور على ال instruction ال عليها الدور في ال memory (على مكانها) ، وكدا 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

② (fetch) أخدتها وحطيتها في ال IR وهو بيشيل ال instruction ال بتنفذ دلوقتي وبكدا ال Instruction decoder بيبدأ decode ال instruction الموجودة داخل IR

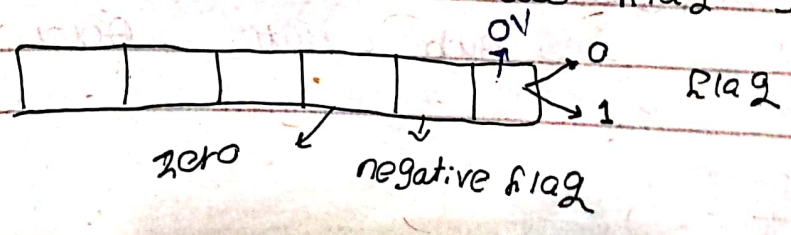
③ ، فيه كمان ACC (Accumulator) وال register دا فيه عملية ال operation ال عملناها 3+5 = 8

و بعدين دايما في النتيجة ال هتطلع بيها بيعمل ال processor

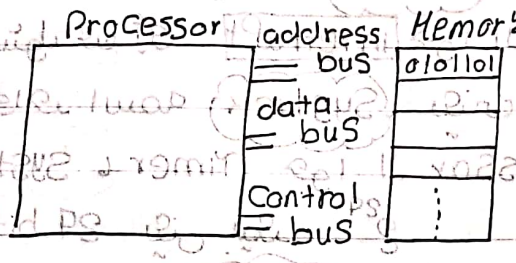
④ كمان PSW وهو من أهم ال registers الموجودة في R bank (Processor status word) وفيه مجموعة من ال flags

ال flag معناه 1 bit

وال flags دي بتعبر عن حالة ال processor



بين الـ Processor والـ Memory والـ PrePherals بيكون فيه buses



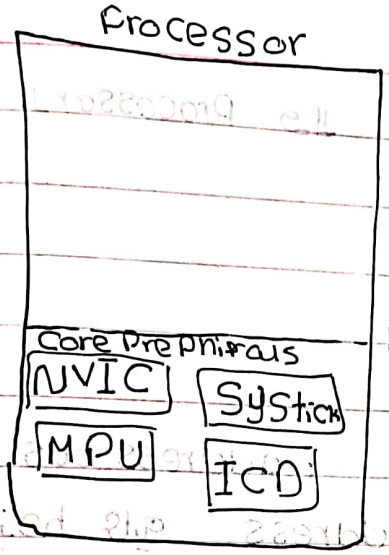
① الـ address bus:
بنحط عليه address الـ location الـ عايز أكلمه سواء location في الميموري أو address في PrePhiral

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

③ الـ Control bus:
فيه مجموعة من الـ signals بتتحكم في العملية الـ عايز أعملها
فتكون R/W أو interupt وغيرها على حسب الـ signals
وبعد كدا نطلع الـ data بتاعه العملية على الـ data bus

ملاحظة:
لما ذكرنا إن الـ Register أسرع بكتير لأن الـ Connection مباشر بدل ما يكون داخل الـ Processor من غير الـ busses بالرغم إنها تعتبر الفيزيكس بتاعة الميموري

ARM بقى قالت هعمل حاجة جوا الـ Processor اسمها Core PrePhirals، يعني هعملك PrePhirals بقى جوا الـ Processor التعامل معاها سريع جداً وقوية جداً وفوق دا كله الـ arm العملاق أول



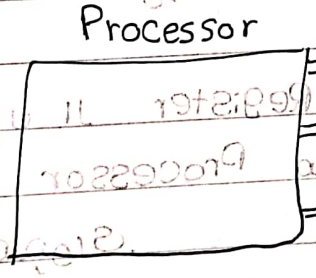
① أول Core Prephiral اسمه NVIC وهو مسئول عن أي interrupt جوا الـ Sys

② و تاني واحد اسمه Systick يعني Timer ، System 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 (هو هيحدد أي حاجة) يعني يخلي حته Memory معينة محدش يقدر يقرأ منها أو يكتب عليها

④ الرابع اسمه ICD اختصار In Circuit Debuger هو عبارة عن tool جوا الـ Processor تقدر تعملها debuge وديباج معناه إننا نشوف الـ Processor أثناء الـ runtime وده بيخلي الـ Processor يمشي بسرعته 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

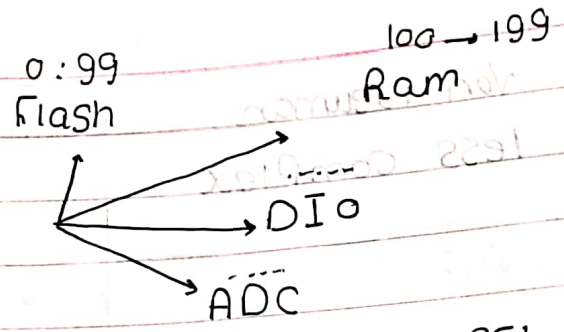
* الـ address bus بيعرفني عدد الـ locations الأقدر أكلمها

يعني أقدر أكلم 2^8 location يعني 256 location from $\boxed{0:255}$

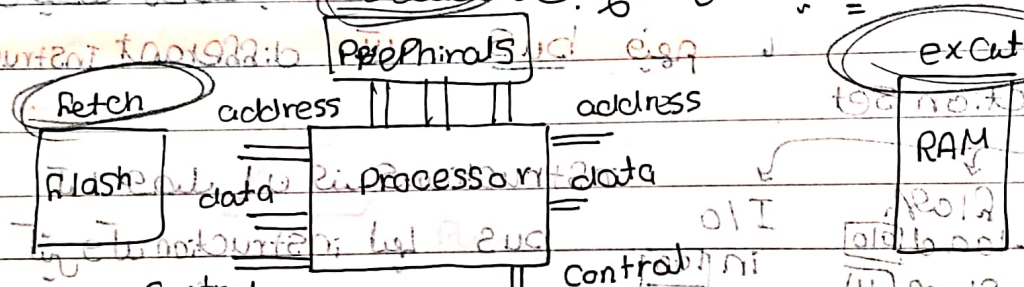
الـ data bus بتعبر عن Size الداتا في الـ location الواحد.

الـ word size هو Size الكلمة ، ولو أنا هعت داتا فلازم يكون حجم الـ data bus مساوي لـ databus . ~~لازم نفس~~ بتاع الـ Processor بتاع Prephiral ، لازم يكونوا متساويين

مثال



وإنا عندي 256 location وكل واحد فيهم 8 عدة ثنوية location
عايز يـ Connect مع الـ Processor ، فهيوزع (P) الـ location
عليهم مثلاً Flash 100 ، Ram 100 ، DIO 32 Address
، ADC الباقي . وكل واحد ليه address مختلف ، والتفاصيل هنا
خاصة بالـ digital design . واسم العالم Von Neumann
، حيث بقى Harverd قالت لما تكون دايرة الـ Fetch بتعمل
Fetch دايرة الـ decode بتعمل ايه؟ ولا : خاصية مستنية الـ Fetch تخلص
وهكذا الـ excute أثناء الـ decode
فكروا إنهم الـ 3 يشتغلوا مع بعض



وهكذا

Control + data + address يعني Bus Set
التلاتة مع بعض .

فكذا كملوا Processor ليه busset للـ RAM والـ Flash
والطريقة دي اسمها Pipelining الفكرة عندي أكتر من
bus set ، والتاسع دي اسمها Harvard Arch

PortC → Pin13 → led on MicroController

Von Neuman	Harvard
* less complex	* faster
	* complex software

↓

نكتب كدا: int *Ptr = 0;
*Ptr = 100;

كدا نكتب على الـ address (0) قيمة 100، فالعادي إن في Von Num الـ address الـ zero موجود في الـ flash، لكن في Harvard عندنا address للـ bus تبدأ بـ zero للـ bus set

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

Instruction set

- RAM: Read 000, Write 001
- flash: load 010, Store 011
- I/O: in 100, out 101 → OP code

كدا حصل إن عندي instructions كتير وكل instruction ليها bus معين، فأي instruction كانتها تروح للـ bus الخاص بيها

* compiler (C) بيفترض إنها بتكتب على الـ RAM، فهنا نضطر نكتب بالـ assembly على شان كدا الـ Harvard software complexity أقل، الأبسط هو Von Num

هنا بقى ARM هيختار السرعة والبساطة !!!

التسطيب باقي السيستم

نهاية L1 + البرنامج

Start L2

ازاى دا؟ لاحقا

② تانى element هو الـ Memory

وهو الجزء الـ هيشيل فيه الكود بتاعنا.

* الوقت الـ أنا بكتب فيه الكود اسمه development Time ①

خلصت الكود بعمل build

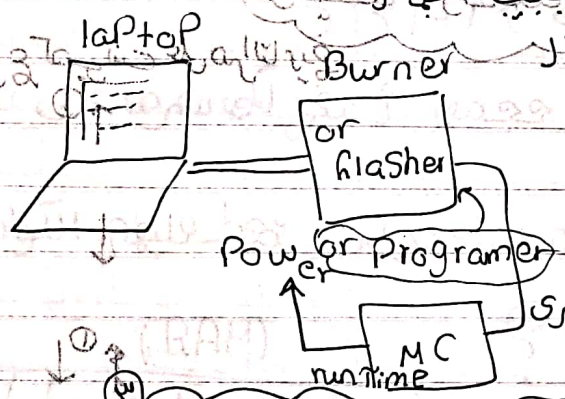
هى عمليه تحويل الكود بالسى لـ binary format الـ هينزل على الميموري وهيكون

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

هو عمليه تحويل من C إلى binary، لسه كده الـ Micro controller

لسه مظهرش، ده لسه على الكمبيوتر، hex ده الـ محتاج (حرقه)

على الـ Micro controller (فيجيب جهاز اسمه Burner)



وده جهاز بيوصل مع اللاب بالـ USP وبيوصل مع الـ kit

أو الـ MicroController بتاعى

وبيبدأ بيعمل download ①

للفايل .hex من اللاب إلى الميموري

الداخلية للـ Micro controller

وده اسمه Burning Time أو flashing Time ③

بعد كده هفصل

الـ burner بتاعى وأوصل Power على الـ M.C وده اسمه

runTime ④ وهنا بقى يبدأ الـ Processor

كده أنا محتاجة Memory تكون عليها الكود.

* ومن هنا بدأت فكرة الـ Electrical Component

لو دايتها دايما تحتفظ بيها وهى تبقى الـ Memory

أول فكرة جت في دماغهم ساعتها انهم يعملوا Memory based on الـ Capacitor وهو عبارة عن خزان بيشحن ويفرغ

Memory based on

Capacitor

وكل مكثف يمثل bit لو مشحون يبقى 1 لو مفرغ يبقى zero

- charged
- not charged

Transistor

- Saturation mode (short circuit) ①
- Active mode
- cutoff mode (open circuit) ⓪ (as switch)

كل bit تبقى بـ Transistor لو في open يبقى (0) ولو short يبقى (1 one)

أيهما أفضل؟!

* أسرع كمبيوتر على مر التاريخ ↑①

يعني لو bit من 0 إلى 1؟

* Performance (سرعة) ↓
* Cost ↑①
* Size ↓↑
* Power consumption ↑↑

المقصود بيها الـ Power ↓↓①

الاحتياجها علشان تحافظ على الداتا

لو بعد سنة هلاقيه Transistor في الـ Saturation لو حطيت كود على زي ماهو ، إنما لو Capacitor وجيت بعد مثلا ٤ ساعات هلاقيه متفرغ فهو ليه leakage factor عالي معدل تسريب عالي ، فممكن بعدما كان 1 رجع zero بالتفريغ ، وده محتاج اشحنه كل شوية ، فبنحتاج طريقة اسمها

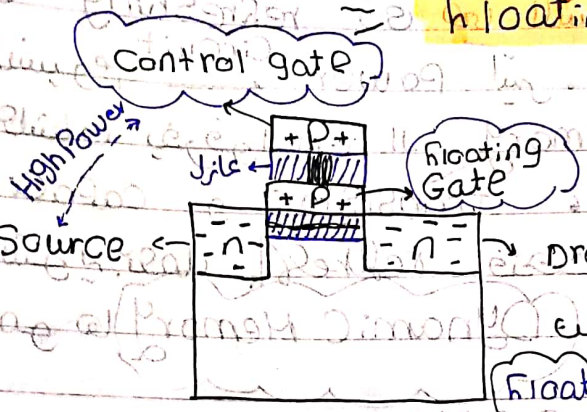
Refreshing Circuit

وهي عبارة عن دايرة مربوطة بـ Timer كل وقت معين حسب بها
ال design بتاع الميموري بتعمل كل شويه refresh
و refreshing دي هي السبب الأساسي إن ال capacitor flash
بيكنسيوم circuit power كتير.
* علشان يفرقوا بين ال Memory ال هي based on capacitor و Transistor ، فابعاً إن ال based على capa
كل شويه تعمل refresh فتحت إن فيه حاجه Dynamic هو هتاج
فسموها Dynamic Memory أو Dynamic Ram
(DRAM) (رخيصة)
أما ال هي based on ← Transistor فعلشان ثابتة
سموها Static RAM ← SRAM
فكده SRAM سريعة وحجمها أصغر لكن سعرها غالي

لكن الحقيقة هما الاثنين مينفعوش يحط عليهم اللود ؟ مشكلة ؟
ليه ؟
(RAM)
ب على ال
غدر نعمل
ج: علشان هما الاثنين من نوع Volatile memory
يعني بيحتفظوا بالداتا طول ما في كهربا متوصلة عليهم لو الكهربا
اتشالت الداتا ال عليهم هتروح
طيب نعمل ايه ؟
عايزين electric Component لما أحطه في وضع معين يفضل
محافظ على وضعه حتى لو شيلت من عليه الكهربا.

لحد ما اخترعوا electronic component عظيم جداً ن اسمه

Floating Gate MOSFET



فكرة بيشتغل ازاى

بعد الفكرة دى.

جابوا مادة عازلة وحطوها تحت

ال gate دى وكذلك فوق ال Drain

ال gate و بقى محصور بين مادتين

عازلتين علشان كدا اسمها Floating gate

تحتها هواء وفوقها

وبعدين عمل Gate تانية

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

ل Channel فى المادة العازلة تسمح بمرور كهربا بشرط

apply high power بين ال Source و ال Control Gate

ولو شيلنا ال high P هتصبح عازلة تانى.

← هنوصل ال High P هتمتص electrons الإلكترونات دى

لحد ما توصل ل Floating gate و شيلنا الكهربا العاليا فالالكترونات

اتحبست (Charged).

طيب عايزين نوقف الشحنات توصل الكهربا صغيرة بين ال Drain و

ال control gate هتمتص ع و تبقى (not charge)

فسواء كدا أو كدا ال MOSFET هتبقى على نفس الوضع هتمسك

يوصل ل 50 years وهنا بقى Non volatile Memory

Charged — 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 عيب !!!

← 1 بيحتاج high Power علشان اعمل create للـ channel

← 2 عنده high access time (بطيء) ماهو علشان أعمل كدا هحتاج أعمل apply high Power فترة معينة.

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

ROM (Read only Memory) وده based on 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 مقصد مش Provide 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 وهي زي 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 ضوء الشمس زي مثلاً بتستخدم في الأجهزة المحتاجة protection زي آلات الحرب والأسلحة والحاجات الخطيرة

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
① Byte access (W/E) → write, erase	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
وكانت ال Flash ال block بتاعها 100 byte
↳ on EEPROM ⇒ 100 usec
↳ on Flash ⇒ 1 usec

كدا ال Flash أسرع

فالإجابة: على حسب إنت عايز تكتب ايه، لو عايز تكتب حاجات صغيرة جداً byte بعد byte، فالأسرع EEPROM، لكن لو حاجة كبيرة بالجيجا زي 1000 بالصيجا يبقى ال Flash أسرع

باقي المقارنة :

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

ملاحظة

الاثنين مصنوعين بالـ Floating MOSFET لكن ترتيبهم في كل واحد مختلف ، علشان كده تلاقي الترتيب ده 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

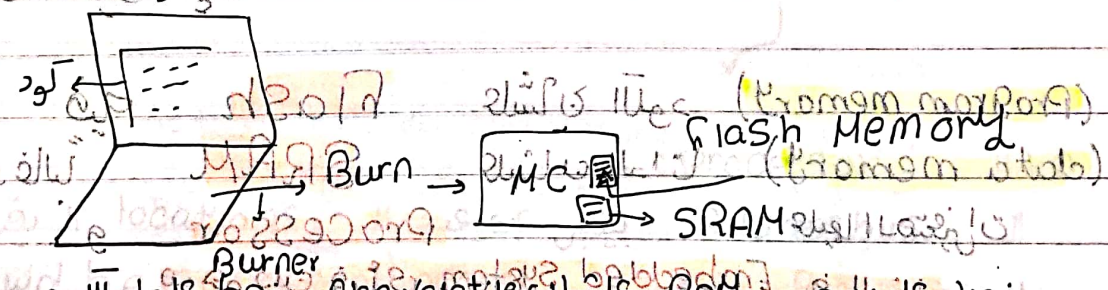
أيهما أفضل في عالم الـ Embedded system.

تعالوا نشوف الأول المميزات

عندك نوع

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

هاسؤال نستخدم أي واحد فيهم في الـ Micro controller ؟



فهنا عايز الـ ROM علشان non volatile نحط عليها الكود.

طيب أي نوع ؟ فيها ؟

* الأولى منطقية هو ناجح أكثر براحتي وأحط كذا مرة و هكذا

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

فالأفضل هنا الـ Flash أسرع وأرخص لأن WRITE ← 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 والموبايل
* 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 حبت الحجم ده في الـ SRAM هلاقيه سعره عالي جدوداً

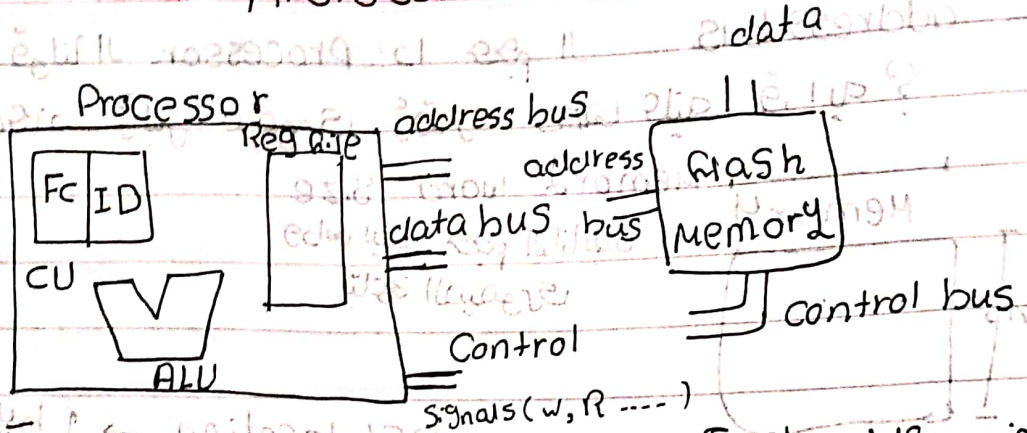
كده متفقين إن أي Micro controller

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 لسه 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 ، 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 ؟

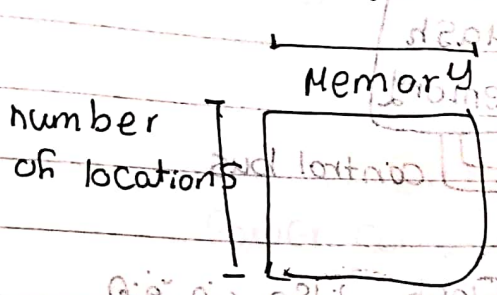
هنا ال Processor :

① ال Address bus بيعبر 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 ؟



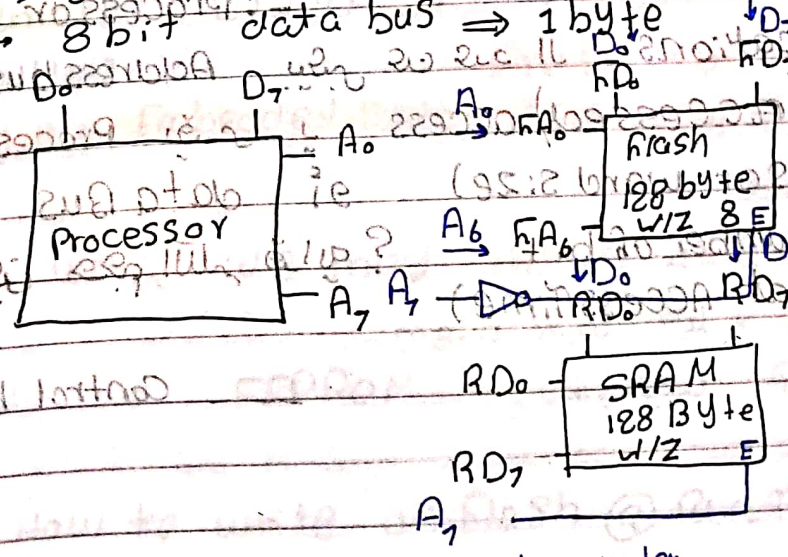
Memory word size
وده بيمثل حجم الداتا بس
بتاعة الميموري

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

فلو قولنا زي الأول ال Processor ← 8bit معناه
إنه يقدر يعمل access على 256 ← locations
طيب كل location بقى قد ايه ؟ ؟ : حد يقي حجم ال data bus
فلو 8bit هو كمان يبقى محتاج أشتري memory
فيها 256 ← locations
يعني address bus = 8bit
و ال data bus = 8bit .

طيب لو

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



يعني لو حطيت على أصفار على A 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 location في ال flash	0	0	0	0	0	0	0	0	flash
آخر location في ال flash	0	1	1	1	1	1	1	1	
أول location في ال SRAM	1	0	0	0	0	0	0	0	SRAM
آخر location في ال SRAM	1	1	1	1	1	1	1	1	

لو أنا بقى ك Processor طلعت على A_7 ← zero هنا كدا A_9 هتبقى 1 بعد ال inverter فكدا flash هتبقى enable و SRAM disable وباقي الأصفار هتدخل على ال flash و تأكسس أول location في ال flash.

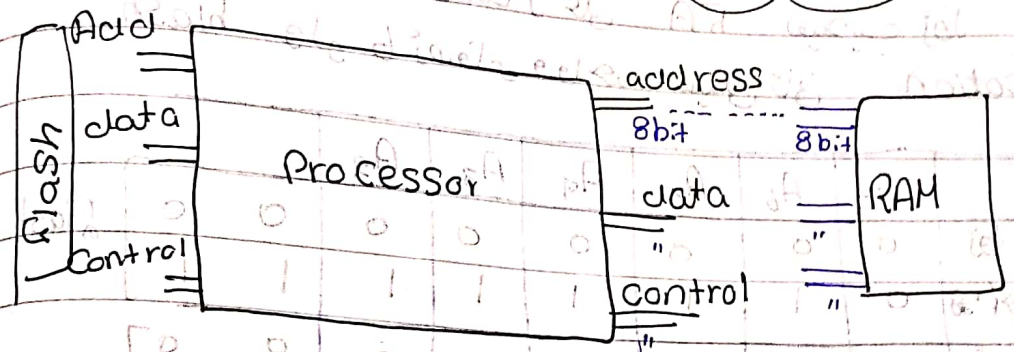
ودي بقى طريقة von neumann

المهم تكون عدد ال location مناسب لل Processor حتى لو ال Processor ليه addresses زيادة بنسميها reserved 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 → 0000

يعني على lines مختلفة

Harvard
↳ Pipelining
↳ Not incremental Addresses

العيب الأساسي في Harvard هو ال Software Complexity

↳ different Instruction Set.

هو ايه بقى؟

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 Assembly

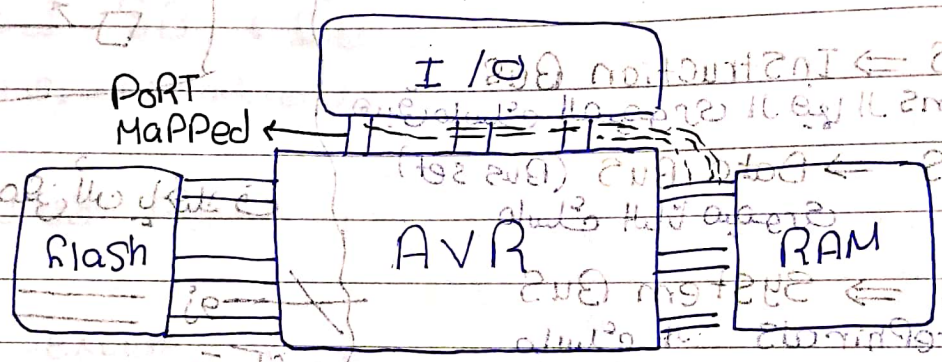
ماهي ملهاش حل.

لأن في Assembly instruction لكل واحد فيهم مختلف عن التاني زي

RAM
* load
* Store

Flash
* Read
* 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 حطيت I/O على مكانين على 2 addresses bus لو مع ال RAM اسمها Memory map لو حطيتها لو مع ال PORT MAPP ، بالتالي كل register هيبقى ليه 2 addresses

* (0x20) = ✓ فلو عايز تكتب

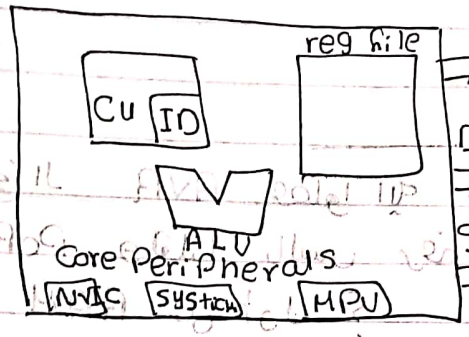
* (0x50) = ✗ زي Assembly

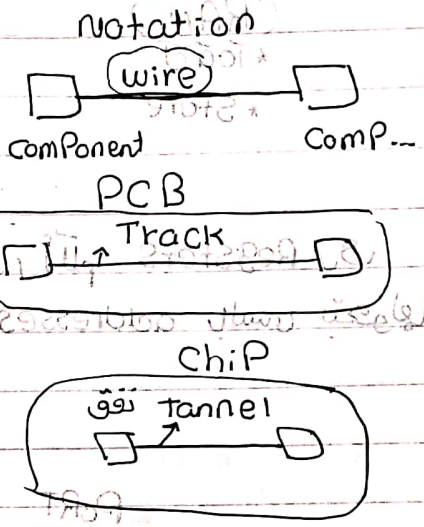
* (0x20) = ✓

لأن لو كتبت بالسي هتخسر فكرة ال Pipelining و هتكتب على ال RAM (0x50) على Assembly لأنها الكتب على ال I/O فعلشان

~~فكرة ان الكتب من خلال (0x20) هتحقق ال Pipelining~~

ARM Cortex M3





IBus ⇒ Instruction Bus
هيبقى ماسك الميموري ال فيها ال instructions

D Bus ⇒ Data Bus (Bus set)
ماسك الداتا ميموري

S Bus ⇒ System Bus
ماسك ال Prephirals

هو هو harvard لحد دلوقتي

هما نفس المعنى لكن باختلاف المكان.

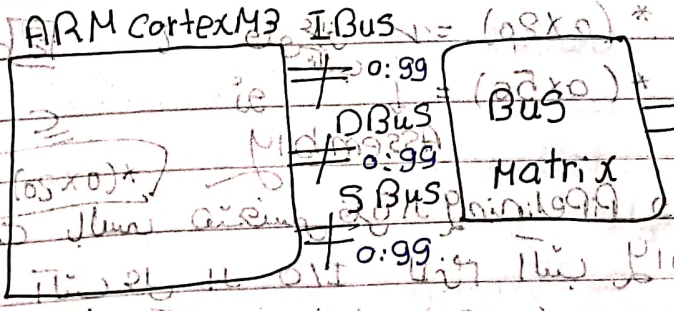
Bus ⇒ زو

مكان مجموعة من الكونكشنا ثلاثة من سلك

ولو Bus set #

Bus Matrix

لكن ARM عمل ChiP عظيم جداً اسمها Bus Matrix بتاخد الثلاثة buses دول وتطلع bus كبير



32 bit data
32 bit Address

ex 300 Address
↳ 0 → 99 RAM
↳ 100 → 199 flash
↳ 200 → 299 I/O

طيب وهنا هو كان في الأصل harvard وبعدين بقى في الجزء التاني
تابع ال Bus Matrix وكأنه بقى von numann
*
مثلا عندي هنا 300 Address من بعد ال Bus Matrix
RAM ↳ 0:99
flash ↳ 100:199
I/O ↳ 200 : 299
هنا كتوصيل 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 address 100 زي الرسمة الفاتت
فهنا ال Bus Matrix
لأنها بتحول ال addresses من Range قبل ال Bus Matrix
إلى Range ما بعد ال bus Matrix
هي مش بتكمل fetch و decode و excute لا هي
بتكمل ال translation في وقت بيكافئ ال fetch أو
decode أو ال excute يعني بقدر دا كله في فترة واحدة وهي
clock واحدة
يعني مثلا ال IBus هتطلع address فال BusMatrix تحولها وتقوله
تقوله روح اعمل fetch ، DBus تطلع addr فال
BusMatrix تحولها وتقوله روح اعمل fetch فيكون IBus
خلص fetch هيدخل في ال decode
* فكدا إنا كيوزر user ال Processor مش شايف الموضوع
من بعد ال BusMatrix وكأنه von neumann وبتأقدر
أكتب C بمنتهى السهولة
ملاحظة: ال Buses (I, D, S) هنا
not Accessable for user
(إنا بشوفه بس ال BusMatrix فبتخيله إنه von neuman لكن

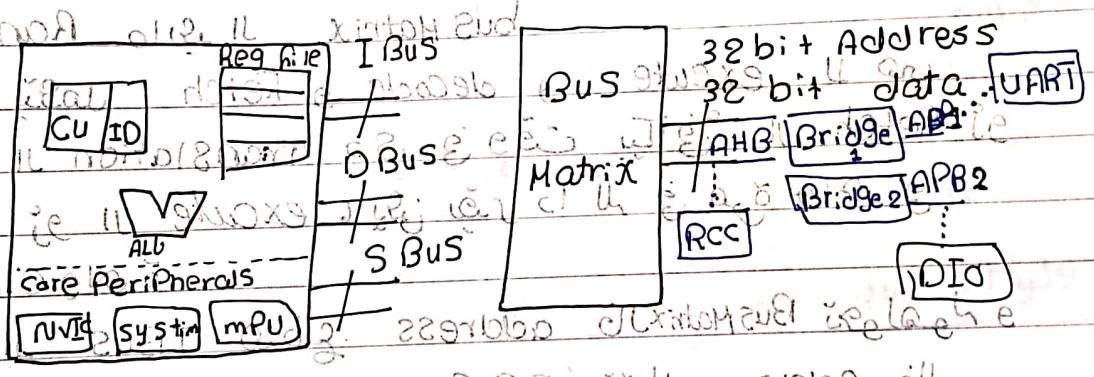
هو في الحقيقة من جوا Harvard وادا السحر الا عملته ARM
علشان يجمعوا بين الاثنين (سريع (Pipering)، سهل في ال Coding
لأنه بايت Von neumann .

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

End ?

Lec 3

ARM Cortex M3



دحنا دور ARM بينتهي ويجي بعدها ال vector يبدأ
بعد ال PerPhirals بتاعته من RAM، Timer، Flash...
لكن في ARM مستنتش قال هعمل قواعد يركب فيها ال PerPherials
بتاعته، هعمل Protocolls يركب بيها ال PerPherials
متوضح فيها شكل التوصيل وغيرها
في ARM اخترع حاجة اسمها

AMBA

(Advanced Micro controller Bus Arctichure

(27)

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

AMBA
- AHB
- APB
- وغيرها ....

دول الموجودين في الـ MicroController بتاعنا
AHB: Advanced High Performance BuS
APB: Advanced Peripheral BuS

* AHB:
- بيدي أعلى Performance في الـ AMBA وهو
- * High Power
- * Complex design
- * Full duplex يقدر يتكلم ويسمع في نفس الوقت RW
- * Multi Master يعني ممكن أكتر من 2 يقدروا يتكلموا على الـ BuS دا

APB:
- هو اللي بيستخدموا منهم الـ PerePhirals.
- * 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 عايز Performance عالى
زي RCC والـ PeriPherals دي مش بيناسب الـ ARM الـ عملاها
لأدي الـ عملها الـ Ventor
كدا الـ PeriPherals دي كدا هتكسر الـ Performance عالى فكده هنكتب
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 أبطأ 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 Mica

② جزء الـ Bridge الـ ARM هي العلاه

ST: Register Description
Documents
ARM: Programming Manual

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

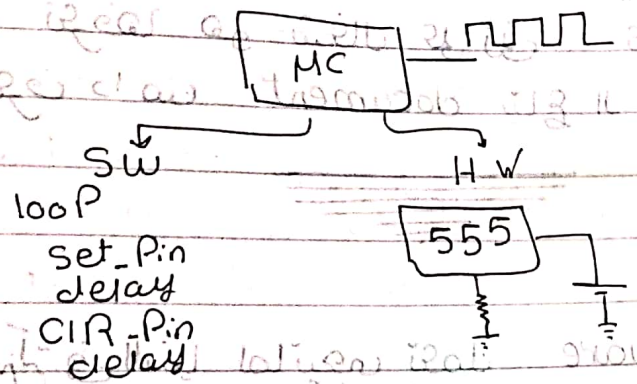
مثال :
عندنا مثلا PerPhiral اسمه SysTic جوا ال ARM
هنفتح ال Programming manual

* لو أنا قولت شغلي ال PeriPheral ال اسمه كذا ، لازم أزود
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 له
① محتاجين نعرف التفاصيل بتاعة ال System دا.
فبتبعتله شخص اسمه Analyst 1
بيروح يقعد مع ال Customer ويقوله عايز تعمل ايه؟ ويأخذ كل
تفاصيل ال Project اللي عايز يعملها ويكتبها في document
اسمها RFQ (Request for Quotation) اختصاره
عادة ال Analyst مش بيفهم هندسة وال Customer
بيسأل منا حاجتين هنعمل بكام؟ وفي وقت قد ايه؟
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 أكتر Technical
و بيعملها التيم الهينفذ الـ هو عرض على الـ Customer السعر والوقت
و CRS بيبقى فيها requierment كتير جداً.
و بتقابلنا معضلة كبيرة جداً إن الـ requirement دي هتتعمل بالهارد
ولا بالسوفت زي مثلا



ومن هنا يجي الـ System Engineer و هو شخص
أكتر حد خبرة و هو الـ هياخد القرار على مستوى الـ System
وكل requirement هتتعمل بالـ Software ولا الـ Hardware
و من 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 و الموجود
في الـ SRS كل Software محتاج كود يعمل كذا
و في HSI و هنا بقى الـ MicroController نوعه ايه والـ ChiPs
عاملة ازاي والـ Connections بين الـ MicroController وبين
ChiPs عاملة ازاي وكل مقاومة كام و كلام كتير .....
غالب الشركات في مصر (الكبيرة) بيبدأ دورها من عند SRS
و في اللحظة دي بيبقى معانا document فيه المطلوب Software
requirements

ومن هنا أول خطوة في كتابة الـ Software
نبدأ نقسم الـ SW لمجموعة من الأجزاء الصغيرة وكل جزء من
الأجزاء دي هيعمل حتة من الـ SW وبعدين هنركب الأجزاء دي
مع بعض ونعمل الـ SW الكبير الـ بيعمل القصة دي شخص
اسمه Architect هو المسئول عن تقسيم الـ SW الكبير
إلى SW صغيرة ونحدد بقى كل SW من الصغير إيه الـ function الـ هيبدأ
بتقديمها وبتعمل إيه وبتاخد إيه من الـ Parameters وبترجع إيه والوقت
والمنطوري وكل حاجة لأنها إطار عام للـ SW
والشخص دا بيقرر بهما حتين واحدة اسمها Static Architecture
والتانية بالـ Dynamic Architecture
نبدأ بالـ

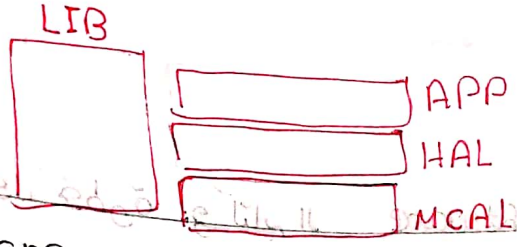
Static Architecture :

بيقول فيه إيه الـ Components (SW) أجزاء
الـ عندك في الـ System وبيرسم
القصة دي في حاجة اسمها
layered Diagram
يعني بيقسم السوفت وير بتاعي
لمجموعة من الـ layers
وهي وهمية لكن تنظيمية
أول layer من تحت اسمها
MCAL (Microcontroller Abstraction layer)
وده هنحط فيها الـ Software
الخاص بأي Peripheral جوا
الـ Microcontroller زي الـ DIO
والـ ADC، فوقها layer اسمها
(Hardware Abstraction (HAL) layer)
وهيكون فيها drivers بتاعة أي هاردويير
بره الـ Micro زي Motor مثلاً أو
أي سنسور أو الـ wifi

Dynamic Architecture

تشتغل runtime

بيعملوا الـ Archtect لكل Function عندك
متاخد execution Time وقت قد إيه
و RAM consumption يعني فيها
variables قد إيه في الرام و
Flash consumption وده كله بيقدمه
في document مهم جداً اسمه
HLD (High level design)
فالـ Architect واحد من أهم
الـ Position المطلوبة في مصر
ومن هنا نبدأ نبني الـ SW
الصغيرة دي يعني كل component
لكل مهندس علشان ينفذها
* قبل ما نبدأ ننفذ كلنا بنمر
بمرحلتين



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 ال هتربط الناس دي كلها ببعضها

يبدأ لما يمسك كل Function الدالة الكومبونتس ببعضها وال layers دي

ال Archtect ويرسمها Diagram ثابتة لكن ممكن تختلف بزيادة وهكذا

ويقول هي هتشتغل ازاي وأشهر ال APP ال main

* مثلاً Function اسمها SetPit في قاعدة بتقول إن أي كمبونتس

بتاخد رقم ال Channel values في layer مقدرش يكلم غير ال

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

يعني ال designer يقول هعمل يكلم MCAL. الكلام دا يعني call fun

Condition لما تكون أكبر من في بقى layers اشهرها اسمها

num of channels لو yes في library LIB وهي horizental layer

هيعمل End لو لا يبقى وامختلف عن ال vertedal لأن فيها

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

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

set Port لو low يعمل CIR أشهر حاجة اسمها standard types مش int

وبعدين end داكلم على وزي BitMath زي الابتعل set و clear

شكل flowchart علشان ودا بقى أول مرحلة بيقوم بيها ال Archtecture

يوضح ازاي نلف الكود. ويبدأ يرسم ايه الكمبونتس الي 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

كمبوري وكوفت وال اسمها و ال Arguments وايه return.

designe يستخدم لغة

اسمها UML وهي مهم انه اكرها مع نفسنا

(Unified Modeling language)

عبارة عن شوية رسومات

علشان مش depended على لغة برمجة معينة

تفهم الحاجات دي مهم جداً

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

والـ designer بيحط عمله دا في document اسمها CDD

CDD بتعمل Component Design Document

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 أوي 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 مش شرط يكون مهندس خريج هندسة

بس لازم يكون فاهم OOP

بعد كدا مراحل الـ Testing :

1) unit testing بيعمله الـ (Developer) مع آخر حاجتين تانيين

unit → function

فبنمسك كل function ونعملها Test و نبعتلها الـ argument

بتاعتها ونشوف هتطلع صح ولا لا

2) Component testing بيعمله (Tester)

وهنا بناخد كل الـ functions بتاعة الـ Component الواحد

ونعملها Test

3) Integration testing بيعمله (Tester)

وهنا بنعمل Test لكل الـ Software مع بعضه

* كدا الكود كله مظبوط، لكن فاضل الـ Hardware

4) Validation testing (Validator) موجود في مصر عادي

وهنا بنثبت الـ System على الـ Hardware

* لو الـ System مشتغلش في الآخر مع الـ Customer

هما أخطر اثنين الـ Architect و Validator.

هنا بقى في الكورس هنركز على الـ Developer و Architect

ملاحظة :

① CoTS : Component on The Shelf

② Cash Memory

هي ميموري موجودة جوا الـ processor نوعها SRAM دورها

بشيل فيها داتا مهمة شبه شوية GPR لكن هي أكبر بالكيلوبايت

أو جيجابايت، دورها أسرع من الـ RAM لأنها أقرب من

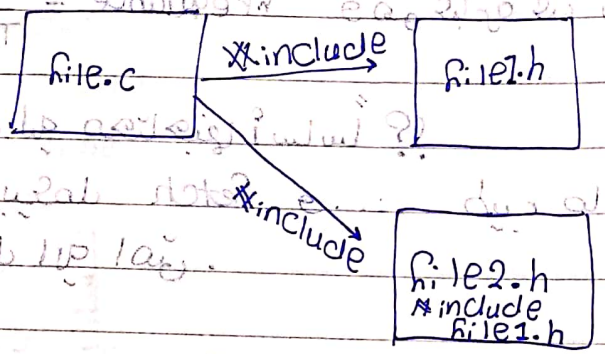
البروسيسور، وأبطأ من الـ registers هنعرف السبب قدام

الداتا سريعة الـ access سواء read أو write.

Configration Management [W]
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
وبكدا file.c موجود فيه file1 مرتين ودى كارثة
فبدأ ال developer يعملوا طريقة تحميه من إن الخطأ دا يحصل
اسمها Header file guard هو تكنيك لو عملته الخطأ دا
فأول include بس هو ال هينفذ والباقى لا، الطريقة ازاى
#ifndef Hamada → اسم
infile.h

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

!=
Code
#endif

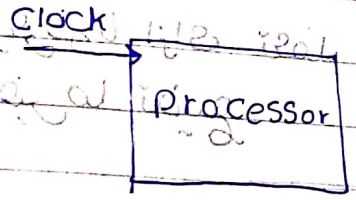
PreProcessor directory

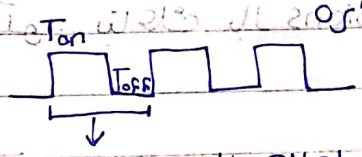
* شغل Software

Clock systems

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

Clock دي عبارة





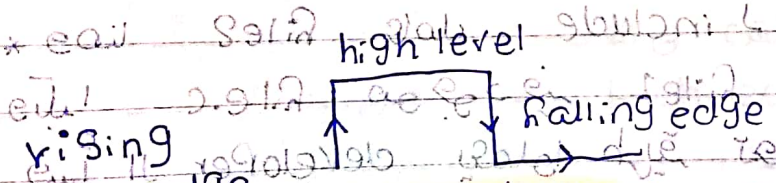
T → clock cycle

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

* طيب ليه محتاجينها أساساً ؟؟

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

cycle



كل واحدة فيهم اسمها MiCro Cycle والـ Processor بيستخدم كل MiCro Cycle علشان ينفذ حاجة معينة ، فالـ Cycle الواحدة بيحصل كذا حاجة وكذا اسمها clock used for synchronization

إزاي هنعمل الـ clock ؟

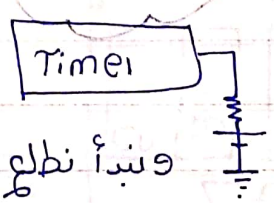
* الإجابة قادمة ...

Clock System

هي في الدنيا نوعين والمقصود بيها ال System ال بيطلع Clock

Electrical

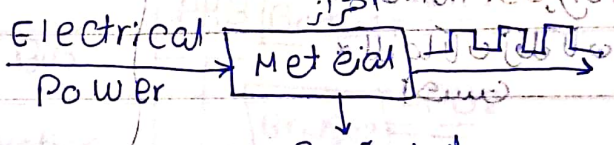
هي دايرة RC
R → resistor
C → capacitor
وهي عبارة دايرة فيها
Timer زي 555



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 ويتهزاز vibrate
بيعمل ال square wave



المادة دي

- Crystal Oscillator
- Ceramic Resonator

والاتنين دول *
هما الاتنين ليهم طرفين واحد بنأبلاي
عليه كهربا وطرف بناخد منه ال clock
زي الكريستال ال عارفينها *

* اختيار واحد من ال ٣ بيعتمد على ال System بتاعنا :
مقارنة

	electrical	Mechanical	
	RC	Ceramic Resonator	Crystal oSCillator
Cost	أقل سعر ↓	وسط ~	أعلى سعر ↑
ACCuracy	↓	~	↑
Settling Time	↑	~	↓
	يعني Time الاستقرار وهو الوقت اللي بتاخده من f=0 لحد ال nomenal الرعاية أوصله * كل ما كان أقل كان أحسن		الأفضل
Noise Immunity	↓ يتأثر جداً	↑	
	مناعة ضد التشويش (الأهم في المقارنة كلها)	بسبب المقاومة والمكثف	
↳ Temperature noise	↓ بتتأثر بالحرارة	↑	↑
↳ EMI noise	↓	↑	↑
↳ Vibration noise	↑	↓	↓

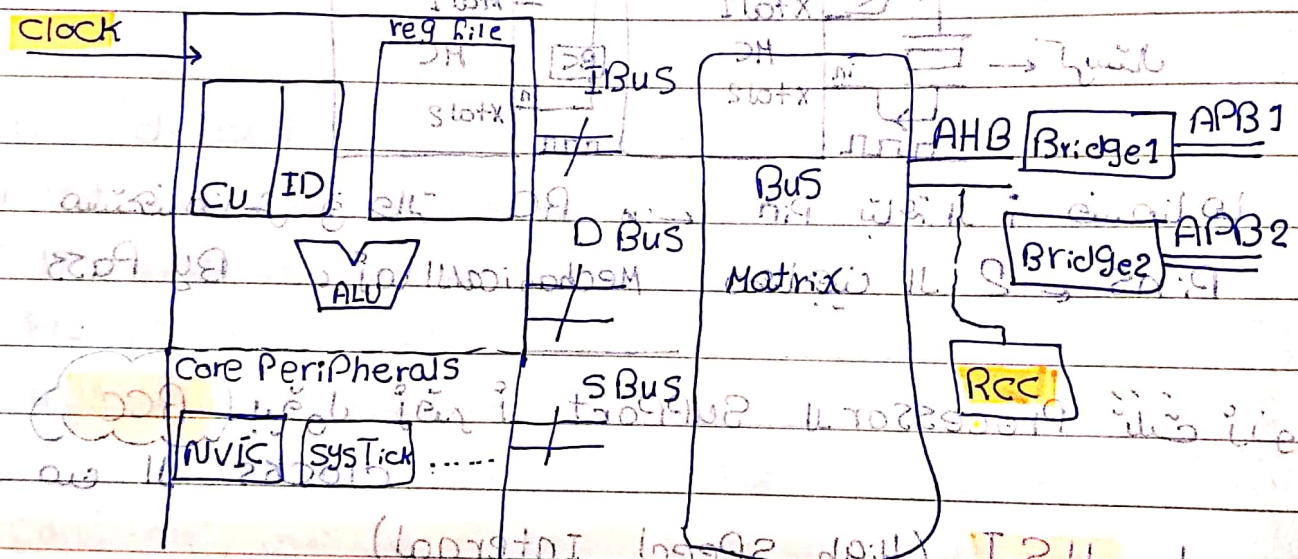
علشان كدا لازم أعرف الأول ايه ال noise ال system هتتأثر به

احنا عندنا في ال MicroController بتاعنا ال RC و Crystal
وال PerPheral ال اسمه RCC هو المسئول عن اختيار
إن Clock System ال هيشتغل.

End 3

Start 4

ARM Cortex M3 Architecture



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

سؤال: Difference between 1_Reset Hard 2_Reset Soft
3_Reset warm 4_Power off Power on

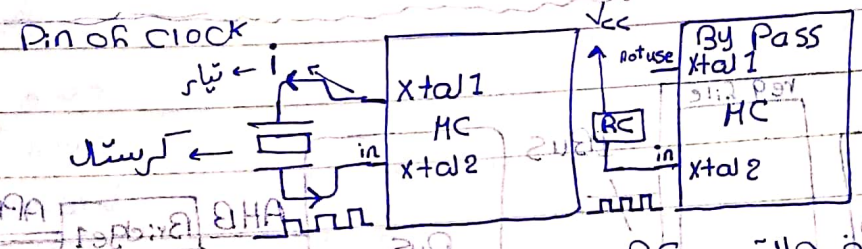
Reset & Clock Control

أي RCC هو Peripherals ال ST من العملاء لأنه برة ال Core Peripherals يعني برة ARM

* أول معلومة

1 Vendor Peripheral

2 Clock Connection



مستخدمناش في حالة RC ← Pin بتاعة ال i ضمناها

By Pass أما ال Mechanical محتاجين ال 2 ← Pins

RCC بيقول أقدر أ Support لل Processor ثلث أنواع من ال clocks

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

clock → PLL → clock * X

up to 72 M

أومال ازاي الـ Processor ممكن يشتغل على 72M وتحوّل 25 ؟

* لأن أعلى clock هتوصل هي 25، لكن ممكن قبل ما أوصل الـ clock دي على البروسيسور أكبرها، عن طريق دايرة اسمها PLL وهي دايرة بتاخد clock وبتطلع clock

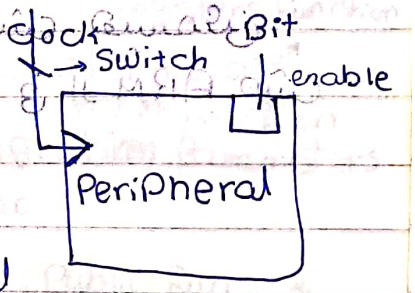
* هنا بقى الـ RCC أول حاجة عملها هي اسمها ① Processor clock selection ميزة

تكمل Reset & clock control

① Processor clock selection

تاني ميزة

② Enable / disable Peripheral clock



أغلب الـ Peripheral بيكون ليها Bit بتكون لها enable (0 or 1)، لو Peripheral بيكون داخل عليه clock سواء هي الـ داخله على الـ Processor أو Pre Scaled منها يعني داخله على دايرة صغيرتها شويه

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

- المنطقي لا.

لكن هو بيـ Consume باور ازاي؟

تعالى نشوف السؤال الأول How the Peripheral is enabled?

زي ما قولنا عنده Pin اسمها enable ①

وعنده دايرة اسمها ② Enable check circuit

Periodically check the enable bit

بتبقى تبص على الـ bit دي لو

① is written to enable bit

دا معناه إن السيركيت دي تروح تشغل الـ Peripheral circuit

is written

Disable Peripheral circuits

لكن السيرلت دى شغالة على طول علشان كدا ال PeriPheral
لو هو disable هيحصل إنه Consume Power صغير جداً
* طيب نعمل ايه علشان مي consume ش Power ؟
هى الدايرة دى محتاجة Clock علشان تشتغل
* طيب ايه رأيك لو قطعنا عنها ال Clock و هنحط ال Clock
على Switch وال هيتحكم في ال Switch دا هو ال RCC
* كل PeriPherals في ال ARM هيكون ليه enable
و disable لل Clock .

ملاحظة : ال DIO مفهوش Clock
أول خطوة ① بنفتح ال Clock على ال DIO مثلاً PeriPheral
② بنروح نشغل ال PeriPheral دا
لو قافلين ال Clock ال PeriPh دا مش هيسمعني :
* ودى ميزة عالية في ال MC بتاعنا علشان كدا في ال ARM مش
بيحصل استهلاك باور كتير

* كل ال RCC ليه وظيفتين :
② بيعمل enable و disable لل PeriPheral
① وإلى اختار ال Clock الداخلة على ال 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 لل Clock اللي عليه من ال RCC ثم نعمل
enable لل Per... نفسه .

معلومة :

open @ :

PLL & FLL → How it works?
↓
base

* 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 functions, Private function)
- → XYZ_ Config.h → configration Parameters — ممكن أغيره

* Public fun نكتبها من جوة ال drivers اللي بيستخدموها بس function زي EnableClock & disable

وفيه Private func مش أي component بيستخدمها

② Include

ترتيب ال include

↳ ① libraries — أول واحدة

② interface file of lower layers

③ own Driver file
- ① interface file
- ② Private
- ③ config — أحياناً ومش فارقة

10 في الداتا شيت

كل Register ليه اسم وهو عنوان ال register وليه offset

لكن بالنسبة لحاجة اسمها

كل Peripheral ليه

↳ Base address أول أدرس

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 دا فين

هيبقى محتاج أعرف ال Based address بتاع ال RCC

casting

((U32)10) = 5

افهم address 10 دا هو عنوان وسمعت ال

dereference وعمل U32 على إنه

وحط فيه 5

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 في حاجتين

إنه بيكنترول ال input 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 و ال 3 ، من 5 registers

بيعملوا enable & disable لل Clock على ال different Peripheral.

AHBENR لل AHB Enable Register

APB2ENR

APB1ENR

أى R بيكون 32bit [0:31]

Naming Convension

① Public function

SWC (Software Component)

SWC_ReturnDescriptiveVerb

② Argument

Copy_typeDescriptiveNoun

③ Macro

↳ All Capital

↳ SWC_DESCRIPTIVE_NOUN

→ on data sheet: (Rac)

CSS : clock Security system

0 → disabled

1 → enabled

هو SYS بيشتغل Security لو عملنا enable ، فلو كنت مشغل (HSE) والـ HSE خرجت 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 لك 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 مع كل زيادة إنت بتزود 90kHz ومع النقص

كذلك

example

error

8MHz + 80kHz

هتنقص من الرقم دا ؟ في كل مرة 90kHz

$\phi 0000$ = 16

~~$\phi 0110$ = 19 وبقى كده~~

→

بنجمع

مهم

HSICAL[7:0] H.W calibration

يعني ال Hardware هو ال بيعمل بيقيس ال error ال 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 1010

وساعتها هيفضل 10 KHz error ←

8 MHz + 10 KHz

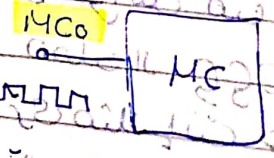
فدور ال TriMMing هو التحسين وآخره أنقص 16 step
أو أزود 16 step

CR

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

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

---



ده Pin؟ ممكن أطلع عليها أي Clock

لو طلعنا عليها 0XX يبقى مفيش Clock ، ولو حطينا عليها 100
يبقى طلع عليها System Clock وهي ال Clock ال طالع على
ال Processor

lec 5

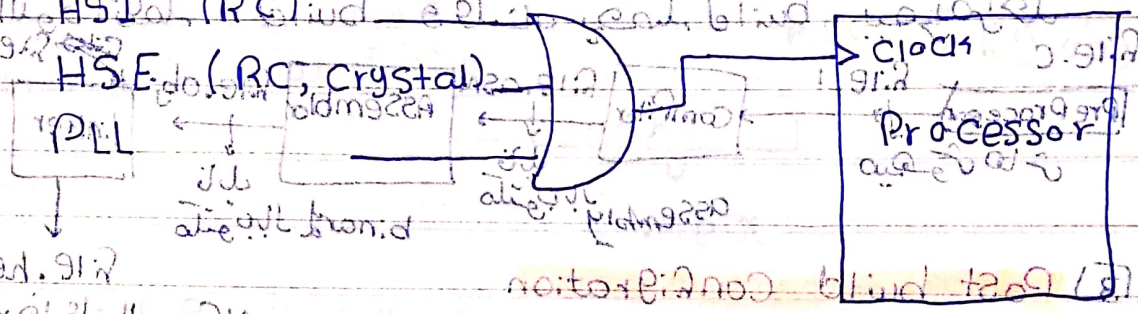
RCC

* Clock Control :

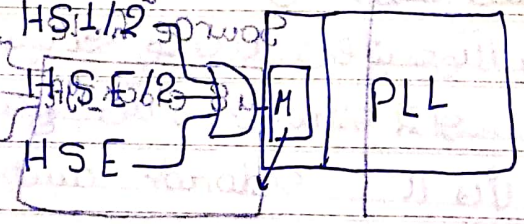
① enable & disable for ~~all~~ Peripheral Clock

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

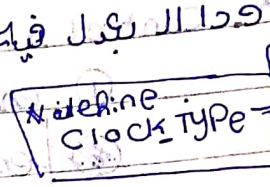


Multiplier جوا ال PLL

PLL هو هيكون ال out .

Configuration Types :

وده اللي بعدل فيه وبناءً عليه الـ Code يتغير.

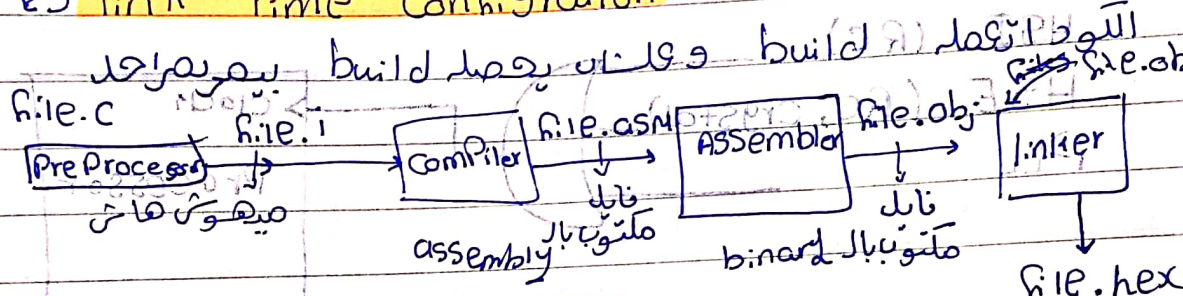


1) Pre_build Configration

اليوزر هو بتعمل Configration للكود قبل ما يـ build (Compile) ، وده 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

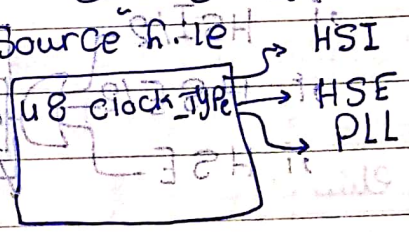


3) Post build Configration

ويدخل على الـ MC

تكملة ، هو بيقول إن الكود نفسه عبارة عن object file وهديلك

Source file



وبعدين تعمل buit فطالع object file تاني (file.obj) ثم تـ link الفايل الأساسي file1.obj مع file2.obj الـ هو طالع من الـ variable مع بعض علشان تطلع file.hex

link Time

RCC_Program.C

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

اعمل build للفايلدا
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

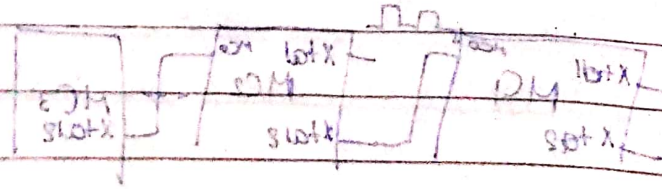
.hex

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 حصلت اللوجيك بتاعي لأن لو حد بيشتريه وهو هياخد
ال file.obj مش ال Source file ، لكن في هنا مشكلة
هي ال Memory لأن ال variable مش ال #define
فهيقل ال Memory consumbsion على

أفضل واحد منهم

3) Post build configuration

قال مخليك تعرف تغير ال configuration أثناء ال runtime
مش محتاج كل مرة تعرف اللوجيك جديد



ازاى ؟ هناخد حته من الميموري

Init → Read للميموري

Memory
Clock Type
Configuration

والتغيير عن طريق أى
Communication Protocal
من حيث تغيير القيمه الى فى الميموري وأنادى
على الـ function تانى من غير ما أعرف
الكود تاني .

open Q

Software

Why IMCO?

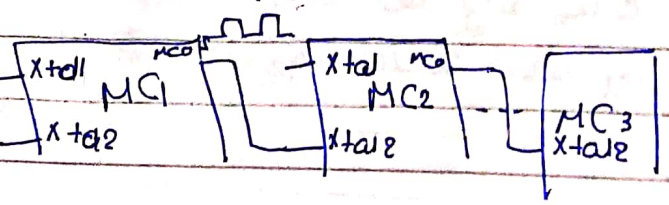
① for Diagnostics

لو عندي شبه clock معينة وعايز أطمن الـ clock اللي بعلمه
للمثال مشغلة RC وعايز أطمن شغال على 8M مظبوط
أعمل ايه؟ أطلعها على Pin بتاعه MCO وأرسمها على الـ osrescope
وأعرف قيمتها قد ايه والـ error قد ايه واعمل Trimming قد ايه
وكذلك للـ PLL

② clock cascading
Pros → Resource Reduction
→ Synchronization
Cons → Dependency (عيب)

شايفين نفس الزمن
هو ايه الأول

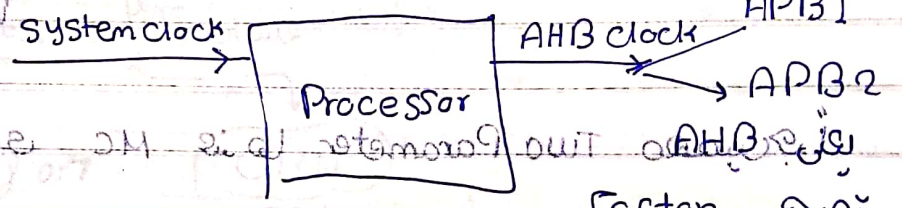
وعايز اشغل الاتنين على 16M
بدل ما أجيب 2 Crystal
أجيب واحدة للأول وأخليه يطلع
على الـ MCO الـ 16M وأخدمه
الثاني MC لـ Xtal 2 على as input
وده اسمه
clock cascading



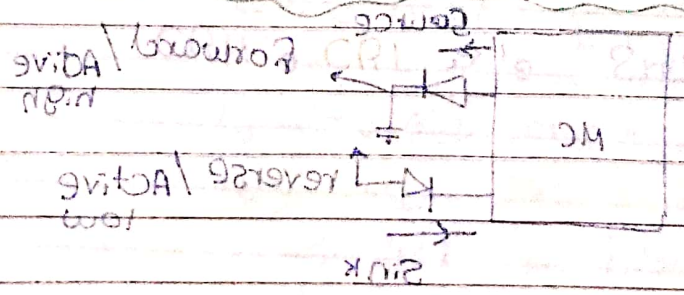
الميزة فيها إني وفرت فلوس وعملت Syncronization
لكن فيه عيب خطير هو
→ Dependency
انه معتمدين على ال Clock بتاع الأول فلو باظ هيبقى الـ
Sys كله باظ وهكذا التاني

CFRG

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



يعني AHB
تصبح Factor
من ال SysClock وبعدين من AHB بيطلع APB1 و APB2
يعني كدا هما الاثنين Factor من AHB
يعني مثلا لو حطينا أرقام زي الفي الباتا شيت على HPRE
فا معناه إن أي Peripheral متوصل بال AHB هيدخل
عليه ال Clock دي وكمان هتغذي APB1 و APB2

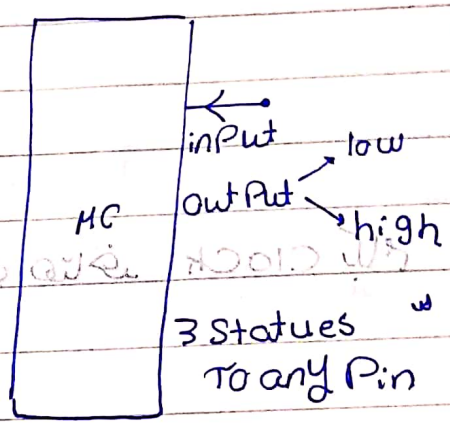


GPIO

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 الانقدر نطلعه لو output

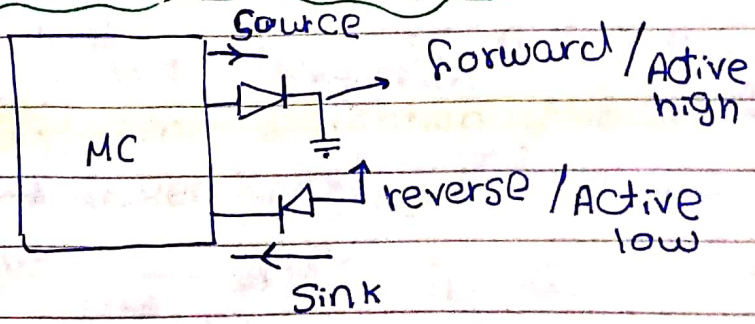
Sink Current : لو بتاخد current

* في الغالب ال Source بيكون أقل من ال Sink

الماتيريال كنترولر :

Source Current = 25 mA

Sink " = 150 mA



io Pins بتـ Source قيمه أقل من قيمه ال Sink

الـ Reversed أفضل

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

* كل C

Source Current = 0 mA

Port C بتعمل بس Sink

STM32F103	AVR
Data direction Registers	DDR direction
* each Pin have 4 bits for mode	PORT output regist
Port 16 Pin ⟹ 16 * 4 = 64 bit (16 يعني Mode)	PIN input registe

↓ bits

عشان كل 8 Pins عندي 2 Regs

CRL 0:7 — 32 bit — عبارة عن 32 bit

CRH 32 bit — 8:16

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

4 * 8 = 32

CRH ، كذلك 7 من (8 : 15) High

وكل Port ليهم 2 registers واحد CRL و CRH

*

4 bit Configuration for 1 bin.

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، مع Pre آخر

input

00 → Analog

01 → Floating

10 → Pull up or down ← يعني حتى لو متوصلش بحاجة فهي متوصلة من جوا

11 → Reserved (Not Allowed)

00 → input

01 → output @ 10 MHz (أعلى)

10 → output @ 2 MHz

11 → output @ 50 MHz

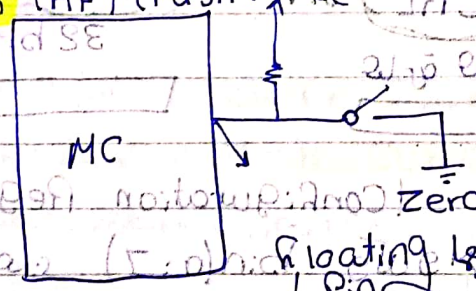
اهتماماً بال Power، فلو هقول output لازم أحدد ال freq

* 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

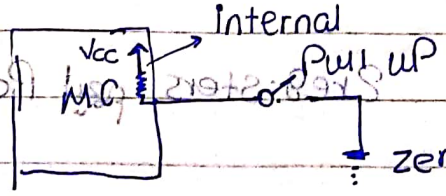


لو مش مناسب يبقى ال pin عليها floating

ولو مناسب عليها zero

وعلشان ميقاش عليها noise بنحط مقاومة

Sensitive to high & low



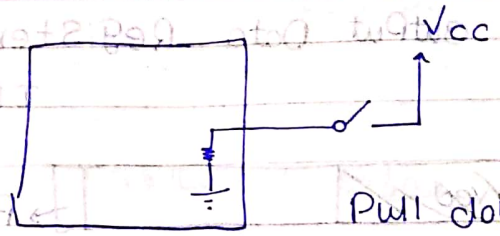
لو مناسب هتبقى zero

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



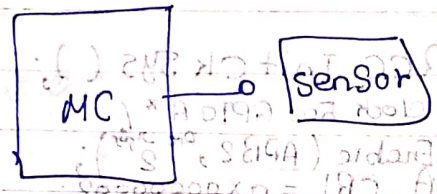
Pull down Pin
(sensitive to high)

ال low مش بيأثر فيها.

طيب امتى استخدم ال Pin floating أو pullup أو pulldown

فلو كان Pull أو down مش فارقة مع Switch

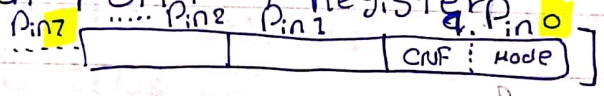
، أما لو مع Sensors هستخدمها floating



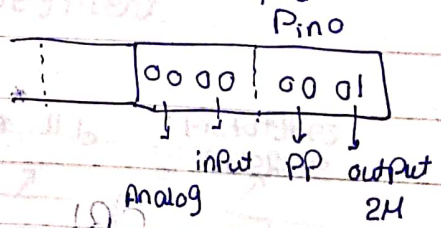
Each PORT 7 Registers

CRL
CRH
IDR
ODR

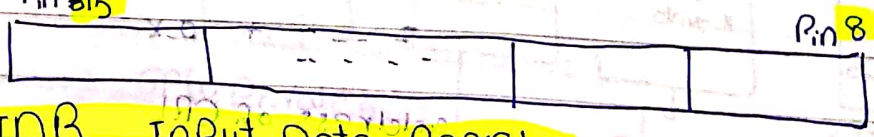
① CRL



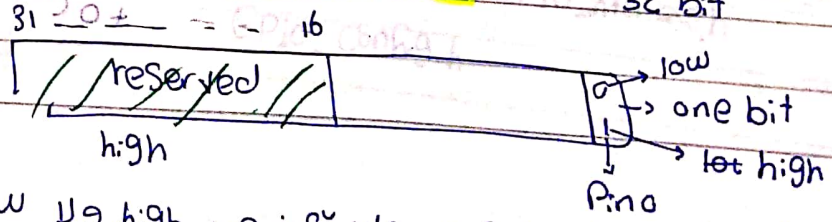
كل R فيها عبارة عن 32



② CRH نفسه



③ IDR Input Data Register 32 bit

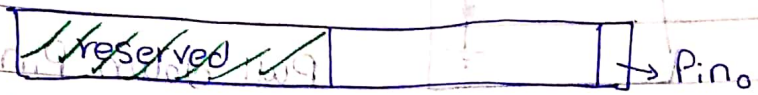


لو عندك Pit هي input و عايز تعرف هي high ولا low خش اقرأ ال bit

المقابلة ليها ، زي PIN R في ال AVR

4 ODR — output Data Register

عكس IDR



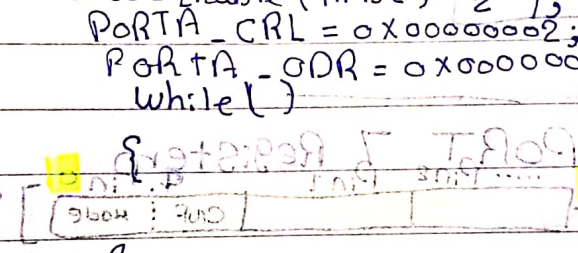
لو عايزها output وعايز أخرج عليها zero أكتب عليها zero
وادي الـ PortA في الـ AVR

عايزين ننور الـ led

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

وبعدين نكمل enable للـ clock على الـ Periph بتاعه
GPIO → APB2

① CRL



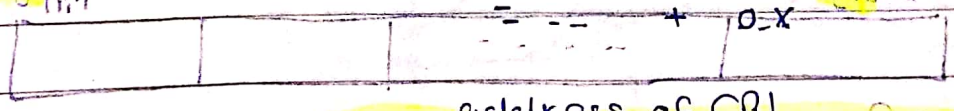
output 2M PushPull
01 00

* GPIO Based address
0x40010800

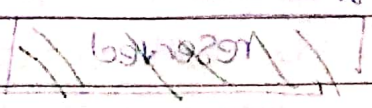
CRL offset 0x00 , ODR offset 0C

② CRH

∴ address of CRL
= ... + 0x



address of CRL
= ... + 0C



Eclipse

① SRC : بنحط فيه أي file.c

② System

فيه library جاهزة هو عاملها

include :

بنحط فيه أي file.h

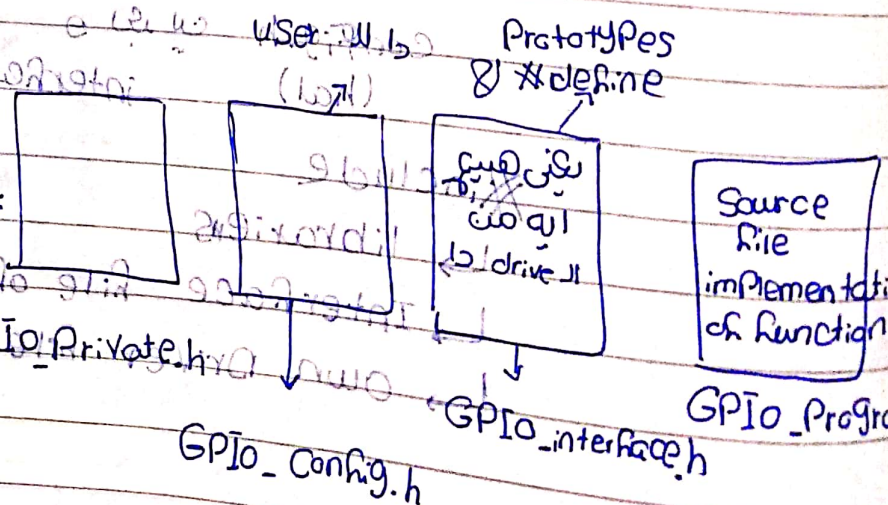
③ ldscripts : هنشرحه في الآخر

---

lec 6

GPIO Driver :

- LEDs
- Seven Segment
- Switch



56

أولاً :

1 أول فايل بنفتحه هو ال Registerfile.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 + عايزين نعرف ال registers الخاصة بال Ports A, B, C

ال Registers

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

5 نروح على Program.c ونعمل

1 #include to LIB

2 #include to interface from low layer (MCal)

3 #include to my interface (hal)

وبعدين config.h (hal)

6 أروح لل interface

#include

↳ libraries

↳ Interface file of lower layers

↳ own Driver file → ① interface

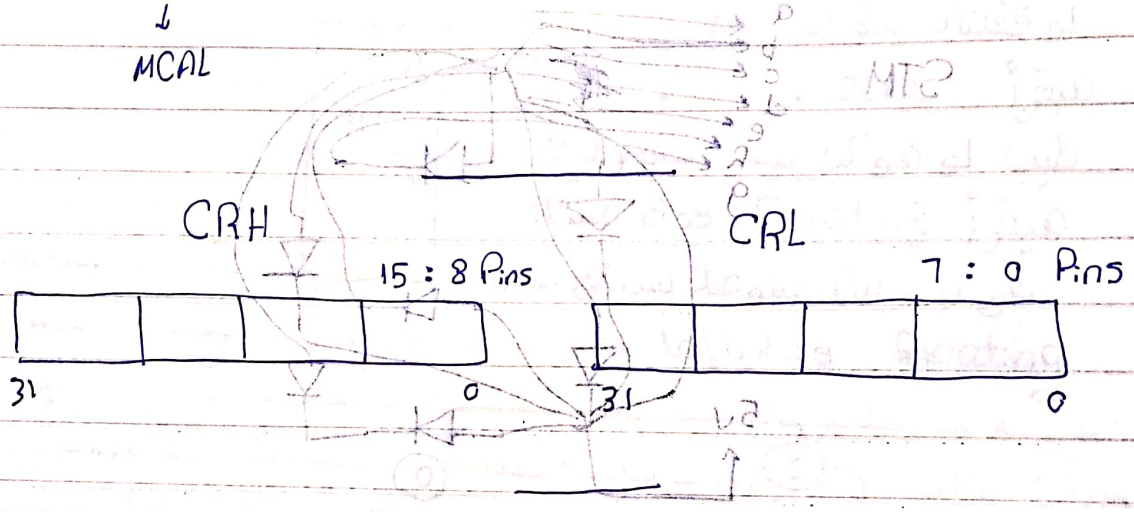
② Private

③ Config

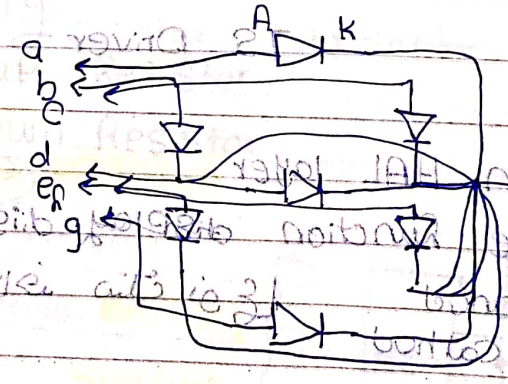
لو جينا في ال register ال اسمه CRL
هنلاقي هو ال 31 : 0
وفيه 4 bits للـ Pin من أول 0 : 3 هي 4bits
للـ Pin 0 وهكذا

① why Reset before set of register?

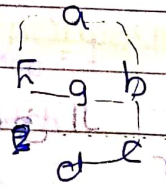
② μGPIO
↓
MCAL



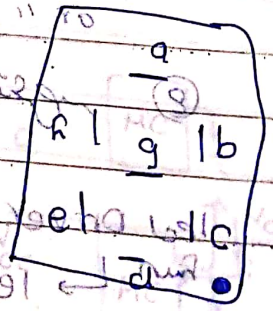
Seven segment
عبارة عن leds وهي وسيلة عرض للأرقام بدل ما كنت بعرض حروف



لو عايز بين رقم 9
يبقى هنور



1101111



لو عايز أستخدم floating number

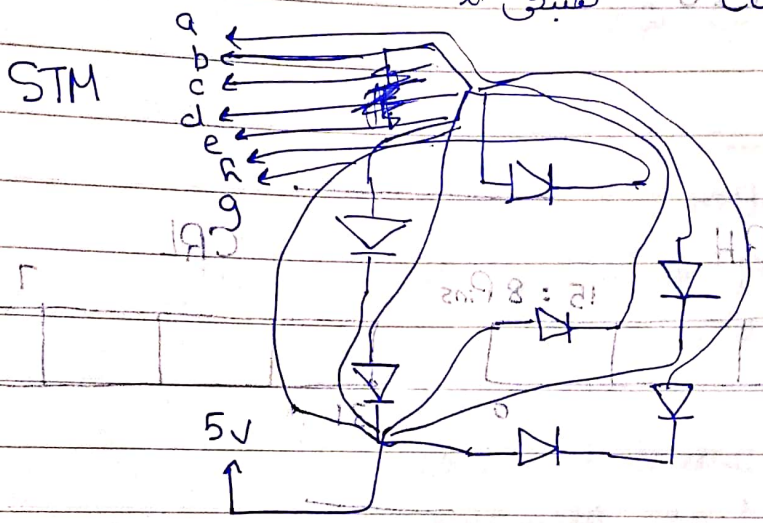
] هنوصل ازاى

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

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

وهنا ال بلنور بيبقى ال zero والمطفى هيبقى one

7S Driver

[1] It in HAL layer

[2] one function display

① COM anod هتاخد منك نوع ال

" cathod

② هتاخد بالرقم ال هتعرضه

وكمان فى ال HAL layer عايزين نعمل Driver لل

led on (Port

Task

① PLL ← هنديهالا 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 كده يعني short circuit

لو قفلناه يبقى ال MC هيقرأ zero .

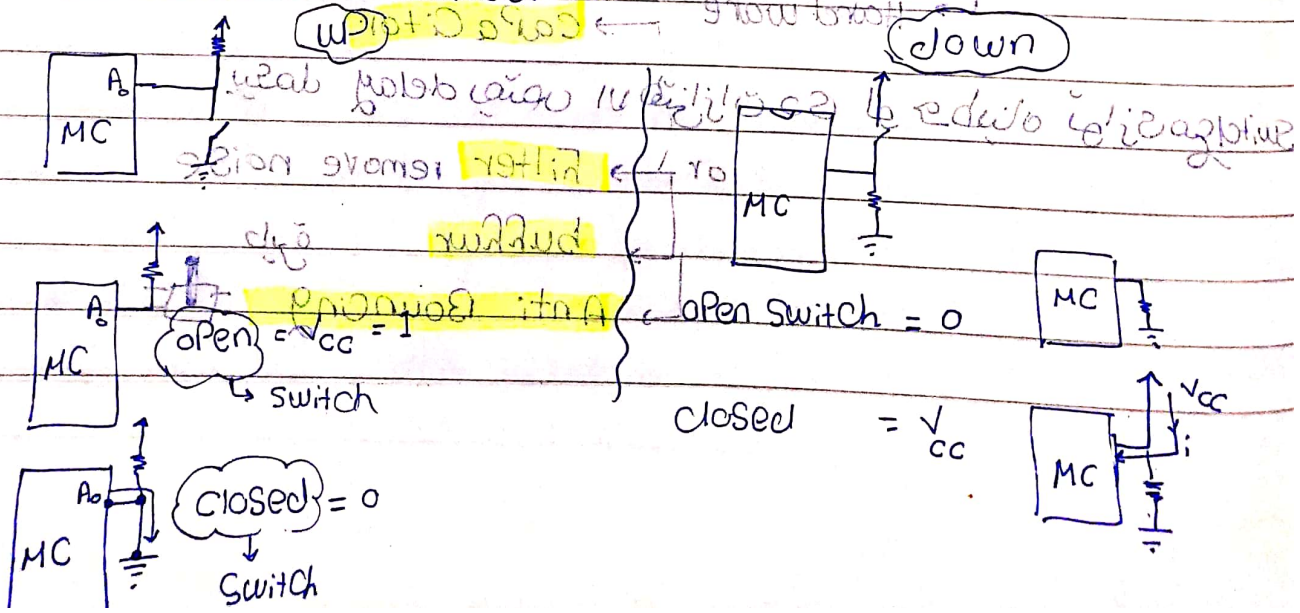
طيب لو هو كده مش مقفول هيقرأ floating وده مشكلة

يعني ال Pin متوصلة بس في الهوا floating وده يا إما 1 يا إما 0

بالشبة noise بأنواعها

طيب نعمل إيه لما نلاقي Pin floating في حاجتين

* floating
↳ Pull_up Resistor
↳ Pull_down Resistor

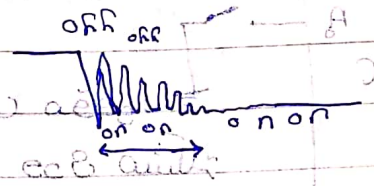


هنا بقى في الـ STM وفروا حاجه داخلية ، فهنوصل على طول على الـ Switch.

* لكن هنا في مشكلة تانية :

* Bouncing :

دي ظاهرة طبيعية موجودة في الـ Mechanical Switch وهي إنه مادة صالبة فلما تيجي تطبق السلكتين فوق بعض مش هنعرف فهيزل ويفضل يهتز



فلو مثلا الـ f بتاع الـ MC بساوي 8MHz يعني يقدر ينفذ 8 مليون أمر في الثانية الواحدة يعني يقدر يحس بـ 8 مليون زبذبة ، لكن ممكن الاهتزاز دا يوصل لـ 8 مثلاً فهيحس بيه الـ MC زي ما

* يعني أنا وأنا بدوس على Switch ١ مرة واحدة بس لكن بدوس ٢٥ دوسة

طيب نعملها إزاي؟ ليها حلين :

- Software → (delay) → 10 ms حد أقصى
  - if(switch closed)
  - delay(10 msec)
- Hard ware → Capacitor

بيعمل delay يمتص الاهتزازات دي لو حطيناه توازي مع الـ switch

or → filter remove noise

- buffur طايرة
- Anti Bouncing

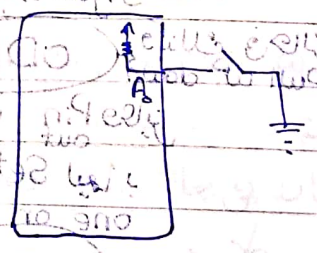
طيب هنا هنستخدم ال SW ولا HW

* على حسب ال APP :

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

ولو مش ذي التكليف يبقى SW أفضل لأن مش هتحرق لك السكرق.

* هنوصل ال Switch (Pull up)



Value	Action
0	Pushing
1	Not pushing

طيب لو Software MC (Pull up)

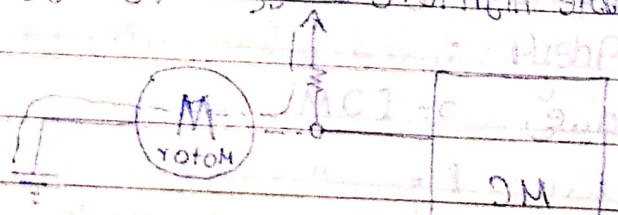
config ← mode

if Pull down → ODR = 0

Pull up → ODR = 1

مش صح لحد المحاضرة الجاية

Software code. MC End.



1 open drain output:

Conf	mode
0100	01

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

CRL
CRH

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

ODR

وكذلك في حالة down ← Pull لو input *

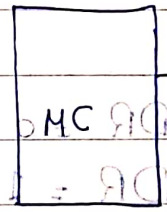
* ولو عند Pin out وعايز أعمل Set ليها بـ one or zero

IDR

بتاع input وعشان أقرأ Pin ← input

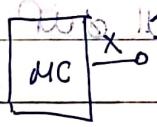
يعني كأن ال Pin دي disconnect من على ال MC

open drain Pin



لو عليها zero فكأنها واصلة على Gnd

ولو عليها one كأنك قطعت الـ bin دي من على ال MC



ده ميزة كبيرة جداً:

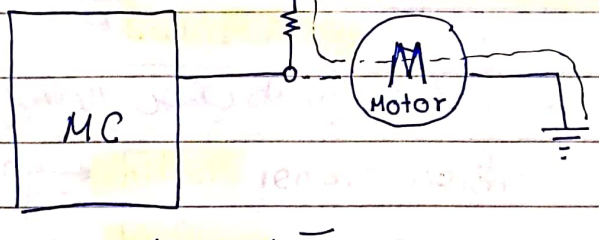
وهنا ال open drain ليه استخدامين مهمين جداً:

1 ميزة Configrable high leve V_{cc} voltage

0 → 0 V ⇒ off

1 → floating Pin

فكأنها disconnect فبنجيب مقاومة على V_{cc}



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

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

إنت الـ Motor لما تكتب 1 أنى volt هيدخل على Motor ودى اسمه:

Configurable high level voltage

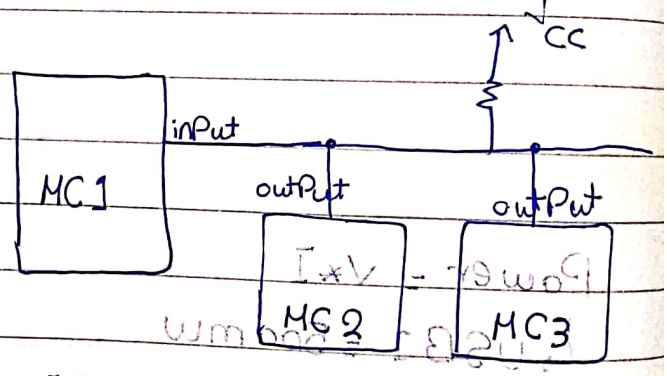
ودا عن طريق أوصل مقاومة على الـ volt العايزة وأكتب one على الـ Pin.

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

note [2] تانى ميزة

open drain Bus

بيستخدم فى حالة Communication



واصلين على نفس الـ Pin وبقدر يبعتوا داتا لـ MC1

* أول حالة فى مود PushPull:

MC2 , MC3 = 0 ، فهيبقى MC1 = 0

" " = 1 ، " = 1

لو حد فيهم zero والتانى 1 ، هيحصل Short circuit ، هتولع

* هنا بقى لو الـ 2 Pins دول open drain:

MC1 → 0 =

0 0

1 1

يعنى الاتنين بيفصلوا نفسهم عن الـ MC فال Pin دى floating

فبعد فتوصل ال pin دى على ال V_{cc} .

* هنا بقى لو حد وصل zero والتانى one

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

وال حصل منعت حصول ال Short Circuit

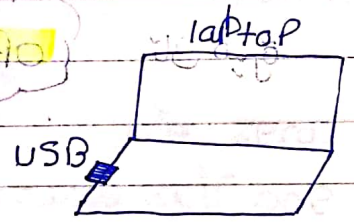
وظهر عندى ال Consept بتاعه

① Dominant bit → 0 مسيطر ولو واحد

② Recessive bit → 1 لو كلهم كتبوا one

تانى ملاحظة :

note [2] LED Power :



Power = V*I

↳ USB1 = 500 mW

↳ USB2 = 500 mW

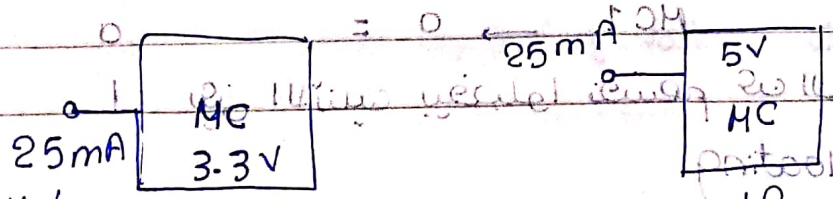
↳ USB3 = 750 mW

		default	Max
USB 1	5V	100 mA	300 mA
USB2	5V	100 mA	500 mA
USB3	5V	150 mA	900 mA

② LED : Max Power : = "

↳ Range 40 mW : 160 mW

آخرها 160mW يعنى كلهم هتحرق عشان USB

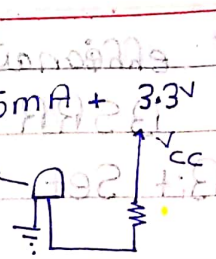


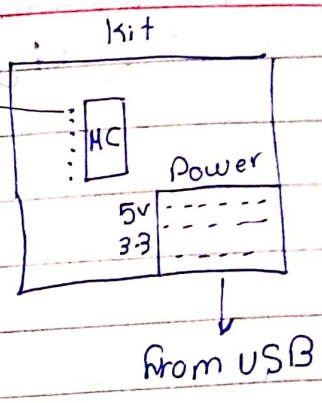
غالباً ال Max level

outPower = 3.3 * 25 < 160 mA

outPower = 5 * 25 = 125 mA

فلو كدا ال led مش هتحرق 25mA + 3.3V





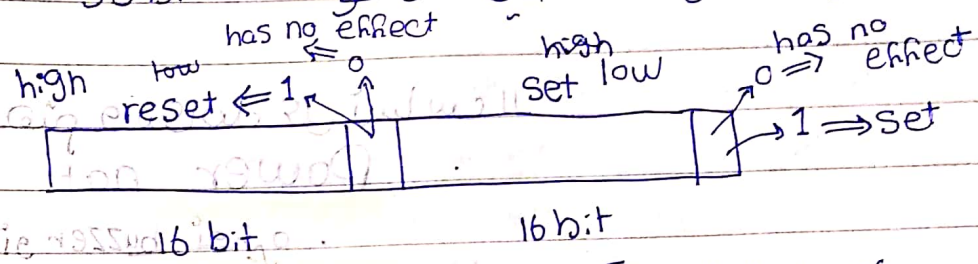
فاحنا نهتم ونتعامل على أساس ال
Power not Volt
مع led أو 7S أو buzzer أو غيره.

note 3 BSRR , BRR

2 ريجسترز في ال GPIO لكننا شغلناه من غيرهم
فهما بيقدموا حاجة زيادة عايز أستخدمها تمام أو لا تمام.

* مثلاً عايز أعمل Set Pin في ال MC
فبعمل
ODR | = 0x01
لو أنا قولت كدا ODR = 0x01
دا غلط لأن كدا بعمل SetPin و أعمل Clear للباقي
فعلشان أعمل صح يبقى OR
ODR | = 0x01
لو بال assembly ممكن ياخد 3 سطور.
1- (Read) ODR
2- change bit
3- write back value to ODR
طيب أنا عايز bin

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 نرى الـ ال أعمل ايه؟

BSRR = 0x01

من غير أن أأثر على الباقي لأن 0 هنا (has no effect)
وبكدا كتب one على Pin 0 في instruction واحدة بشكل مباشر
فكدا أقدر أسرع في عملية الـ write 0x0 وكذلك جزء الـ (reset) جزء جزء

example

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

one instruction.
أسرع

طيب هاي اعمل reset

```
void ResetPin(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

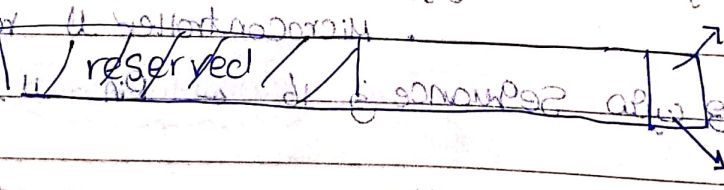
set has higher Priority

فكدا ال set ال هتنفذ وال reset لا

⇐ فدوره set و reset بشكل مباشر

① BRR: Bit Reset Register

32bit



*وهنا حلينا مشكلة إنك اجمع 16 على الـ Pin

علشان أعمل reset

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

Cases.

Example

↳ ① set Pin 0 & clear Pin 1 → BSRR

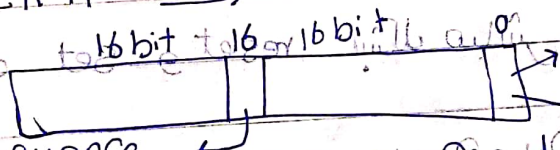
↳ ② clear Pin 0 & Pin 5 → BRR الأفضل

• ↳ ③ write value 0x3421 on PORTA ⇒ ODR

لأن هنا هقول القيمة دي بتساوي كدا الـ ODR مش
هنشوف عدد الوحايد بقى وكلا نزى BSRR و BRR.

Challenge *
speed up set Pin function
or 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 للـ Microcontroller.

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

* يبقى كدا الريجستر دا LCKR بيعمل lock لأي Pin
في CRL و CRH مش عارفا تتغير كل Pin ليها onebit
في الـ register وكل Port ليه LCKR Register.

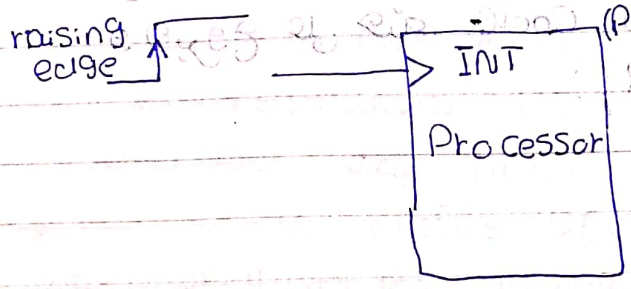
مطلوب: Function في الـ DIO تاخد رقم الـ Pin وتعملها
واسطة lock.

Interrupts

أولا: بشكل عام :

وهو معناه إني بعمل حاجة وفي حاجة قطعتني زي مثلا بالله وتلفون
ك سواء هترد أو لا.

* أي Processor في الدنيا عنده Pin باسمها INT Pin
وهي بتكون أول ما يبعتي عليها (Pulse) raising edge



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 يبقى إن فيه
event أهم من الـ بيعمله، فيبدأ يـ act عليه.
وهي 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 الـ بتعمل INT

Internal INT → 3 أنواع جوا الـ Pr بنوع عملوا INT ؟

*MC Peripheral
1) UART
2) ADC
3) Timer

Core Peripherals (عاطرنا عملتها في اللود)

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

Software Interrupt — وهي عبارة عن assembly instruction — SWI
ودي لما الـ Processor بيلاقيها في أي وقت بيقول أنا هعمل interrupt وينفذ لود الـ INT.

What happens when Processor got interrupt ..?

يبدأ ال Processor

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

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

```
main()
{
  =
  ISR
}
```

```
ISR()
{
}
```

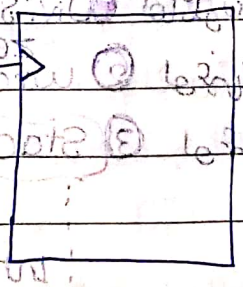
يبدأ ينفذ اللود بتاعها ويخلصها ويرجع لنفس المكان الوقف فيه
سواء ال ISR دي جالي عن طريق hardware signal على
بناية INT بتاعته أو من Code Periph أو من سواء
اللود الكان بينفذه عمل لقى فيه Trap أو لقى instruction
اسمها SWI.

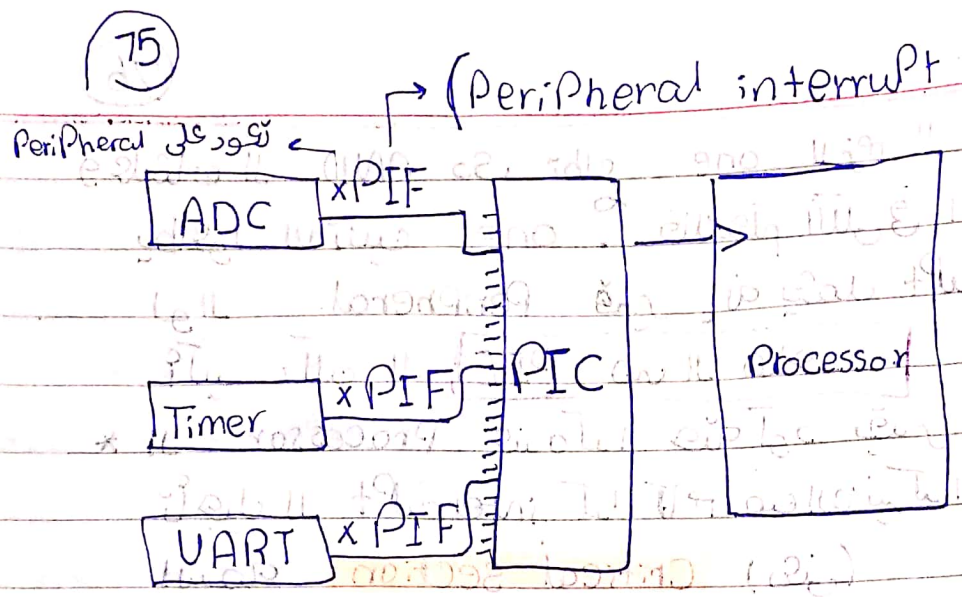
يعني ايه يعني SWI؟

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

Interrupt Handling Techniques:

ازاي كل ال Peripheral هتوصلي على نفس ال Pin
INT بتاعي





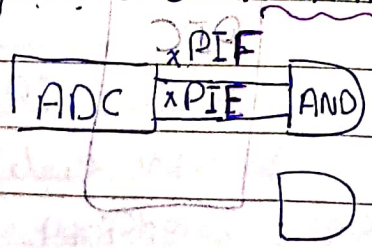
PIF : السلك دى بتبقى بـ one لما ال Peripheral حاجة يعمل interrupt ، مثلا لما ADC يخلص conversion ال PIF يبقى one ، وال Timer يعمل overflow ال PIF بتاعته تبقى بـ one وهكذا

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

طيب انا كل أوصل Peripheral كبير بـ Pin واحدة هنا هنحتاج Peripheral فى الشكل اسمه (PIC) (Programmable Interrupt Controller) ، فكدا لما حد فيهم يعمل interrupt يبقى الـ PIF بتاعه بـ one فال PIC هيبلغ الـ signal دى للـ Processor

طيب افرض إنا 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 ،

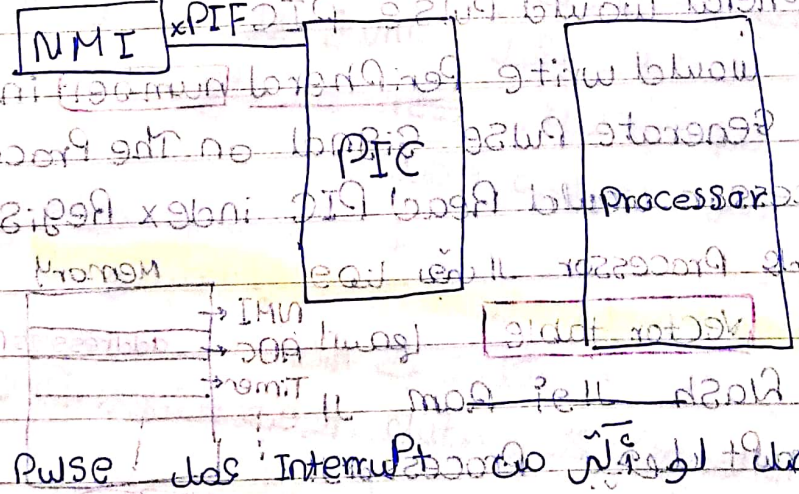


وكدا خلت كدا PeriPheral يقدر يعمل INT على ال Pin الواحدة الاعلى ال Processor.

إلا ال PeriPherals معينة وهى كالعادة عندها xPIF وداخل direct على ال PIC من غير ANDGate ولا ليها GIE ولا PIF وده اسمها NMI (Non Maskable Interrupt) اما تيجى مينفعش أقوله متحصلش زى مثلا ال reset وال reset دى PeriPheral بيأبلاى Signal على ال Processor و بيقوله سيب الدنيا وأعمل apply على الكود تانى ، فمش منطقى ال Processor يقوله لا انتظر، فعلشان كد متوصل direct على ال PIC

* لحد هنا كل ال Processor متفقة 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] ليه بقى ال هيحصل لو أكتر من Interrupt عمل Pulse فى نفس الوقت

تعالوا نجواب سؤال تانى الأول هو لما ال PIC يبعت Pulse لل Processor ويقوله خلى بالك فى interrupt حصل ازاى ال Processor بيعرف أن ~~ker interrupt~~ ال PeriPheral عمل interrupt؟

ماهو برده الكود الجاى تعمل لما ADC يخلص conversion

غير اللود الجاي تعمله لما UART يخلص recieve ؟
في 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

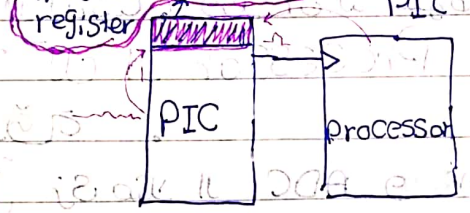
1) **Vector table Based System.**

دا نوع من ال Processor

← بيعمل ايه بقى؟

بيكون عنده register اسمه Memory location

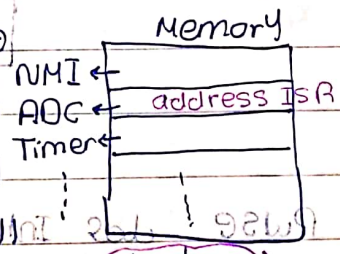
← Interrupt register index دا في ال PIC



*** 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 اسمها Vector table ممكن تكون حته من ال Ram أو ال Flash ، هنا أول ما ال Processor بيجهل interrupt بيقرأ رقمه وبيعمل jump على حته الميموري دي



وهي عبارة عن جدول locations بنفس عدد ال locations اللي في ال PIC وبنفس ترتيبها، كل location فيها ابن responding ل Peripheral معين.

5) Processor jump on vector table to corrosponding 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 وإنت بتبني build
اسمها linker هي اللي بتقرر كل function عندك هتنزل فين
في الميموري ومش مكان ثابت
بس ال location اللي هو اتحط فيه ، فيه عقوبة
بيجي هيشوف كل function متدول 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 هيعمل jump
عليه هيلاقي الكود هينفذه وهكذا

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

PIC
latch memory

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

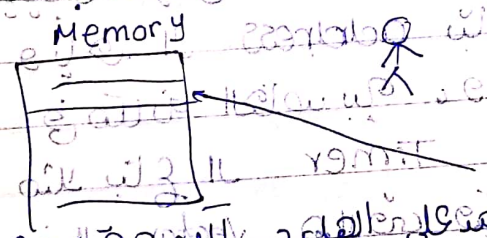
ال PIC قال هيبقى عندى حته مبعوى كدا اسمها latch memory لو حصل أكتر من interrupt هعملهم save في ال latch دى وأطلعهم واحد واحد . طيب هنطلعهم بأنى Priority إيه PIC ، قال أول واحد فوق هو أعلى Priority وال جاى بيقل . والترتيب بتاع ال Peripherals الاعلى هوال بيمثل ال Priority يعنى على حسب رسمنا إن NMI أعلى وبعده ADC وبعده Timer وهكذا وبالتالى ال Priority بتاعه ال INT بتاعه بتاع ال MC التركيب ال Peripheral سبتعتمد كليا على المصنع

∴ أول نوع ال هو vector table ... سهل جدا

سؤالين :

① فين ال vector table في ال flash ولا RAM ؟

② مين الكتريت ؟



لو حصل كدا ال INT ؟

عيوبه ان ال هينفذ الأول معتمد على الهارد ولكن مقدرش أغير

2 Flexiable Priority system

النوع التانى

بيحصل إيه :

لما يحصل INT ال PIC بيقول أنا معندش index register ~~أنا عندى معلومة أقولها لل Processor أنى Peripheral~~ ال Processor كدا فى ورطة ! . هيعرف ازاى ؟ كدا ال Processor قال أنا كمان هلغى ال vector table وأياً يكن ال Pulse ال هتيجى 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

الـ address بتاعها موجود في الـ interrupt vector

فالـ Processor بيـ jump to ← interrupt vector لـ find

الـ address بتاع ISR ؟

إنت بقى كـ developer إعرف الـ INT بتاع

أي Peripheral فتعالى نقول ISR {}

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

وخلي بالك لسه فيه PIE و ANDgate موجودة

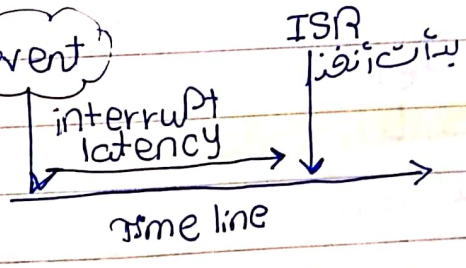
طيب استفدت ايه ؟

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

كدا بقى أنى أفضل vector table or Flexable Priority ؟

vector table (ATMel AVR)	Flexable Priority (PIC Microchip)
* fixed H.W Priority	* Software Priority ①
* low interrupt latency (fixed) ① وهو نفسه لك ... Peri	* high latency & variable.

تخيل هنا



INT ← generate ال event 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
وجدا هيزود ال latency

طيب الـ ARM بقى :

قال أنا هنبني الـ تنين وهعملك اختراع عظيم جداً

اسمه NVIC ← ودا بتاع الـ interrupt

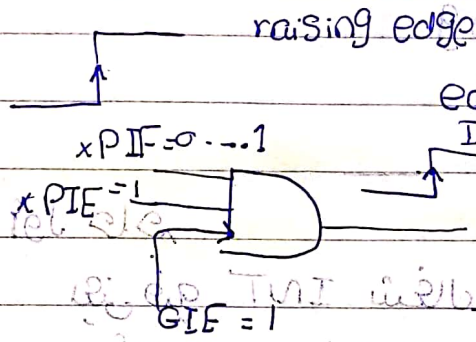
ARM.

وكدا احد دلوقتي دا الـ consept بتاع الـ INT لأي MC

[4] Flag Clearance :

raising edge لازم يحصل عليها INT Pin ف

يعني edge sensitive



هنا مثلا الـ Timer عمل INT

والـ PIF بقى Tone

وخلاص الـ INT اتنفذ الكنترول الـ flag

لسه one ، فلو جه INT تاني هو مازال one افضل

هيحصل INT ، فلازم نعمل clear

علشان أسمح إن لو في INT جديدة يعمل edge جديدة

وأقدر أحس بيه

* لازم نعمل clear للـ flag بعد تنفيذ الـ INT :

* عشان نعمل clear فيه 4 طرق :

كل MC هيختار 2 أو 1 يعمل بيهم clear للـ flags بتاعه

[1] By Hardware, once ISR executed.

[2] By writing zero to the flag

[3] By writting 1 to the flag

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

x = x PIF ;

5 Interrupt Nesting :

وأنا شغال في INT حصل INT تاني :
يقطعني ولا لأ؟
لو قطعه اسمه nesting ولو مقطعهوش ميبقاش nesting
وأغلب ال Processors مش بيـ Support ال nesting

* طيب ال Processor ال بـ Support ال nesting
بتفرق بين نوعين من ال nesting :
* Normal nesting
* Self nesting

1 Normal nesting

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
فكدا في حالة ال Normal تحديداً لازم نسأل عن ال Max level

2 Self nesting :

↳ INT 1 is executed, INT 1 happened again
يعني مثلاً ال ISR بتاع ال Timer لسه مخلصش
راح ال Timer حصل over flow تاني، ودي حاجة وحشة
بتبوظ ال design وبتاعك.
finished general Interrupt.

سؤال عن Save context

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

→ assemply → Context Switching

Push x

هنا ال Compiler بيعمل Save للكود

The NVIC

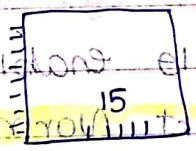
Nested Vectored Interrupt Controller

↓ vector table

1] Core Peripheral ⟹ ↓ latency + Standard (ARM بتاع)

2] Support 255 interrupt

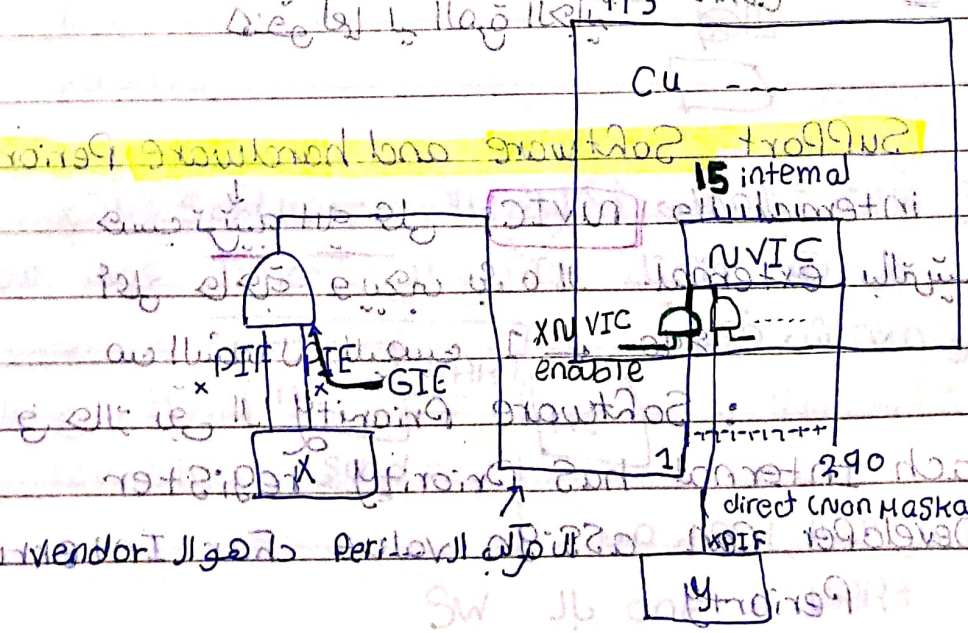
240 (Processor برو ال external) — 15 Internal



255 → 240 Ext, 15 Int

3] Vector Table Based System.

ARM Cortex M3



ال Peripheral دي هو ال vendor

هنا بقى NVIC قال أنا معندش الكلام دا
أى Signal طالع من دول (Peripheral) قبل ما تدخل
على ال NVIC بتدخل على AND Gate ، داخل عليها ال Signal
الجاية من ال Vendor وال Signal بتاعه NVIC enable
وبتبقى يا 0 يا إما 1 ، وفيه enable لكل سلك
* وفى 2 enable :

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

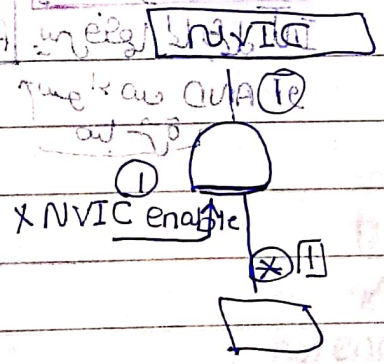
ميزة خطيرة (6) Support Software and hardware Periority
حسب ترتيبه ايه على NVIC وال internal
أعلى حاجة ويجى بعده ال external بالترتيب كدا
من الشمال للبصيت 1 ← 240
* ، فى حالة بقى ال Software Priority :

* for each internal has Priority register
* SW Developer can assign value for Interrupt Periority

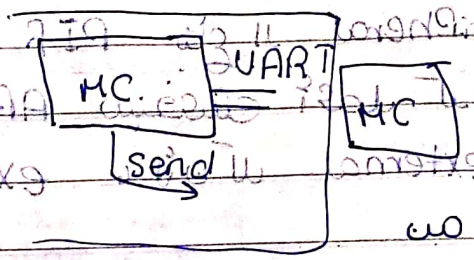
لو 2INT حصلوا في نفس الوقت
ال NVIC هيروح الأول يبص على ال Software Periority
ال dev... قررتها ويشوف فيهم مين عنده S.W Priorty
أعلى يعني قيمة أقل
0 → highest
255 → lowest
وكل واحد فيهم (INT) ليه Regis. مخصوص، هيبص على ال
registers بتاعتهم، ~~شوف كام إنت~~ ويشوف كام إنت كاتب
مثلا في Register 5 والتاني 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 مميز
set flag for each interrupt.



هنا بقى قالك أنا هعمل أخليك بالـ Software تكتب : one
هنا على ⊛ فتمثل وكأن ال Timer عمل overflow
وهو مش عملش فعلا



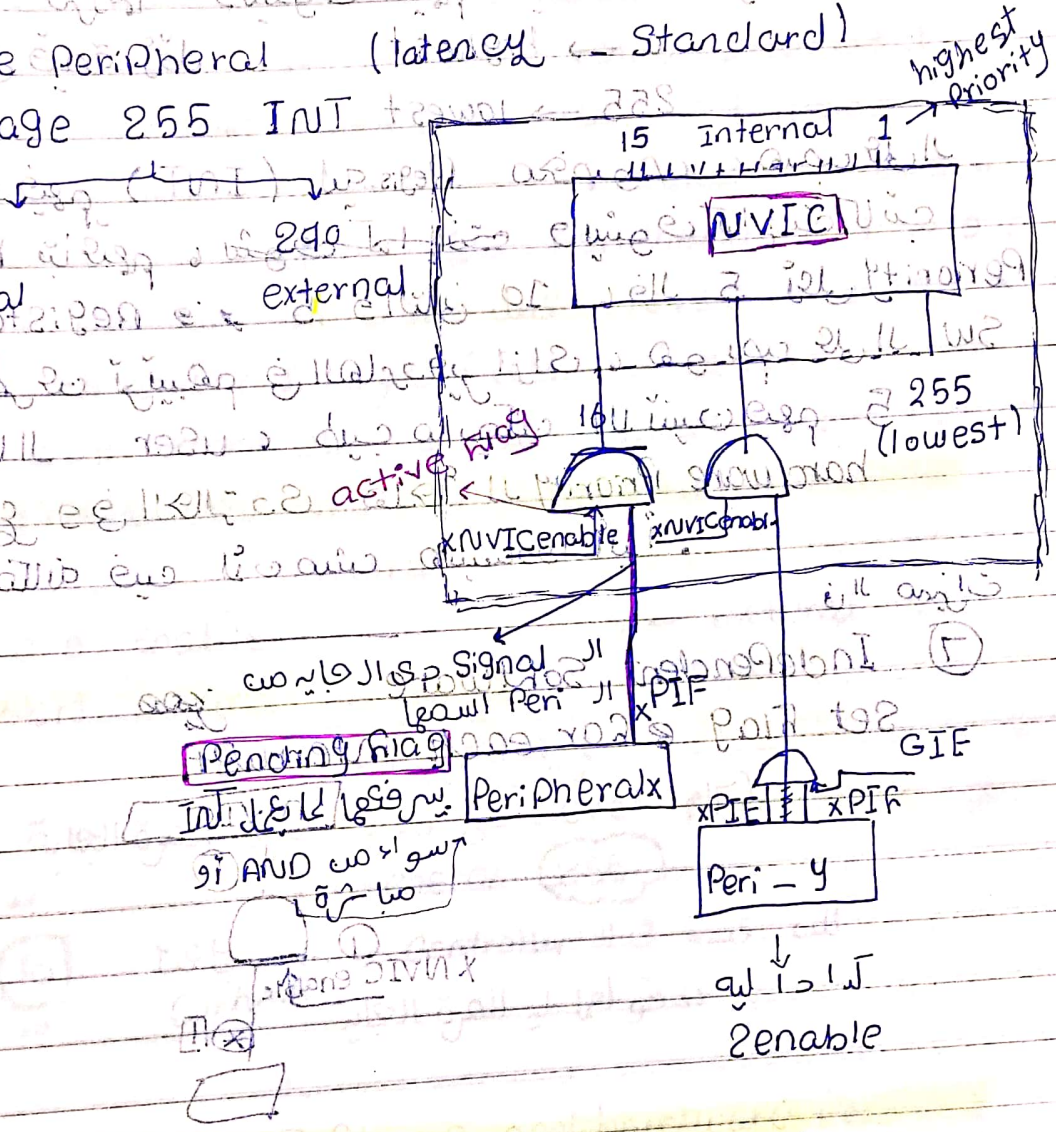
فهتكتب على السكرين الجاية من
ال UART ← one بال SW

End of lec 7

Lec 8

NVIC

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



سؤال: كل الـ external Peripheral are Masked ليه؟

الـ external Peripheral مش بتوصل الـ PIF على الـ PIC مباشرة لأن ممكن يكون في غلط في الـ in general

* لكن في ARM مش بتعمل كده، لأن تقدر تـ Mask الـ external أو enable له external خاص به

نافع معلومة

③ each External interrupt has Two flag

flags → Pending flag

→ Active flag

مش باين قدامنا

لزمته لما ال INT بيرفع ال Pending flag بتاعه ، فإما هنا ال NVIC يدخله لل Processor وينفذ وساعتها بنسميه Active

أو ال NVIC مش هيدخله ينفذ، طيب ليه؟

* لأن إما disable من عند ال NVIC فمش هيقدر

* أو إن في INT أعلى منه في ال Priority شغال فيقدر

* 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 → notenable / higher Priority interrupt is excuting
1	0	State : Interrupt is executing
1	1	State : Interrupt is executing and another version of it fired (معناه إن INT حصل) → Same interrupt is excuting (not self nesting)

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

interal external

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

يعني أي INT عايز يدخل
ال execution يعدي على NVIC
وال NVIC بقى على حسب هيعديه ولا لأ
فدا ال Manage
(Pass, not Pass)

* فأنا كأني لل internal بنظم المرور بس
* طيب هل مقدرش أعمل Clear و Set وهكذا لل internal؟
لا يتحكم بس مش من خلال ال NVIC.

(ex) SysTick (Timer, core Peripheral)
هنا ال NVIC مقدرش يعمله Clear و Set
ولكن ال SysTick نفسه أقدر أعمله

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

(Software Priority)
يعني Register لل INT بتبقى قيمة للـ S.W Priority بتكون من 0 : 255 ، وعند H.W.P على حسب ال Postion
فاعم على NVIC لل ما كان أقرب يبقى أعلى (Priorty)
، ال Positions دي مش لها أهمية إلا لو أكتر INT حصلوا مع بعض أو واحد حصل أثناء ما واحد شغال

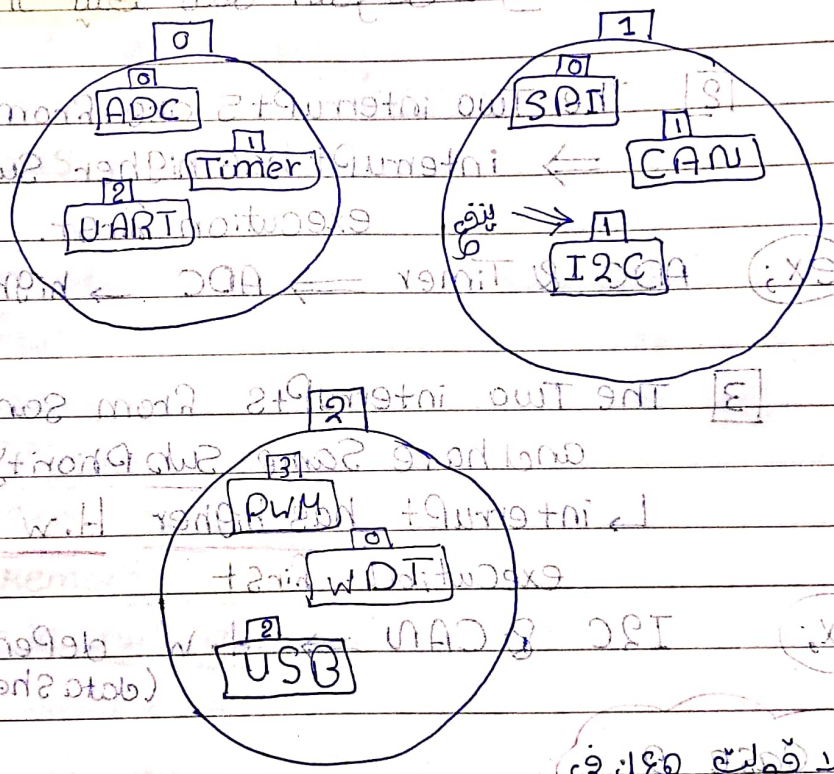
6 Nesting, NVIC supports Interrupt nesting (normal nesting)

no self nesting,

* Interrupt Grouping → خيالية Group Priority → Sub Priority

دا الهيخدمنا الـ nesting دا عامل ازاي:

example:



أنا مثلاً قولت هعمل في الـ System بتاعي 3 groups هجوا كلهم جروب أنا هختار الـ Peripherals اللي فيها Interrupts جوا كل group بتاخد Priority

يعني مثلاً الـ Timer بتاعته نفسه Priority وال Priority بتاعته نفسه هيتحسب الجروب Priority

و هنعتبر إن الـ Peripherals اللي في نفس الجروب إخوات

والإخوات ميقطعوش بعض يعني مفيش nesting

الـ يقطع الجروبس المختلفة

Case 1:

⟹ Two interrupts happened at same time:

1. The two interrupts are from different groups

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

* ماداام هن جروبين مختلفين يبقى ال عنده Group Priority أعلى هو ال يتنفذ بدون 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

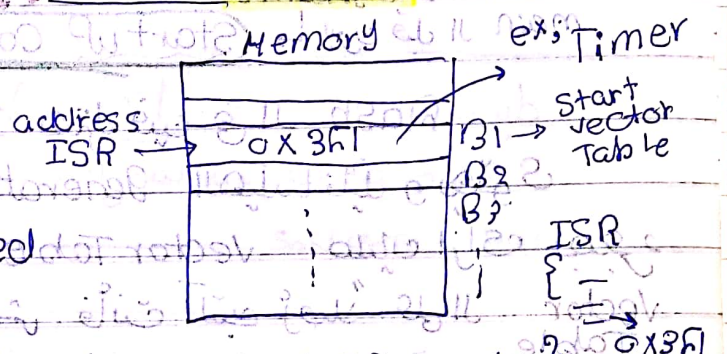
* How to disable nesting ARM Cortex M?

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

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

* Maximum level of nesting
اله علاقة عدد الجروب بال nesting
((مساوي))

Vector table



Facts:

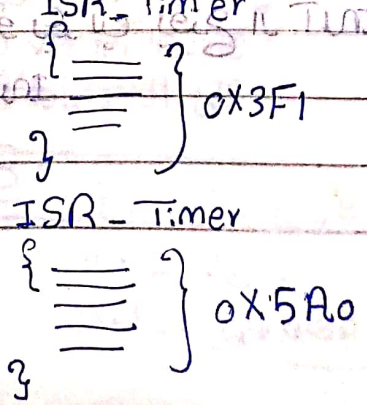
① vector table has fixed location.

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

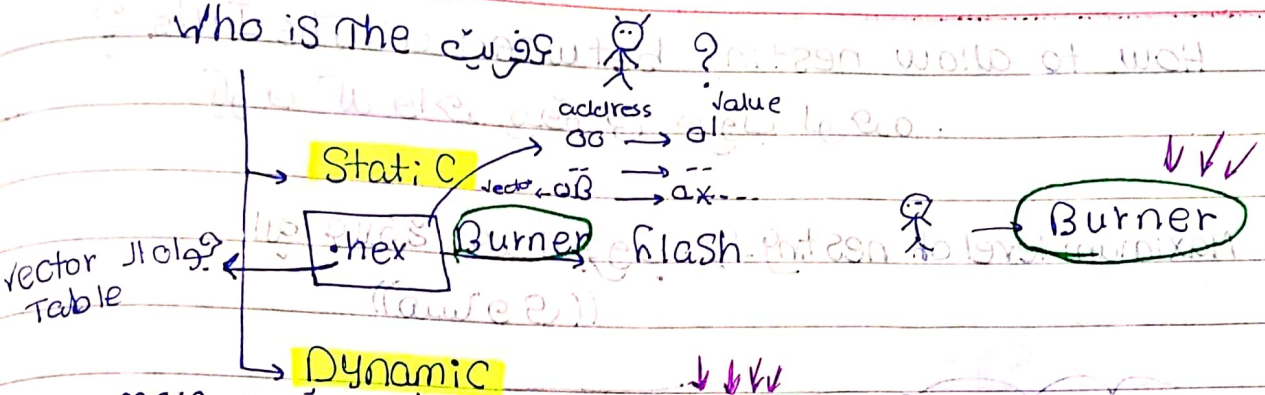
Vector Table

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
وهو ممكن أغير أني بستخدم



Who is The عفريت ?



* معلومة :

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

② * هنا وظايف ال startup code :

إنه يكتب على ال vectortable ، 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 بقى ال startup code قال هيعمل

disable all interrupts

initialization علشان يعمل

vector table وبعدين يفتح ال INT

تاني ، أحسن ييجي INT

كدا ال Static هو الجاهز إنه يستقبل INT في أي وقت
لأنه مكمل initialization قبل ال runtime بواسطة
ال (Burner)
أما ال Dynamic فهو عيب إنا 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 مبيرفش يشتغل Dynamic؟ لسه هنشوف

Software, End

معلومة :

① Private.h & Config.h مفيش Component تاني
معايا في ال Proj مملكت يشوفهم ، ال interface.h هو الوحيد اللي الناس البره
تشوفه

② Private.h مينفعش البوزر كان يعدل فيه بالكن Config.h
يتفع يعدل فيه

NVIC

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

Register Definition
& addresses
at ARM.

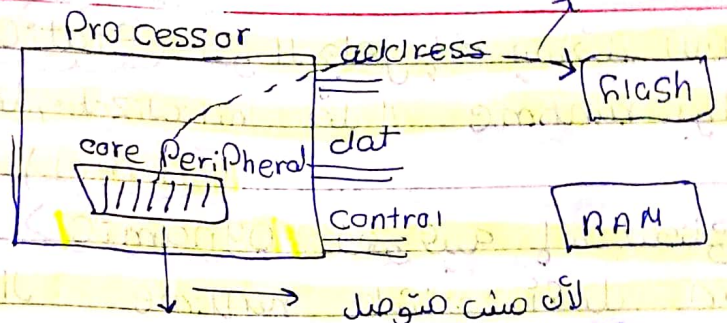
at data sheet — ST

Settable : Can change
Priority

Hard fault : Trap

نسخة تانية يعني (Mirroring)

Memory Mapping



each location address

لأن مش متوصل عليهم buses

بوصلهم بال assembly

* each location has Assembly instruction "index"

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

طيب هنعمل ايه هل هنكتب assembly ؟

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

Memory Mapping

وهي إني هجيب الـ addresses بتاعة الـ Peripherals الداخلية وهعملوه Mapping على الميموري برة

* وأخليه ياخد address من على الـ address bus وأكلمه بالسي عادي .

هل ARM هيعمل كدا في كل الـ Core Peripherals

قال لي بعضهم بس وبعض الـ registers بس

هنا بقى NVIC ← Memory Mapped

بالسي بقى

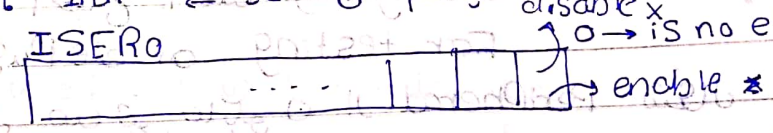
لأن ليه addresses ، لكن بعد كدا هنلاقي البعض ملوش assembly وهنكتب

→ NVIC Registers

1. ISERx → Interrupt Set Enable Register

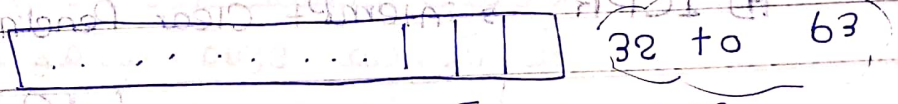
32 bit ← يعني بيتحكم في 32 ← INT ، NVIC بيتحكم في 240 بس

فكذا بيتحكم في ال external INT

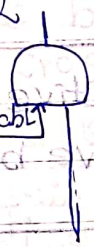


كل INT ليه bit لو 1 بيقي enable

$\boxed{0 \text{ to } 31}$



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

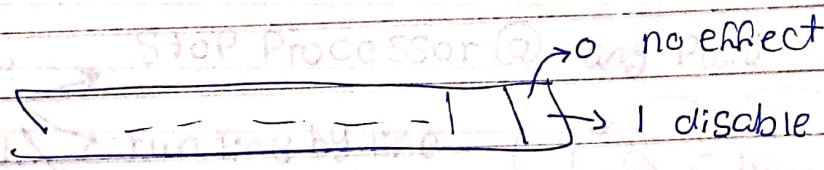


ال ISER هو اللي بيكتب على ال Pin دي.

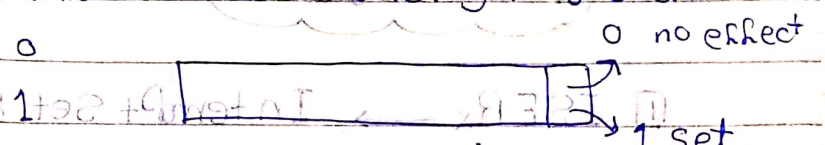
* هنروح بقى على ال memory map علشان نشوف ال offset بتاع ال register كام.

---

2. ICERx → interrupt clear Enable R..

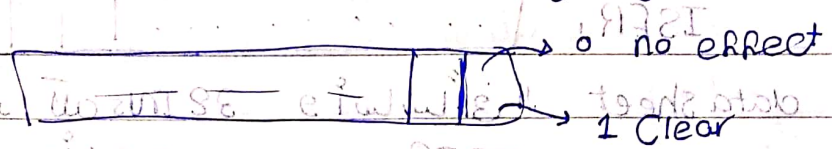


3) ISPR → interrupt set Pending Register



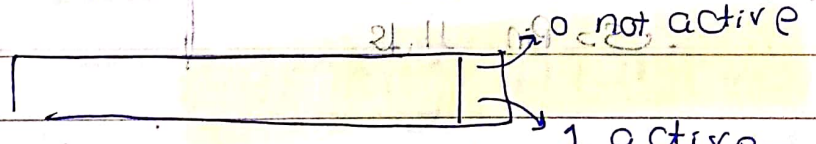
? Set to Pending flag لية ممكن أحتاج أعمل
For testing or debugging
يعني من غير ما يكون ال Peripheral شغال

4) ICPR → interrupt Clear Pending Registe



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

Read only



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 سبب 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

ممكن متنجحش فظهر

Debugger ← Tool تساعدنا أكتر

H.W & S.W Tool هي

Processor support debugger

ICD

Debugger بيوفر ال ICD

debugger H.W Tool

debugging Tool

open OCD ← Tool ودي بتركب على ال eclipse

Stop Processor @ any Point (Plugen) اسمها

run line by line

Read Register

Read/W Memory location

لما أنا دخلت جوا ال Processor

لازم تتعلم ال Skill

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

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

هتخلينى أعرف أتحكم فى الـ Processor.

* أى berner مش شرط يكون بيـ Support الـ debugger
لكن بتاعنا شغال

Lec 9

في ٣ طرق نبني بيها ال driver بتاعنا

1. #define
2. Struct
3. union

→ Priority :

↳ group + sub group

255 INT

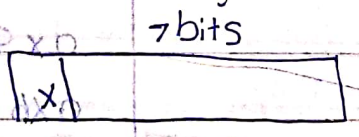
15 internal | 240 external

تقيل

128 | 128

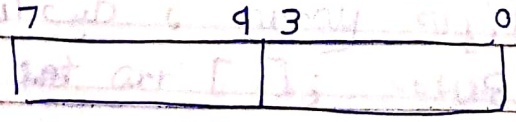
$2^7 = 128$ (7bit to access The group)

يعني محتاج register فيه 8bit هاخد منه 7



in general

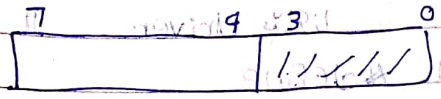
لكن ال STM قالت مش محتاجه كل دا ، أنا محتاجة عدد صغير جداً ال هما 4bit



علشان نعمل Software Priority محتاج 4bits

for STM families : 4 bit have 80 Group

طيب كدا هناخد ال



high ولا low

LSb

هنا هياخد الـ 4 MSb والجزء ال high هيبقى reserved

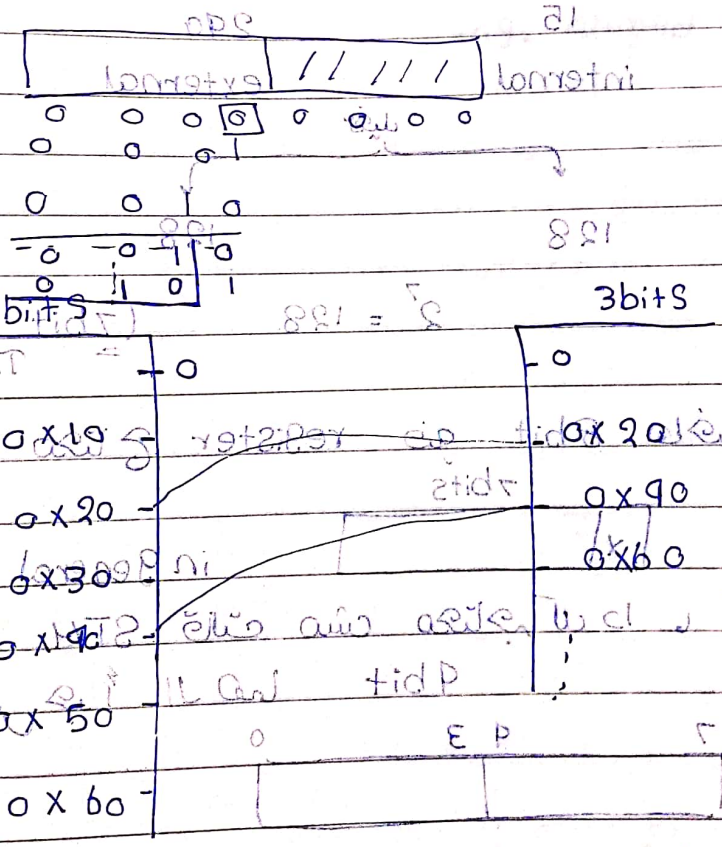
* ليه؟

* 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 → 4bit

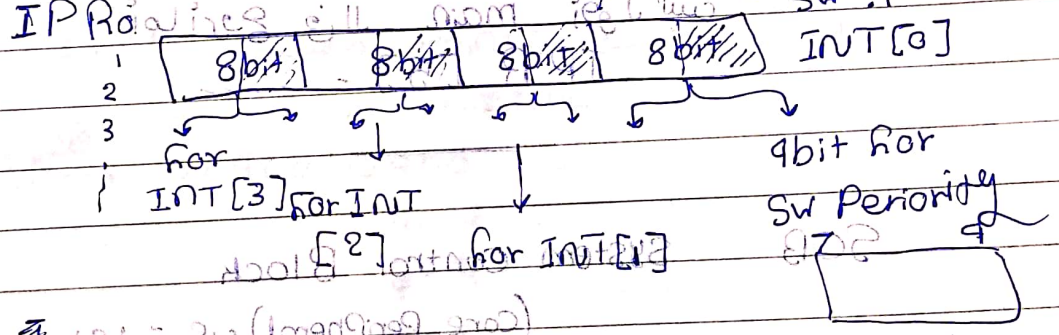
* عندنا في Programming manual

IPRX

* define دا بال register اللي أتعامل معاه

* مكتوب في ال datasheet إن ال R دي byte accessible

ولأنه bits 32 ، sw .P



أول 1

* define NVIC_IPR0 ((volatile u32 *)Baseaddress)

ومش هنحط (derherence) لأن لما أجي أعمل access هعمل access على إنه array ، طيب إيه الفرق ؟

u8 arr[ ];

u8 arr[1] = ...

u8 *Ptr = arr;

للأراي address

يعني أنا ممكن أعرف Pointer بيشاور على ال array دي

طيب عايز access

Ptr[5] = ...

= نكافئ *(Ptr + 5) = ...

5 → offset

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

IPR0 → element 0
IPR1 → 1
IPR2 → 2

index

IPR[ ] =

يعني من خلال ال Index دي هيروح لل IPR 0 أو 1 .. 2
وهكذا.

#define NVIC_IPR ((volatile U32*) Base address)
Pointer لل ((U8*))

أما أروح في ال main أقول بس
IPR [5] = 0xA

SCB System control Block
دا عبارة عن (Core Peripheral)

Register 5 دا R لل Password A

SCB_AIRCR 31	zero	lock	vectkey stat	16
15				0

3bits

Priority دول هما اللي بيحطوا في ال
Sub group و لل Group لل

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 في EXTI_Config.h

when

مش هنعمل 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

وقتيت الPointer يشاور على أول واحد → 0x4001 0400

0x400
0x404
0x408
0x40C

Pins EXTI0 → PA0, PB0, PC0

EXTI1 → PA1, PB1, PC1

طيب مين الهيختار line
هيشتغل على A ولا B ولا C
هو ال 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 Sehedler

1 NVIC

↳ Control external interrupt

- enable
- disable
- Set Pending
- CIR Pending
- Set Priority

Terminology:

1 Development بكتب كود

بعد ما بخلص كود بعمل 2 Building

generat. exe او .hex ال excutable file ال Source file من

على حسب ال Toolchain بتاعك

3 Programming / Burning / flashing / downloading

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

debugger

* أى debugger فى الدنيا بشغل يحرق الكود ، لكن مش
الكلس ، يعنى مش أى burner (flasher)
يقدر يـ debug

ST Link دا بقى يقدر يـ debug احنا كنا نستخدم flasher

* معلومة هامة :
علشان أكتب أى external Interrupt

nameof Peripheral - IRQ Handler
مثلا EXTI

(InterruptRequest)

لازم الشكل دا
ودا الـ INT الهيجى لما الـ EXTI يعمل INT...
و هى دى الـ func الهتروح لـ vector فى ملف فيه
مكان الـ EXTI Table
والتسمية دى تحديداً اجاية من الـ Linker Script
هنعرف وقت خروج الـ bootloader

Not real Time as (delay)

برنامج محاكاه (software) (لتحقيق فكرة مش آلة)

* Simulation ⇒ S.W Program runing on PC
زي برو تس
↳ run on PC Processor
عمرها ما هتكون حقيقية 100% ؟ محدش 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 تحديداً ،
أفضلهم هو ال Emulator لكن غالي جداً

* عايزين نبقى نغير ال Priority

* Each Interrupt has Two Priorities

Group Priority (الا بتحدد ال nesting)

Sub Priority (لو حصلوا مع بعض في نفس الجروب)

هنا ARM قال لكل Interrupt عنده register اسمه Interrupt Priority وهو 8 bit للـ Int external

الـ 8bit دول هيتقسموا في ال Group & Sub Priority

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

$2^4 = 16$ احتمال بس

رغم إن بعض ال MC يكونوا موجودين

Group — 2 bit

Sub — 2 bit

يعني في كل group يكون فيه 4 sub 0, 1, 2, 3

Ex 1; → 00 — 4 groups

01 — Group 0 Sub 1

00 — Group 3 Sub 0

1 bit → 1 — 2 groups

8 sub — 3 bit

100 — Group 1 Sub 4

* في الأول لازم أسأل ايه تقسيم 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 رفضوا وقالوا مش عايزين ال 9bit

ال على اليمين ؟

لأن دا هيزود ال nesting كتير جداً .

طيب دلوقتي أنا ك Software بقدر

أتحكم في تقسيمة ال bits جوا ال

كام للجروب وكام لل sub

ودا عن طريق

* Core Peripheral "SCB"

(System Control Block)

واحدة من وظايفه إنه

* Define 9 bits of Priority classification

يعني تقسيمهم عامل ازاي

في بقى Reg. هو ال 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 حاجة

متشابهة زي 00 11 لكن معناهم مختلف

100 1 عملوا PW ليه

الـ R دا SCB AIRCR 32bit

16bit	low 16bit
Password	Configuration الـ عايزها

مش هتعرف تحط في الـ Config إلا لما تحط الـ Password

SCB_AIRCR = 0X05FA إي عايزة

Password

هو عايز يأكد فعلاً إنت عايز تكتب .

جوا الـ Register دول بقى الـ محتاجهم 3bits : PRIGroup

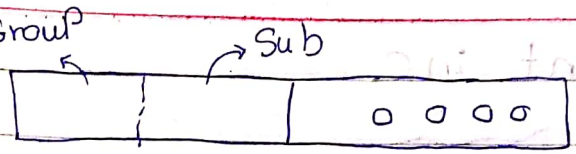
علشان أعرف التقسيط

3bits : (Write only) دي بتعمل Software reset و بتكتب zeros عليهم ، هيجي وضعهم reset عليشان منعملش

* النص الفوق بتاع الـ key

نبص لرقم على 3bits بتوعنا

otherwise



EXTI0 (6)

0100 0000

Group1 Sub0

EXTI1 (7)

0011 0000

Group0 Sub3

Counter / Timer

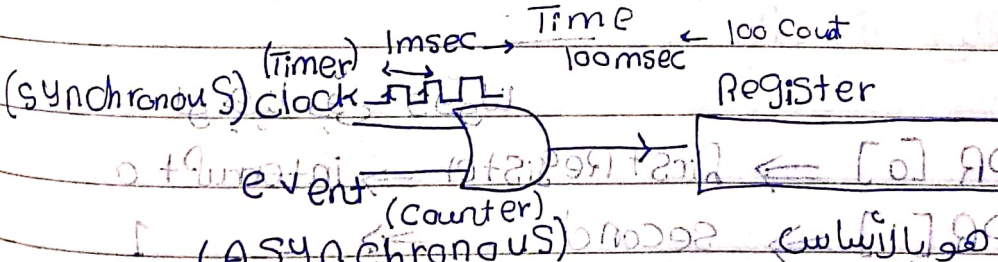
أي Counter في الدنيا عبارة عن register

عرض السجل

Resolution

مع كل edge داخلة على الريجستر ح يبعد عدة و هكذا

ال input signal اذا بتبقى حاجة من الاتنين



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

كل دوسة بعمل raising edge فيعد عدة

لو لفيت 100 فيه بيبقى دوست 100 مرة

لكن هل تقدر تقول خدوقت قد ايه ، لا لأنه Asyn يعني ممكن يكون داس على مراحل مش ثابتة يعني بيعبر (الرقم الي 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 resolution

* over flow Count = 2

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

$$= \underbrace{2}_{\text{resolution}} * \underbrace{\frac{1}{F_{Timer}}}_{\text{frequancy}}$$

• Processor بتاع الـ Timer مش الـ

SysTick

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

  event مش clock بتاعته طبعا input (يعني ال) Switch زي

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

AHB/8 الداخلة عليه

AHB

ISR void SysTick_Handler(void)
{
}

Assume AHB clock = 8 MHz
SysTick = AHB/8 = 1 MHz

$T = \frac{1}{10^6} = 1 usec$

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

Systick_load = 100; // delay 100 usec (Micro)
= 1000; // ... 1000 usec
= 1000000; // ...
and = 16000000
$2^{24} = ...$

ASS : Led Animation

open questions

1. volatile key↔word

ex;

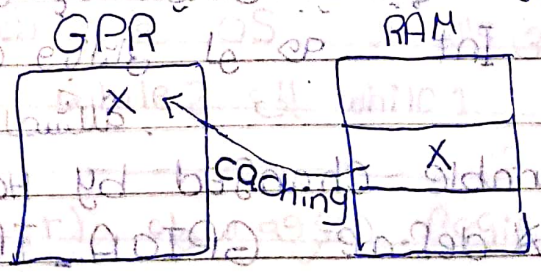
```
main
{
    u8 x = 10;
    while ( x == 10)
    {
        ≡ no change in x
    }
}
```

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 دي Read
ومش هيتغير وكمان هشوار 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
هيخلي يحصل كدا
الشروط دي

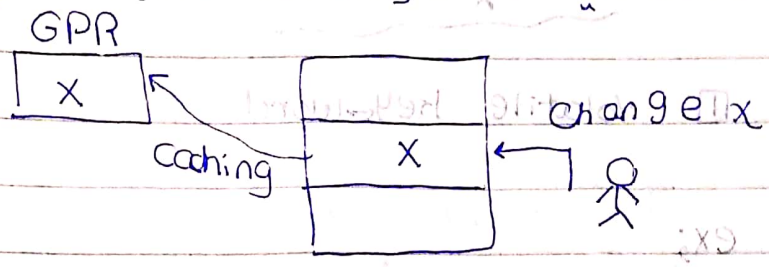


1. No change in The variable.
2. variable is read many Times.

دي الشروط علشان يحصل optamize

volatile

علشان يحصل caching لازم الشرطين دول يتحققوا



فيه عفريت جه يغير ال x الى ال Ram فالـ optamizer
عمال يقارن بالنسخه القديمة الى ال GPR

* فأنا خايف من الفاجعة دى تحصل ...!
والعفريت دا ليه ٣ احتمالات

* Three option for failing

1. Variable changes inside interrupt

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

* معلومة:
linker بيمسح ال functions اللى ال main معملها calling
طيب ISR هيمسحه ؟
لا ، هو بيتخيل إن محدش بينادي عليها وبالتالى x
مش هتتغير ، وبالتالى لو جه int ه ال x هتتغير
وبالتالى هتحصل المشكلة.

2. Variable changed by Hardware

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

↓ read only register

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

ال optimized هل في حد ، بيكتب على IDR فمش هيلاقى

* 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

* طيب لو الـ variable هو readonly مش غيره
ما هي caching ما علشان ملغبطش نفس نحط
Volatile.

③ Multi tasking

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

↓

لكن في الحقيقة الـ operating system
في عالم الـ OS بقدر يخلينا ننتقل من
while 1 إلى while 2 ونحصل 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 */
}
```

اللوديك هيتفذوا :

* لحد الآن مفيش فرق

طيب امتى استخدم دي وامتى دي ؟

function

① can be normally interrupted unless Critical section is used

function call ممكن تعمل Int وده ؟

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 للتاني ،

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

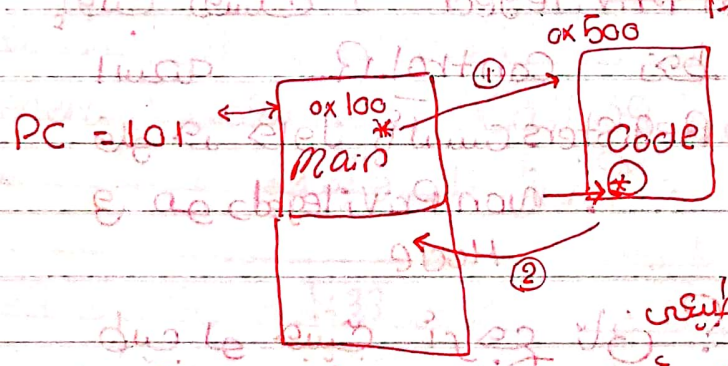
مختلفة كبيرة عن المعلومات ال Saved لما يحصل SWI

بتاع ISR, Context Switching في included

ex: Register PSW (flags)

with function (not) included in function

C.S



PC = 500

~~مبقاش~~ Save بـ 101

علشان أرجعله

في ال func

لكن لو هيـ Save طبيعي

فهيـ Save ، وكمان أ Save ألداتا الـ GPR علشان

لما أرجع استخدمها

ال PSW مش بـ Save في حالة ال function

وهو بيعبر عن آخر سطر موجود هنا ⊛ لأن funct متوقعه

، 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 بيعملها

Save.

بار بيتكتب عليه assembly

ليه لأن ال Control R متعرفش تعمل access
غير لو إنت في ال Privileged موود ، إيه
و ال INT ، رقم بيعمل 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 (Some register) To Security

لو إنت جوا SWI تقدر تأكسز على أي حاجة إنت عايزها
API (Application Public Interface)
ال Interrupt مش بتعمل Calling لل API ، لكن بتعمل generate
instruction بتعمل jump عليه وبتنفذ

SysTick

↳ Control → Clock = AHB or AHB/8
→ Enable to SysTick
→ Enable interrupt
→ flag

بترفع لما ال Int يحصل . وبيـ CIR لما نعمله Read

↳ load → بنحط فيه القيمة الفيه الاعلى عيها (مش بتتغير)
→ 29 bit

↳ value → القيمة الحالية

ex; load 1000 [1000] value

مع ال Pulse ال بعدها يبقى 999 وهكذا
لكن load.R بيبقى فيه القيمة الأولى ومش بتتغير
عن 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 الـ function لحد ما 1000 عدة يخلصوا وأعمل return ، أو لما 1000 عدة يخلصوا يعمل interrupt

2

```
main                          Timer
{                             {
  ===  Timer                    باإما
  ===  Count 1000 عدة          1 Clk
  ===                           lock
                                Processor
                                for
}                             } 1000 Count
```

ونجدي return

، أو

```
Timer
{
  ⟶ return هيعمل
  ⟵ على طول وبكدا مكملش lock لل Processor
  وهي ورجع لل main }
```

تكمل ال كانت بتعمله ، وهيعد ال 1000 عدة بتوعه ولما يخلص
هيعمل Interrupt ⟵

* أيهما الأفضل ، يفضل متعلق في ال function ولا ال main
تكمل وهو يعد مع نفسه ولما يخلص يعمل Interrupt

* **Synchronous function:**

يعني لما يطلب منها
حاجة بتعلق ال Processor لحد الحاجة ال انطلبت تتعمل
ومتعملش return غير لما تخلص . دا اسمها

Asynchronous function (only start)

لما يطلب منها حاجة بتعمل return وتقول روح كمل ال بتعمله
ولما هي تخلص هتعمل return وتديك notification
بعدكدا.

Synch : Asynch

لو عاوز بتظبط بيها وقت معين event يعني عايز أعمل بعد ثانية، (فأنا مستني الوقت دا)، فمضطر استنى

مش بتعلق ال Processor فلو أنا بشتغل ومش عايز أوقف حاجة تانية، يبقى دى أفضل

(2) MSTK_ void Set Busy wait (..) → Sync

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

↓

متعلق في waiting state وشاغلك ال Processor (Synch)

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 يعني أما يخلص 1000 مرة يعمل INT وبعدين يعد تاني 1000 عدة و INT وهكذا

(5) MSTK_ void Stop Timer(...) توقف ال Timer

Asyn

وجا بس في حالة ال

6] MSTK_ ~~void~~ U32 GetElapsedtime(--). → load - value عديت قيمة

7] MSTK_ ~~void~~ U32 GetRemainingtime(.....).
↳ value register.

Pointer To function :

void func I(void);

Pointer: ↓ ↓ ↓

void (*Ptr) (void); → 1] كدا بعرف ال Pointer هيشاور على function ايه

↓

اسم ال Pointer اللي هيشاور على func بتاخد void وبترجع void

2] عايز بقى أخليه يشاور على ال function ال اسمها func دى :

* اسم ال function هو ال address بتاعها

void (*Ptr) void = func ;

لو عملت كدا func() ≡ Ptr()

معلومة: ISR_s بيتعمل في ال SysTick Program .c

مش في ال main

Lec 12

by Eng. A. Atia

→ ① Hardware Circuit

Peripheral Preipheral interrup Enable

PIE

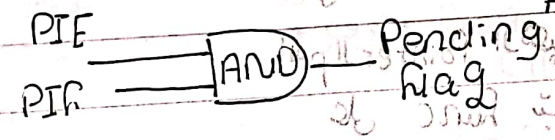
علشان أشغل Peripheral لازم أعمل :

① PIE

② PIF (P.I flag)

لازم يترفع

الاتنين مع بعض داخلين على And Gate



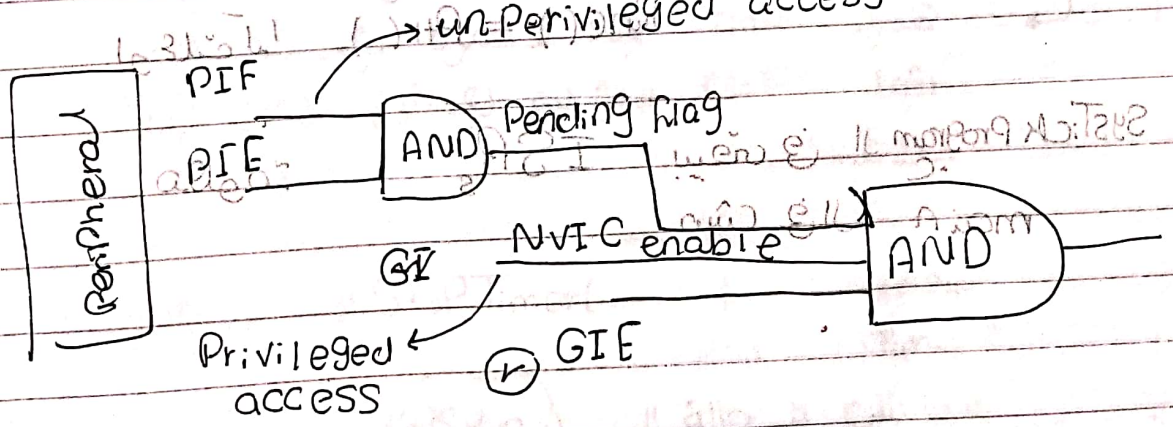
* لحد هنا لسه ال interrup محصلش

لازم شرط كمان أساسي

③ NVIC Peripheral enable لازم دي كمان

④ ولازم معاهم GIE

Global I. Enable



لازم دول يحصلوا علشان ال Int يحصل

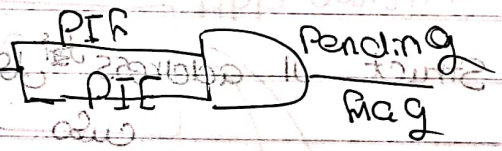
* Global interrupt enabled by default (by H.W)

255 INT

15 internal	240 external
" Control "	" Manage " (دلوقتي)
deliver request to the processor dependent on Priority	① enable / disable ② Priority mangement

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

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



ISR{ → Pending flag Clear
Active flag Set (ISR أول ما تدخل)

} → Active flag Clear (بعد ما ISR أول ما تخلص)

* 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 EXTI — volatile — الأدرس

(↑ EXTI_t *) 0x400.....

اسم الـ struct

EXTI → 0x400 ← 32 →

مش هنعمل dereferance *()

احنا هنعمل aro → أكتر بعمل dereferance

EXTI_t

* وبكدا الـ EXTI بقى بيشاور على الـ address بتاع الـ Struct

كل طيب عايز أكتب على element معين

فهنستخدم بقى → الـ هو اسمه aro

EXTI → PR

هام

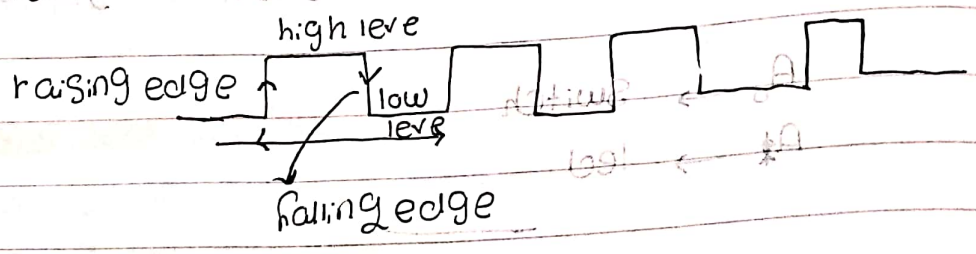
وبكدا في طريقتين بنقدر نأكسس بيهم الـ Struct

1. * (EXTI.PR) = 0 ✓

2. EXTI → PR = 0 ✓

Int أي

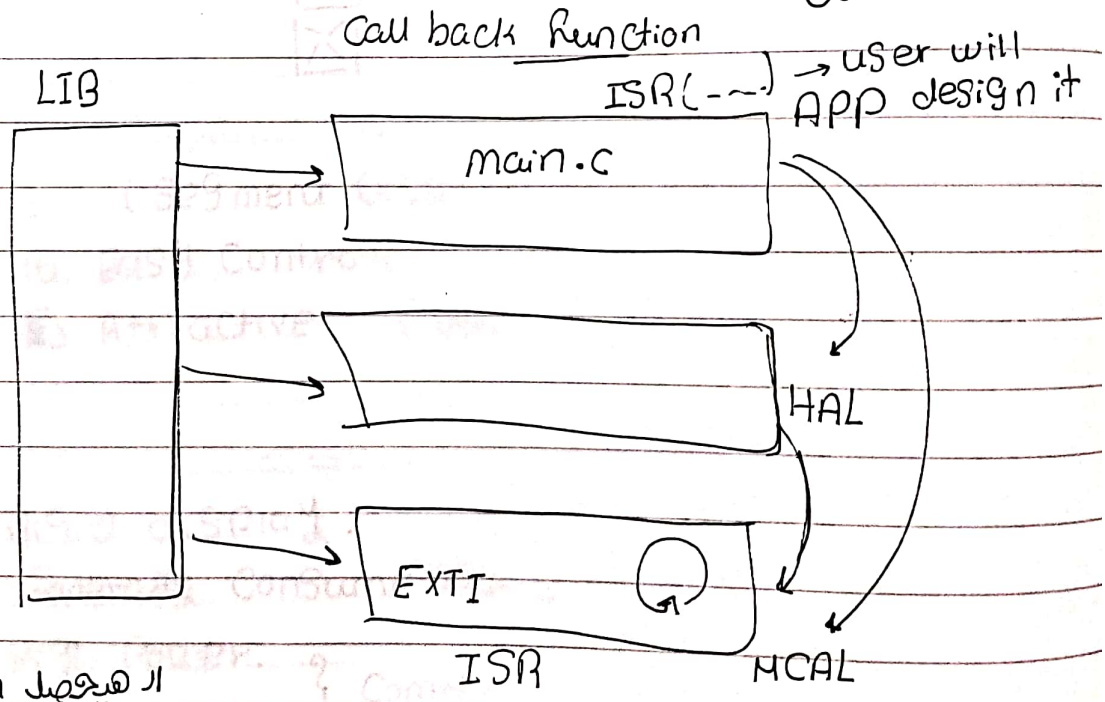
* interrupt → sense mode



Modes:

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

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



based on ISR ال هيحصل

يعني لما يحصل ISR ال بيـ design ال جوا ISR ↑ هو ال user
و في ال APP layer تلك ال ISR في ال EXTI في
ال MCAL و مستحيل ال MCAL تنادي أو تعمل
include لـ APP

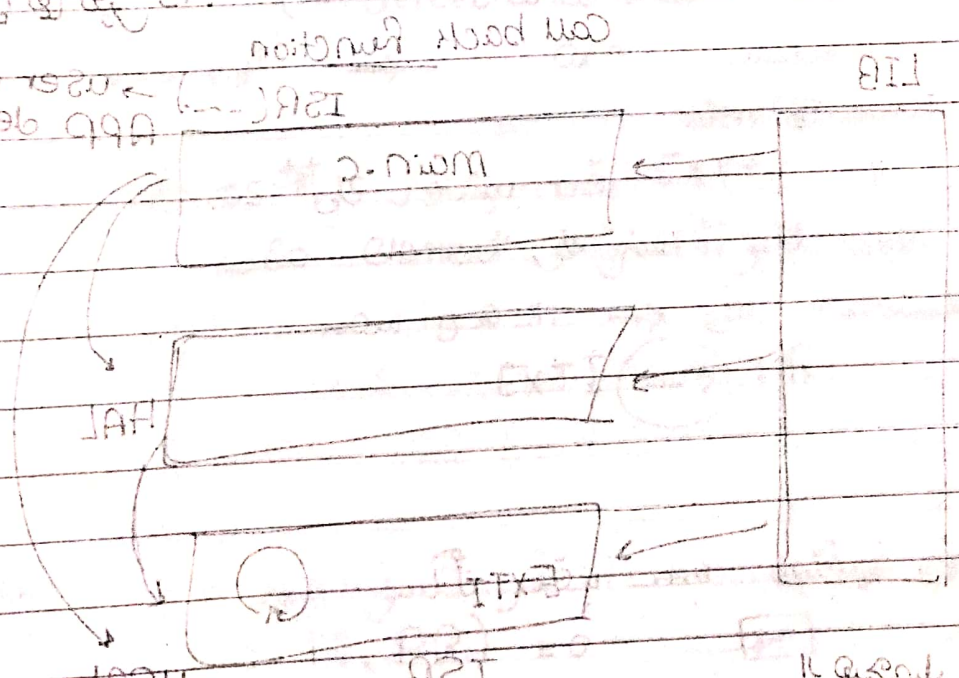
طيب ايه الحل ؟

الحل هو Call back function

A_0 → switch

A_1 → led

end L12 → التوصيل



Lec 13

Eng_Ahmed Assaf

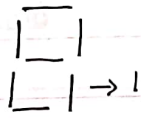
Displays :

أغلب المشاريع بتحتاج حاجة أعرضها

1. Segment Display

كل segment عبارة عن led عريضة شوية مرصوصة 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)

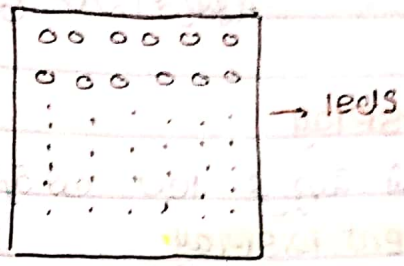
1. Easy Control
2. Attractive (للرؤية علشان ألوانها)

عيوب :

1. Rigid display. (x * y) مقيد الشكل
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

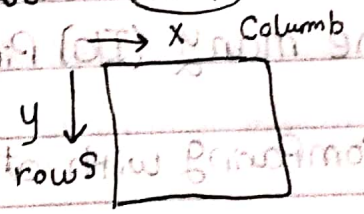
Cons :

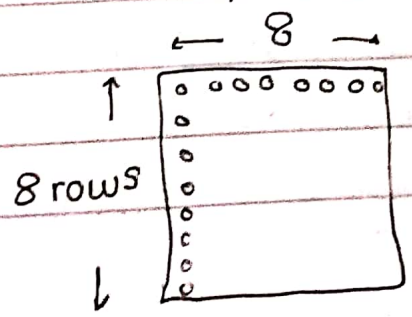
1. Little bit harder Control.
2. High Power
3. Generally Consume big (I/O) number

LED Matrix Display :

1. Specifications: مواصفات

↳ Size (X * Y)

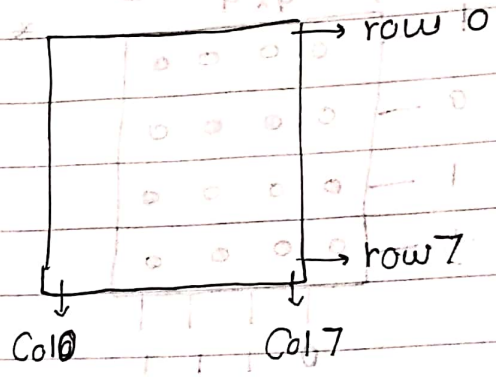




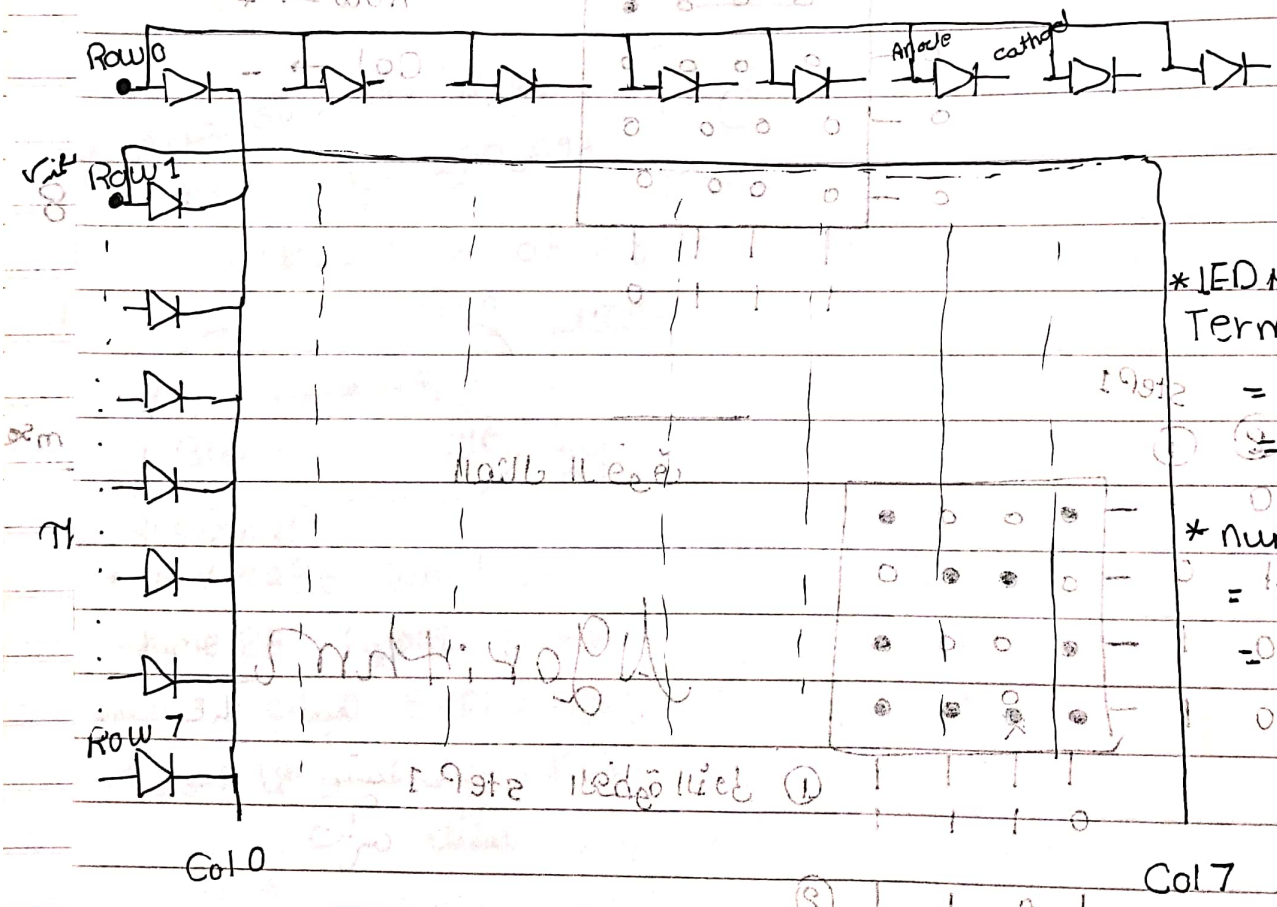
ال مقاس 8*8 led Matrix

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 آنود وسالب

Row → +
Col → −

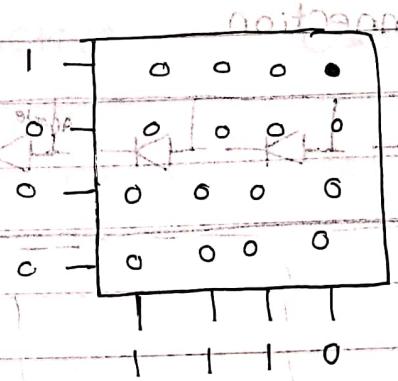
بتاعنا دلوقتي لكن ممكن يقابلك غيرها في السوق

* Concept of operation

"فرضاً" 4x4

* operation Mode
  - ↳ Column Control
  - ↳ Row Control

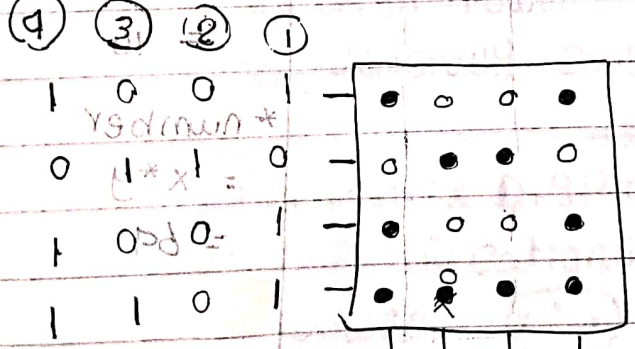
active Column at Time



Row → +

Col → −

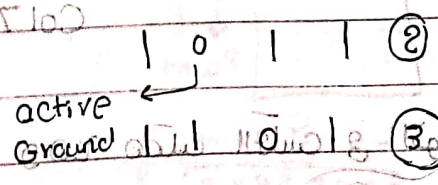
step 1



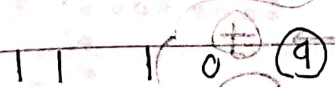
المثال الفوق

Algorithm

① الخطوة الأولى step 1



active Ground



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 باسمه

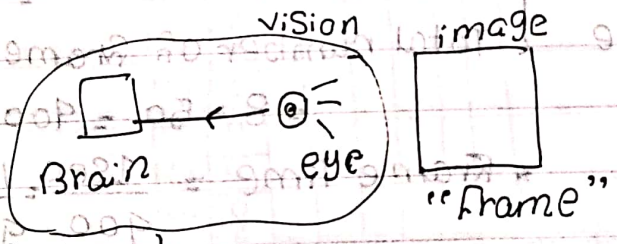
Presistance of vision
POV

كسور الرؤية

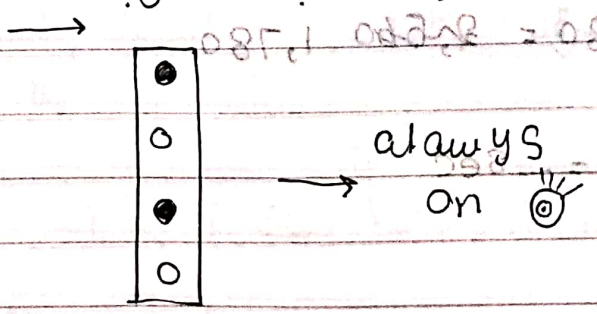
ودى خدعة
* علشان نشوف الصورة مرة واحدة.

POV Concept
← مرعبة جداً والقصة بتاعة الفيلم اللى بتخلى
← وهى معروفة دولياً
تالما

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 كلام بعضه
فلو احنا اشتغلنا على 50 FPS هش هنشوف فيها صور ورا
بعض لا، هنشوفها زى الفيديو كدا بشكل مركب.



on → ~~60~~ 50 frame in second
off → 50 frame in second

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

Algorithm

Enable column 0
Set rows value
delay (2.5 ms)
Enable Collum 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 = $\frac{1 sec}{400} = \frac{1000 mse}{400}$
= 2.5 ms

ex 16 × 16 led Matrix
80 Hz ⟹ delay ...?

16
~~32~~ * 80 = ~~2,560~~ 1,780

delay ∴ $\frac{1}{1,780}$ = ... sec

CRT TV Hz 110

normal range → 50 : 60 Hz POV

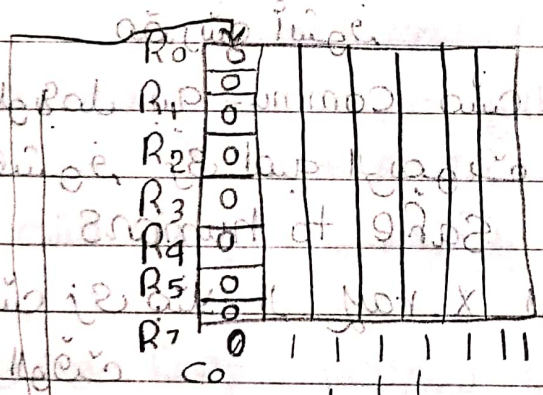
lower → هيسبب مشكلة → Flickering → هزة بتروش

higher → Ghosting → شبح

هتلاقي المفروض ليات المفروض متنورش هتنور خفيف أوى

*** _____ ***

Step1



*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 Col0 (function DisableAllCol)

أفضل من كتابة الـ8 كل مرة meter

Infra Red Communication
IR

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

IR
↳ light can not be detected by human eye

decrease ← | | → increase
visible light
IR

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

مميزات مقدرش أشوفه

1. فلو عمل بيه commu... مش هشوفه زي لمبة الريموت، فدي ميزة مش هشوفه.
2. safe to humans (IR) مش زي مثلاً 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 قوية جداً
4. Relatively long distance
بتتكلم بالأمتار up to 10 meter.
ودا كله
5. very low cost
وهو أرخص Communication Protocal

لهبه البيت بتطلع IR . فهى safe .

أكبر Challenge هتقابلنا

IR Challenge :

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

↳ Interference with The ambient light

ازاى أخلى ال IR ال طالع من الريموت مثلاً يبقى
unige و ميتأثرش بال IR ال حواليه ؟

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

light freq

أى ضوء

و هخلى ال receiver يستقبل ال freq دا بس
أشهر ال freqs دى 38 KHz
freq

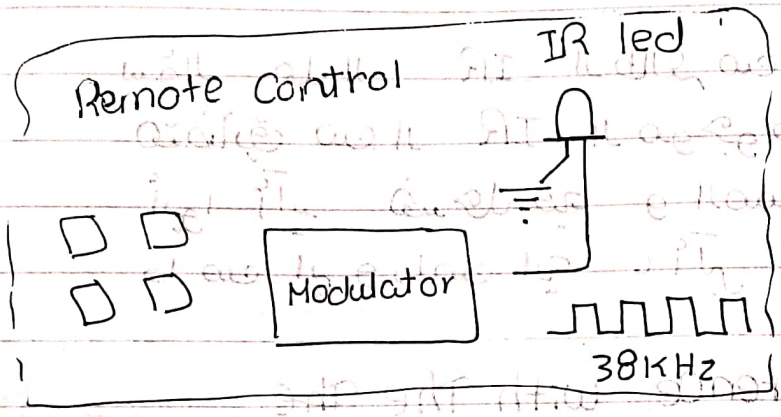
يعنى هعمل IR بتتحرك ب 38KHz و هعمل

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 :

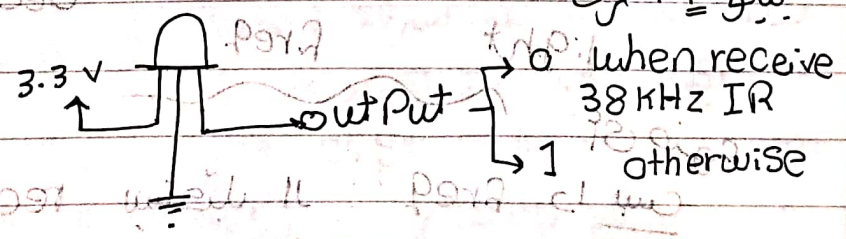
بيتكون من led عادية بس بيطلع IR طرف على Gnd وتاني على modulator ودا ميكروكنترولر دوره يطلع Signal على ال IR يخليها تنور بـ freq مش 1 أو 0 لأ بـ freq 38kH وفي بقى شوية زرار بتطلع بـ freq مختلف هنعرف ازاى دلوقتى



ال System الفات دا اسمه Transmiter وهو ال remote control ؟

لما بنضغط على زرار معين بيطلع Signals بـ 38 KHz بس هنشوف الفرق دلوقتى، الناحية التانية فيه IR reciver

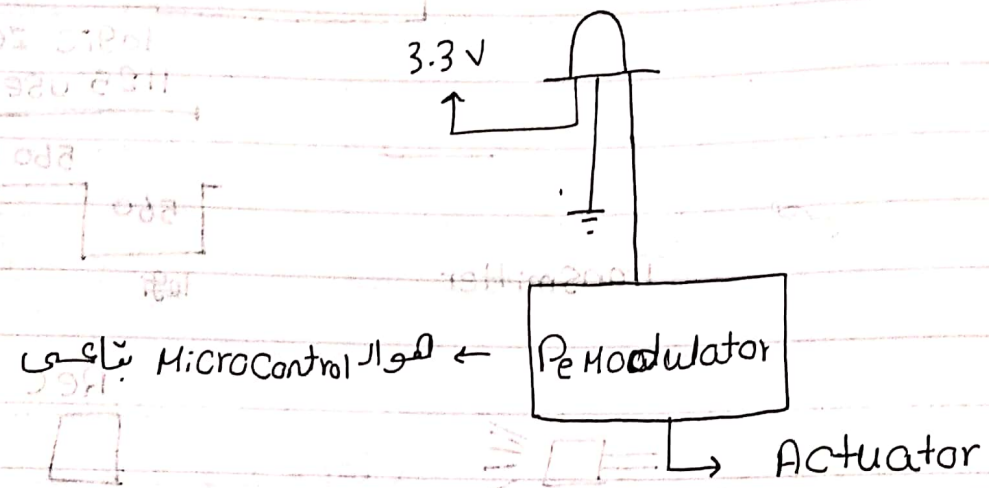
Photo diode اسمه بيكون ٣ أطراف أكتر زاى



يعنى لو مش معمول قدامه حاجة هيبقى (one) ومتوصل عليه تقريباً ، لكن لو جت مثلاً IR هيبقى بـ zero one بردو لحد ما يبقى IR دا بيتحرك بـ 38 KHz ،

بيطلع zero لما يبقى حاسس بـ IR بـ 38KHz

نبدأ بقى ناخد ال output دا ونوصله على Modulatore



هو ال MicroControl بتاعى ←

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

هنا بقى النص الأول ready ال هو remote

من أشهر ال Protocols هو من شركة NEC

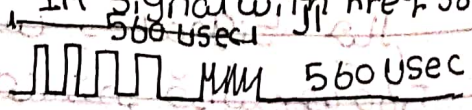
NEC Protocol

↳ Japanes company

هو Protocal عملاه شركة يابانية عملت IR

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

IR Protocal

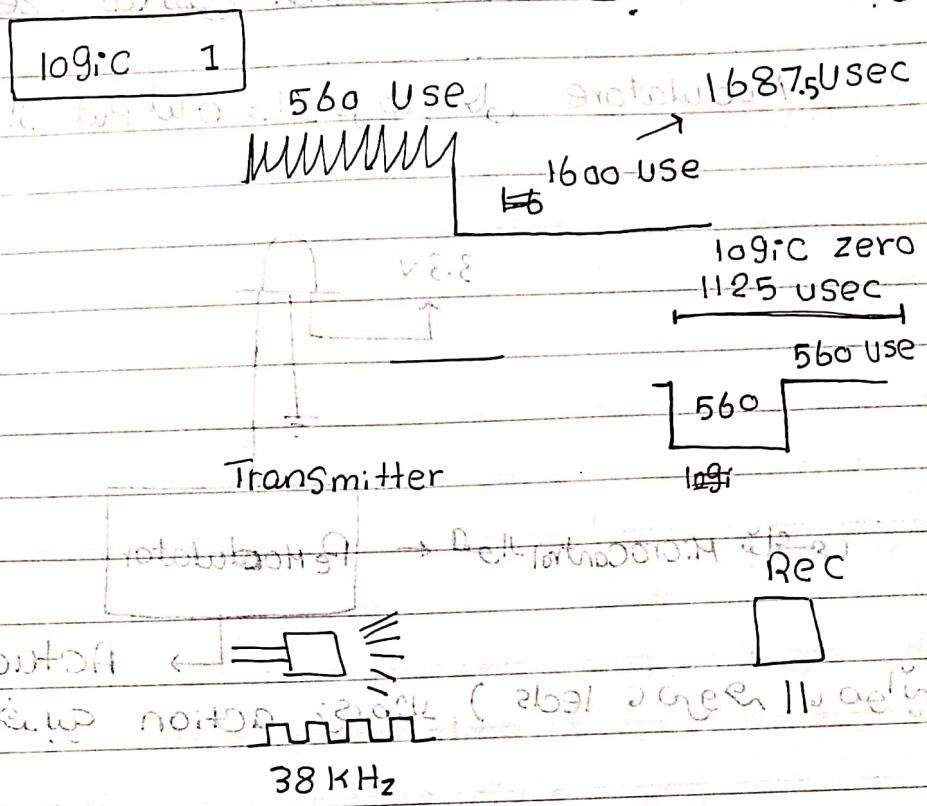


اسمها Pulse train هبعت Pulse train يعنى IR بـ freq

ال receiver يقدر يستخدمه ، طولها 560 use وأقف

نفس الفترة

طيب علشان أبعت



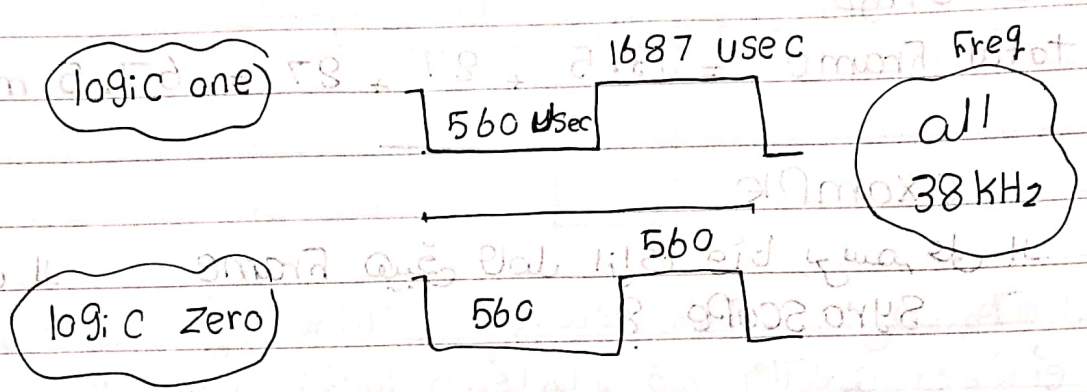
هنا ال Trans هيطلع IR بـ freq 38KHz ولسه ال Rec مستقبلها ش ، طيب ازاى هيفرق أنا باعت zero ولا one ؟
عن طريق الطول ، شوف ال IR طولها كام ، هتبقى مسلط الضوء فترة قد ايه ، لو عايز تبعت zero هتبعت الشكل الى فى الصفحة الورا 560 usec هتنور وتطفى لمدة 560 usec وللت بـ 38 KHz (سرعة) يعنى هشغل ال Transmitter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

0 → 1125 usec

196 1 → 2250 usec

طيب لو عايز أبعت high بقى !

ابعت بردو نفس ال Pulse Train لمدة 560 usec يعنى
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 و Start bit
يبقى كدا كلها 33 bit ، بيعمل 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 ؟

لبتاع ال device ال هكلمه
ولو بيختلف من جهاز ولجهاز ، الاتنين هيستقبلوا
respond لكن واحد هيرد وواحد لا علشان الأدرس

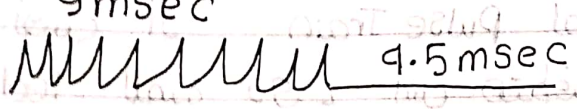
why to send inverted bits ?

↳ 1 Verification للتأكيد ومفيش error حصل فى النص

↳ 2 الأهم 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



طول الـ frame

total frame = 13.5 + 27 + 27 = 67.5 msec

example

شكل الـ frame هيبقى كامل ازاى وأنا برسم على الـ Syroscope ?

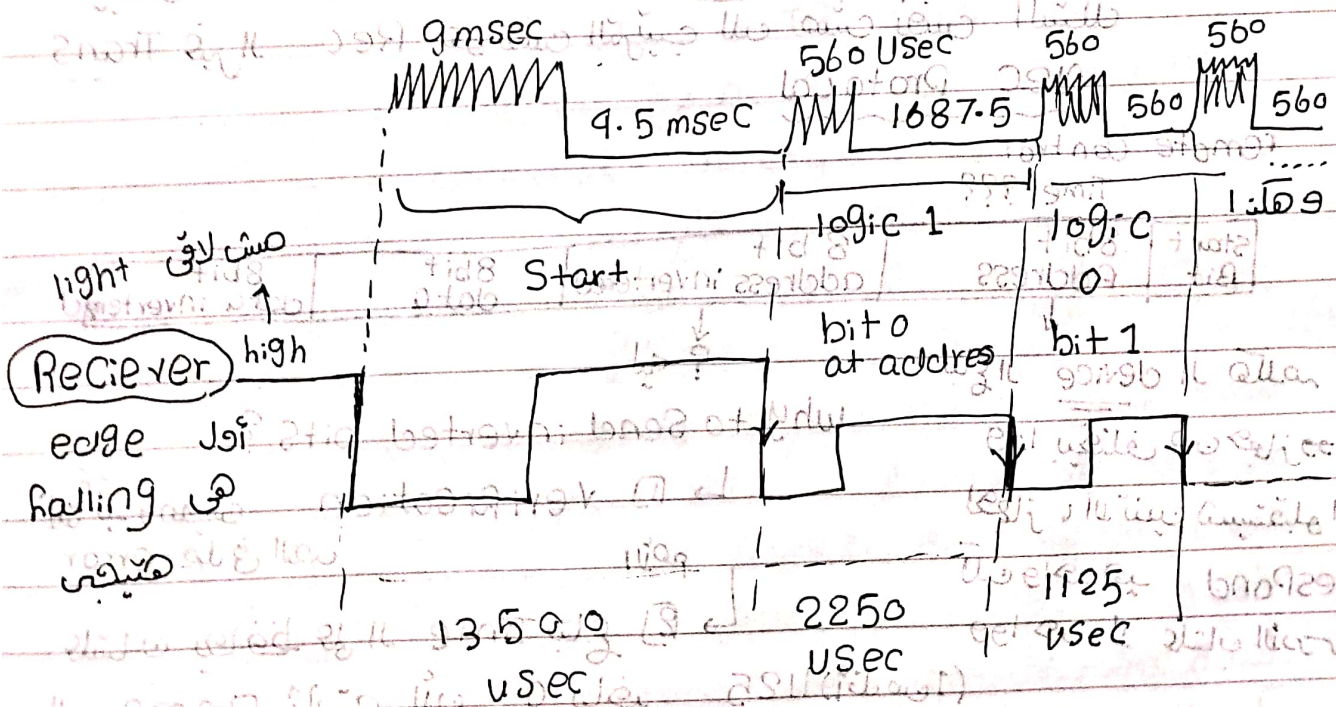
جهاز address = 1 decimal

هبعت data = 10 decimal

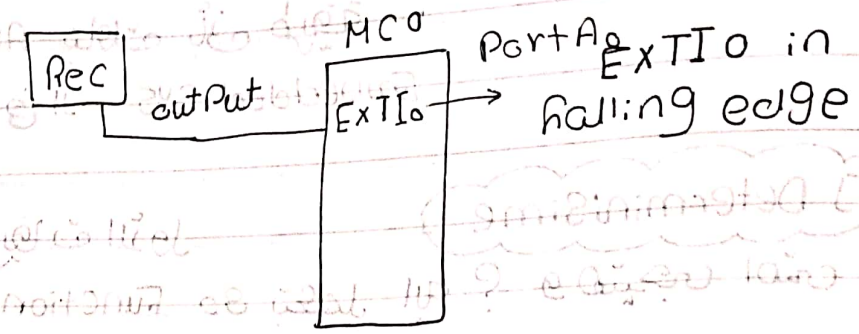
هنرسم كدا من ناحية الـ Transmiter هيبقى كامل ازاى

address هى الأول LSB →

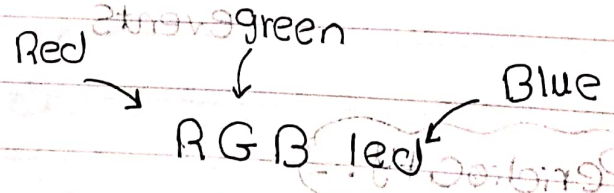
Transmitter 0 0 0 0 0 0 0 1



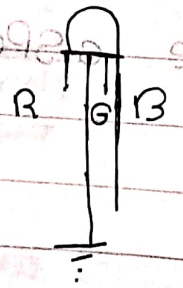
Simple Code :



هنا هكون فجر ال EXTI إن يدينى Int مع كل falling edge
وأشوف هيعدتام عدة ، لو لقيت الوقت العدى فى
ريجيستر 13 500us يبقى دى Start ، ولو 2250
يبقى دى bit قيمتها one ، ولو فى ريجيستر
1125 يبقى bit قيمتها zero



جواها
3 ليدات
ممكن أحط واحد على أى لون
فيهم علشان يتور أو الاثنين
مع بعض



ASS + Homework + Technical Support

Software Architecture :-

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

* شو شو 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

⇐ استخدمناه كتير

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

مميزات

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

يعني تقدر تاخد الكود ده وتحطه على أي Micro Controller

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

loop

* 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

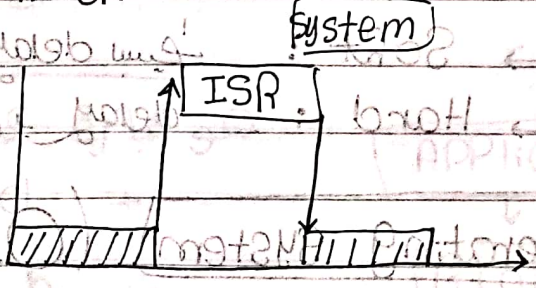
2 ↳ Foreground | background

"Super loop + interrupt"

(ممكن يكون أكتر من واحد) foreground 1

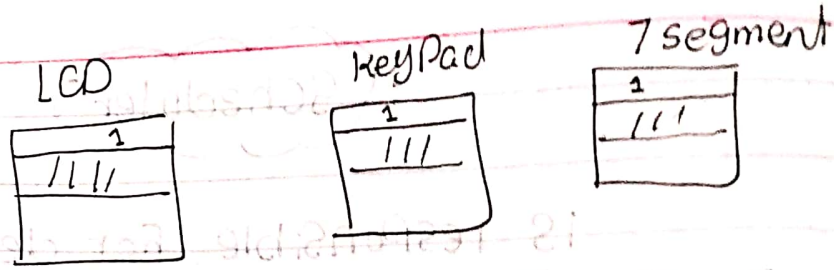
ومن هنا أقدر أستجيب لـ event خارجية أثناء ما أنا شغال

وهنا الـ (INT)



main: شغال فيها الـ

"event trigger": يعني بيستجيب لـ events



يعني حتة من كل حاجة ، وهنا بقى ال OS هو ال بيعمل ال Multitasking

---

طيب ال RTOS موجود فين ؟

هو مرحلة موجودة بين ال Software وال Application

طيب هو بالنسبة لل Two Architure التانية ؟

* 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

دول أهم مكونات 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 هي ال Scheduler :

is responsible for determing
The order of execution Tasks

يعني هو أساس ال RTOS هو المسئول عن ترتيب
ال Tasks ومين ال Tasks ال عليها الدور تروحله
هو دا عصب ال RTOS

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

* dispatcher:
يشوف ايه ال ~~discussion~~ decision ال أخده ال Algothim
ويروح ينفذها

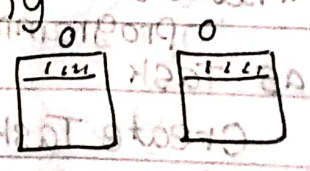
mean: executs the decision Taken by the Algorithm

زي مثلا "Context Switching"

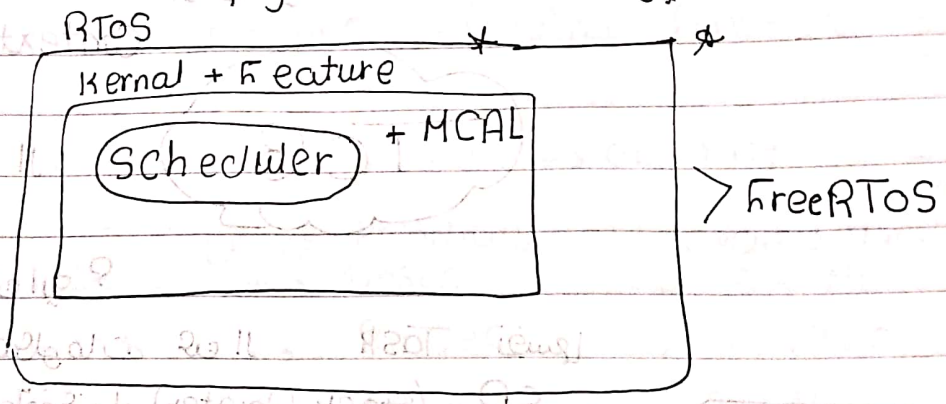
Algorithm: : which "Tasks" should be run
أني تاسك ال هتشتغل ، أني تاسك ال عليها الدور في التنفيذ.

كل Algo ألجوريزم تاني
1) Shortest Task first come. (أقل execution Time)
2) Priority based
3) Round robin "Time slicing"

نفس ال Priority
هنا Timing من دي وينفذها وياخد من التانية
وينفذها وهكذا



في Algorithms كتير، كلنا دول أشهرهم.



⇒ what is The difference between function and Task ?

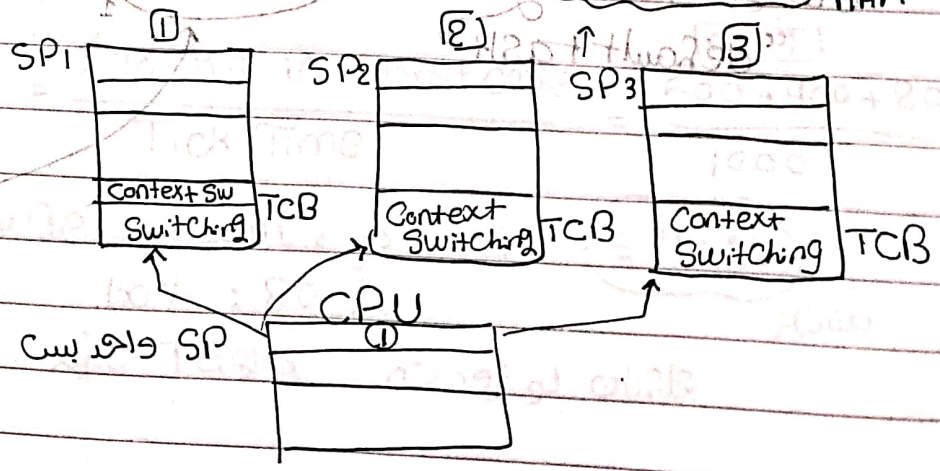
Task ⇒ C function + Timing + Context Switching + Storage

↳ TCB (Task Control Block)

كل Task ذاكرة نفسها خاصة ال CPU لوحدها ودامش حقيقي والسبب في الـ Structure بتاع ال Task وعلشان نحفظ ال Structure دا في حاجة هتنظمنا ال Structure دي وايه ؟

كل Task ليها ال TCB بتاعها

Stack Pointer of RAM



كل Task بيبقى ليه TCB بيتحفظ فيه ال Context Switching

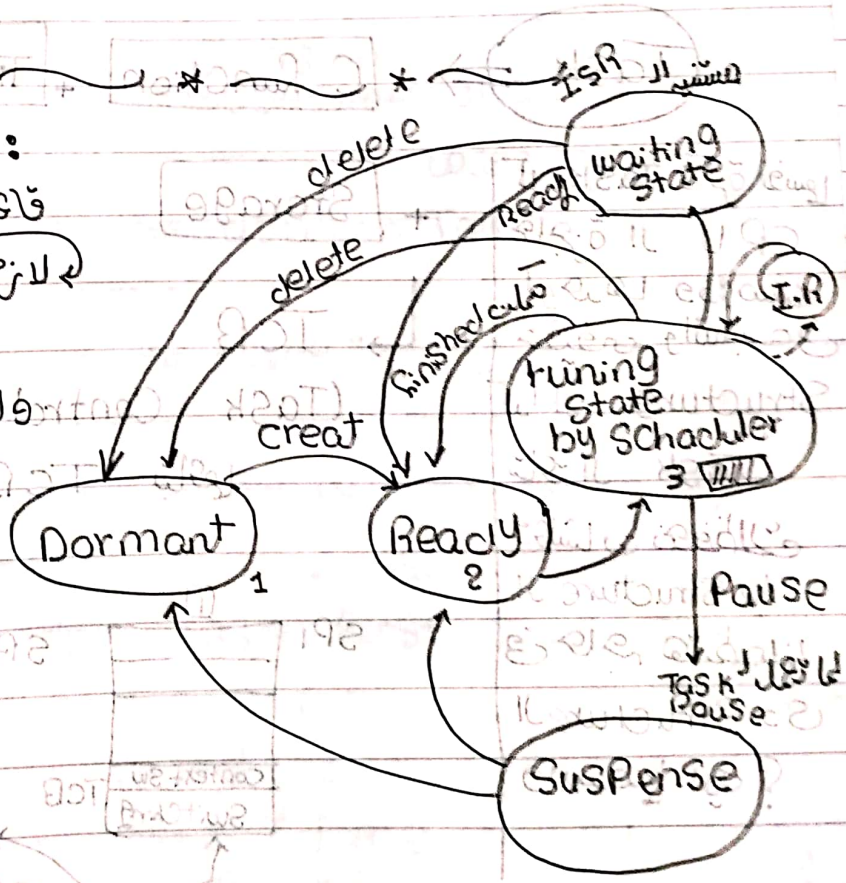
طيب ال TCB عبارة عن ايه ؟

فيه معلومات عن ال Task نفسها

TCB
- SP (Stack Pointer) بتبدأ منين أو واقفة 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 الفاضي علشان نشغل ال running ال هي "default task"



156

Schedulr :→

نعبّره function بتنادي كل Tick Time = 1 msec or 0.5 msec

Schedulr : → Tick → ordering (-priodicity)

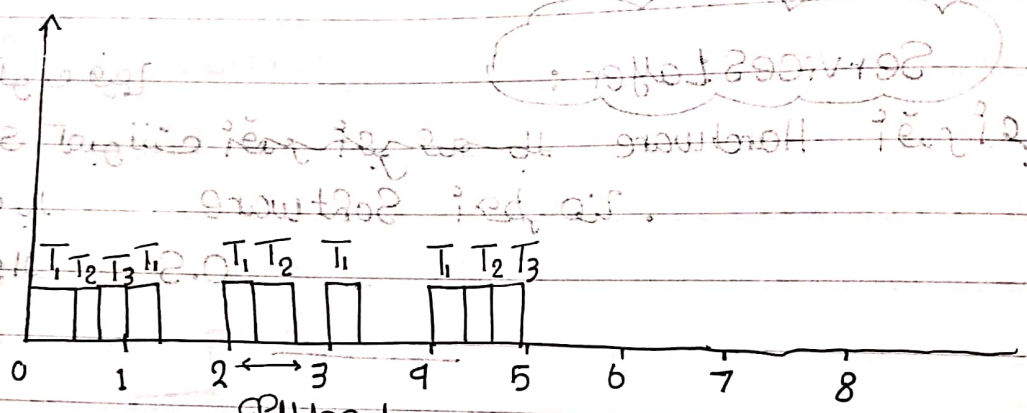
ex;

Task 1	أنادي عليها كل →	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؟

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

$\boxed{0 : 1}$

مشكلة

CPU L لل Range طب أنسب

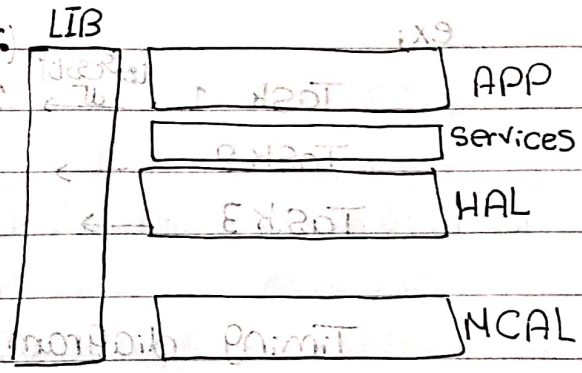
60% : 80%

علشان لو زودت تانيه Tasks

First delay : note مشروحة كويس في ال

*~ *~ *~

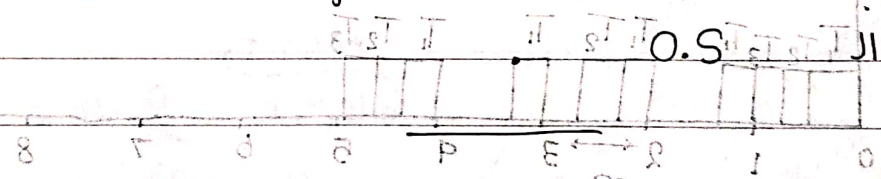
→ layerd Architecture :



Services Layer : بيكون فيها

① أي كمبونت أقدر أعبر عنه بال Hardware أقدر أعبر عنه بال Software أحطه هنا .

② وال O.S



W37

Lec 16

Audio Signal :-

Basics :

Real world — في عالم الـ embedded — Embedded Systems

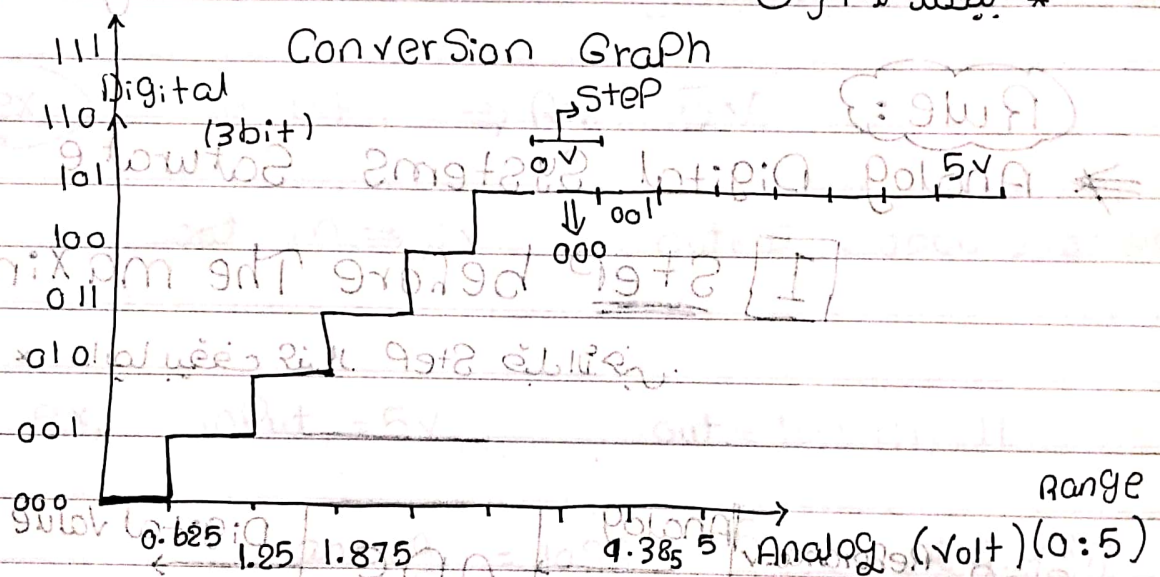
Analog → ADC (Analog To Digital converter) → Digital

Analog عايز يدخل عالم الـ digital عالم الـ

Resolution يعني هتشيل الداتا في bit كام (n)

* بيعمل كدا ازاي

Conversion Graph



هو هيشوف قيمة من الـ Analog ويحطها على القيم الـ digital

الـ عندك

Step :

معناها القيمة الـ Analog الـ تحركني one step في الـ digital

* كدا هنمسك من 0 : 5 ونجزأها 8 احتمالات لأن دا الـ عندي

قيمة الـ

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

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

تحذير:

بعض الناس بيعملو ١ - ٢ 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)

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

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

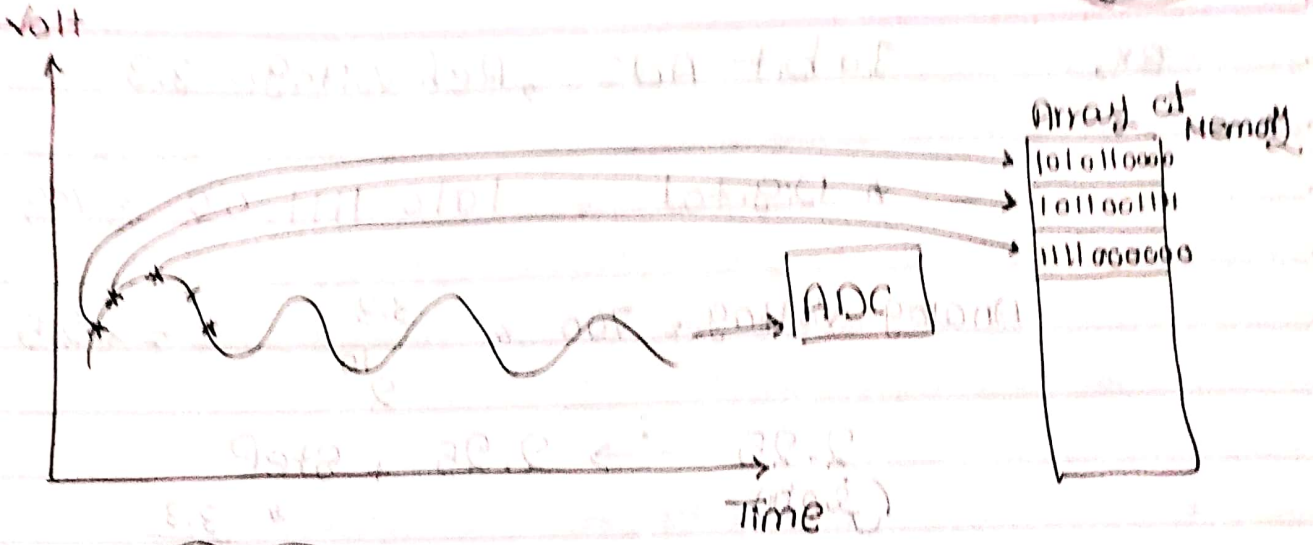
$$\text{step} = \frac{3.3}{1024}$$

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 عند كل sample وامسك كل فيهم وأوديها لل ADC

② Quentization التحويل

يعني أشوفها في أي range

كما في error كل 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 *
20Hz to 20kHz 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

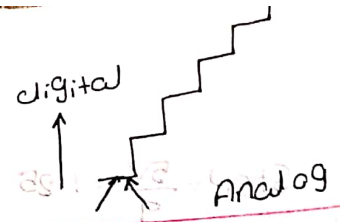
* Samping freq ≥ 8 kHz
for human
voice

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

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

Micr
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 ....?

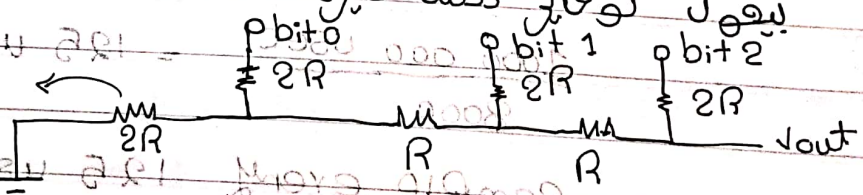
Never

معلومة

ال ADC نفسه بيحتوي على DAC وهو أعقد من ال DAC

R2R DAC

هات مقاومة DAC بيقول لو عايز تعمل دايرة



2R دايماً

كل ما تعوز تزود bits ، تزود R نايمه ← 3 bits DAC وكل ما تعوز تزود

و 2R واقفة

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

2 Bit DAC

علشان نثبت الكلام دا ، على هنثبت على

inPut	outPut
00	0
01	1.25
10	2.5
11	3.75

Homework:

↳ using 8 bit R2R DAC
write C code to generate
Sin wave

rate = 8000 sample / sec

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

* sample every 125 usec

171

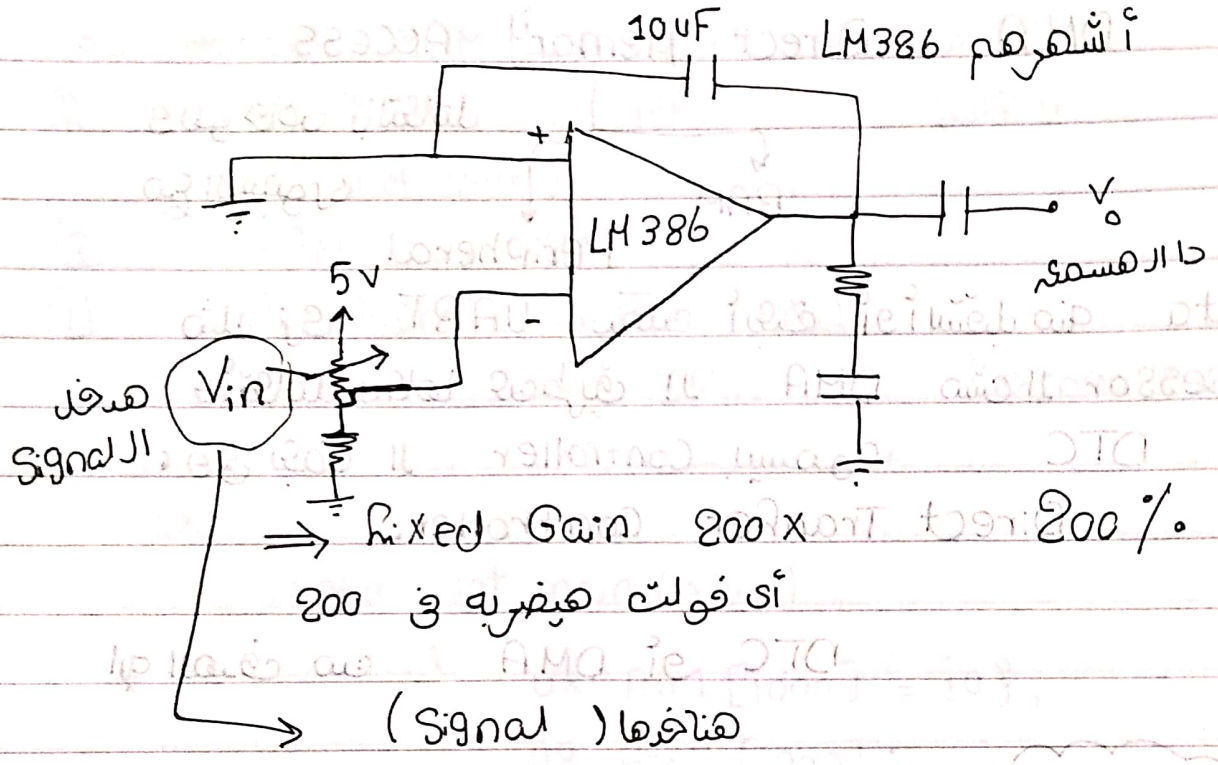
$Max = 3.3 - Step$

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

Amplifier

Audio Amplifier :

أشهرهم LM386



⇒ Fixed Gain 200 x 200 %

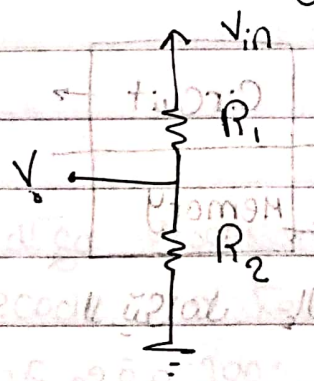
أى فولت هيضربه فى 200

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

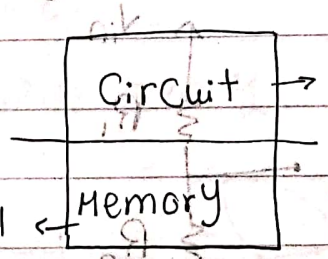
مثلا زي UART بتكلت أبعت أو أستقبل منه data وأتكامل معاها عن طريق ال DMA مش ال Processor، وفي بعض ال Controller بيسموه DTC Direct Tranfer Controller.

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 جزئين.



Try step Buffer

ال registers

وال Processor لا يتكامل إلا مع Memory

For ex;

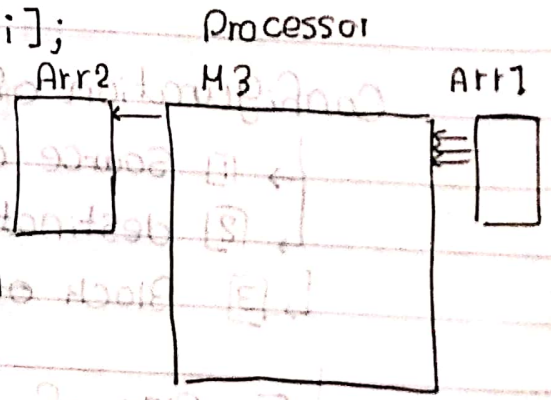
DMA :

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

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

ex;

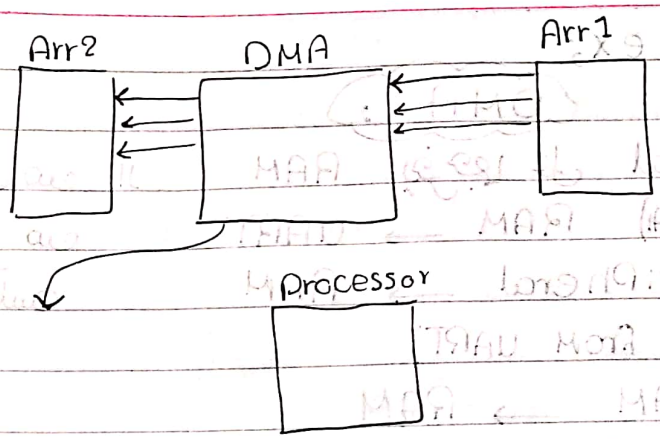
```
~~void~~ int main (void)
{
    u8 Arr1 [1000] = {0};
    u8 Arr2 [1000];

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



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

$1000 * 3 \simeq 3000$ clock cycle

ال DMA



ايه ال هيحصل ؟

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

* ال DMA هتديك interrupt

* After finished will generate interrupt

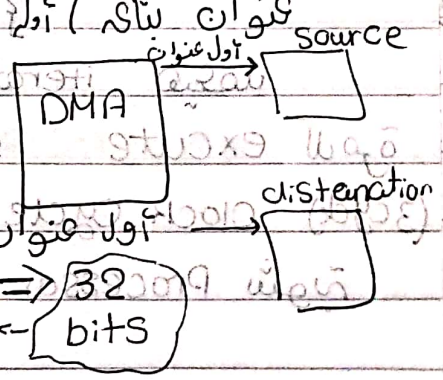
وبكدا هو حل محل ال Processor .

DMA

Core Peripheral مش ST تبع

Configuration of DMA :

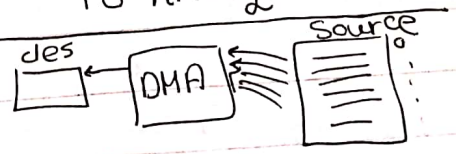
1. Source address (المكان الاول أنقل منه أديله ال عنوان بتاعه) أول
2. destination address
3. Block of length byte كام
4. Size of element :-
   - dependent on data bus ⇒ 32 bits ← بتاعه STM32



ممكن ننقل (1byte , 2B , 4B)

آخرها

1)

STNC	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 بال DMA بدل الرسم الفوق دى

* DMA :

* Can be Configured To Trigger Source anew Data based on Peripheral event

→ Free running (don't wait any Thing)

6 DMA :

can repeat → "(Circuler)"

يعني ينقل الداتا أكثر من مرة

ايه ال interrupt ال ممكن تولدها ال DMA

DMA Interrupts :

1) Transfere Complete :

لما يخلص نقل داتا يدي interrupt*

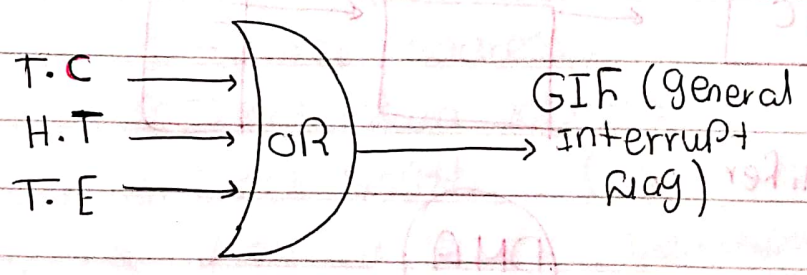
2) half Transfere complete

لما يخلص نقل نص الداتا

3) Transfere error (Memory Consumed)

أشهر ال error

لو حصل error أثناء نقل الداتا



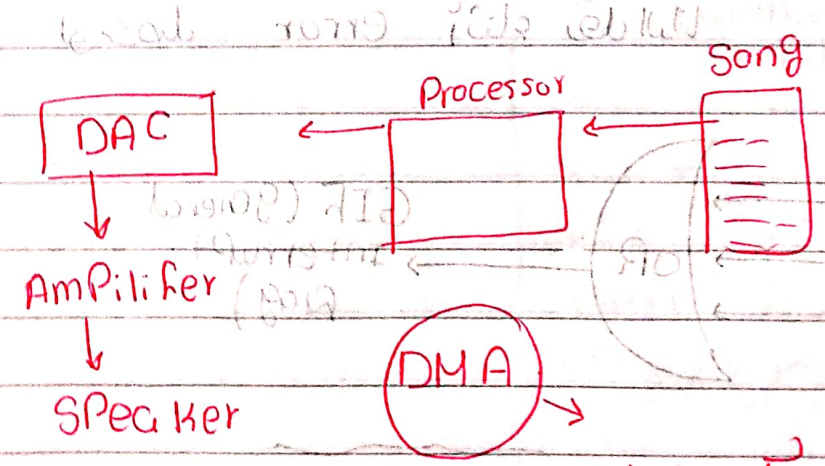
DMA → AHB

→ The Same clock for Processor and DMA ?
which is faster Than?
DMA is faster Than Processor?
(Fetch, d, excute) Processor لأن مش بيعملوا ال
فهى أسرع.

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

So; DMA is Considered another Processor.

أهمية ال DMA:



ممكن دايبدل ال Processor هنا
وممكن تتحكم فى سرعة ال DMA

3) DMA Channels :- 7 Channel

1	2	3	4
UART	SPI	I2C	CAN

ليشوف ال Channel اللي 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

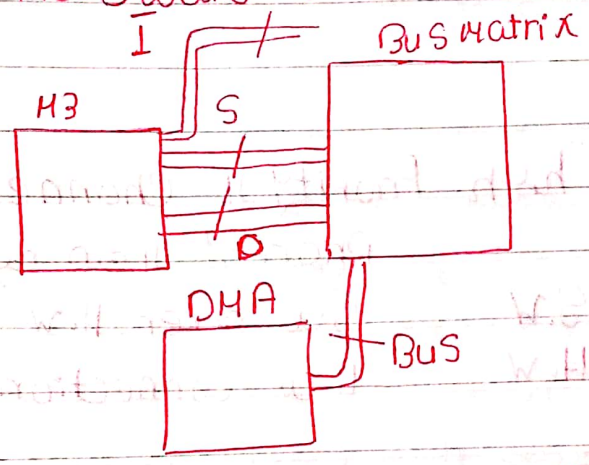
DMA Configuration at file (Config)

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.

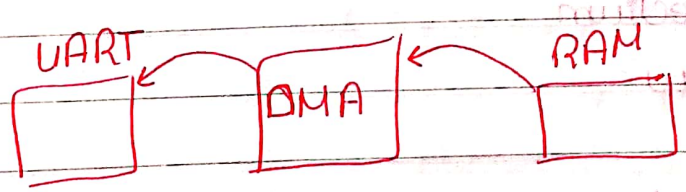
DMA Features :-

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 register لما [3] Flags ليها channel كل

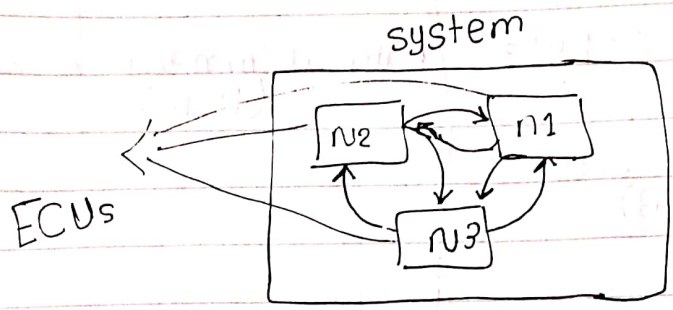
GIF, GTIF, GCIF, ~~GIF~~

وليها Flag بيعمل (ناتج) OR بينهم

الـ Flags دي Read only

Communication :

يعني عندنا system واحد قررنا نقسمه على كذا 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 ؟

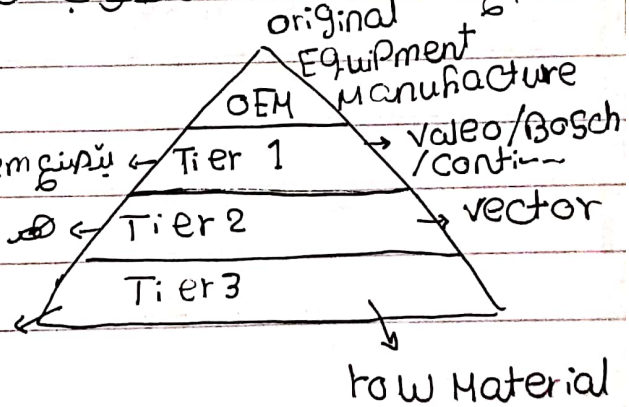
1) Complexity ↓

هيبقى سهل في الـ debugging وبدل هنتعل الـ Complex

2) Reliability ↑

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

3) Dependancy ↓



Tier 1 ← بتصنع system بالكامل زي الكشافات مثلاً والموايش

Tier 2 ← هم بيصنع حته من الـ system بيديها لـ T.1 يكمل بيها الـ sys

زي الـ Compilers والـ debugger

Tier 3 ← بيصنعوا مثلا المايكروكنترولر، زي البلاستيك أو الفتيل أو الـ PCB

Conventional network

كل node تكلم التانية

فوصلو فالأفضل هي طريقة ال

* Communication Protocol

وهي طريقة توصيل معينة

لأي Protocol

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

Standard VS Protocol

Standard: يعني Protocol كل الناس بتفهمه بنفس الطريقة في العالم

مثال
- I2C
- CAN
- LIN

Protocol: يعني طريقة معينة ومحدش يعرفه غير اللي الروم دي

مثال
- UART (تخيل أسهل بروتوكول ...)

* 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 ~~Hak~~ Hack)	↑ ①	↓

⇒ Security (بالنسبة للداتا)

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

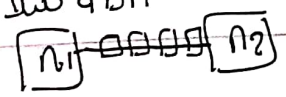
⇒ Safety (يعني أمان بالنسبة للإنسان)

2 Data Transfer mode

يعني الداتا هتتبعت ازاي.

Serial Communication — Parallel

مثلا 4 bit



at The Same Time

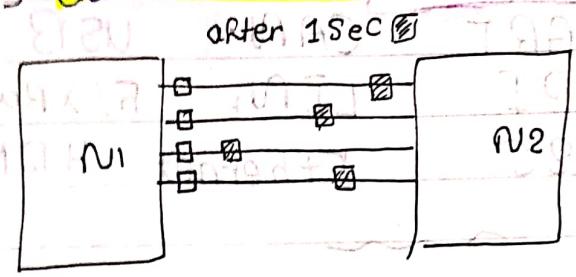
ex; 1 bit ⇒ 1sec

Serial
* 9sec

Parallel
* 1 sec

* **Parallel com. Problems**

① **data skew**

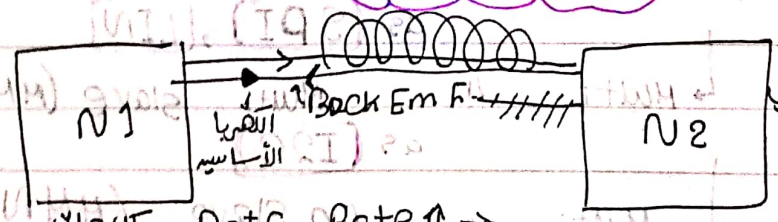


skew

حصلت المشكلة دى لأن الأسلاك دى مقلش مش فى نفس ال connectivity بسبب عوامل طبيعيه مثلاً ضوشه هينفع بعد 1sec أقرأ لأن الداتا لسه مش هتكون وصلت فهتاج أستنى وقت أبطء واحدة وده مش كويس (مش rate).

* طيب هنحط data Rate Limit le

② Problem ⇒ **Interference**



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

كلما تزيد ← Data Rate ↑ ⇒
Interference ⇒ هتزيد

[3] Complex ↑

[4] higher Cost ↑

يعني في عالم ال embedded system ال Parallel اختبار فاشل

⇒ **Serial** ال

- ↳ UART
- ↳ SPI
- ↳ I2C

CAN USB
LIN F/X Ray
Ethernet HDMI

[3] N2N Relation (Node & Node) (Se Pi Ch Ac Ti on)

علاقة ال Node بال Node تانية

[1] Peer To Peer أي حد يتكلم في أي وقت عندهم نفس القوة في ال Power

[2] Master Slave

يعني ال Master هو ال بيتكلم وال Slave لا إلا لما Master معين يقوله يتكلم

↳ Single Master Single Slave (SMSS)

↳ Single " Multi " (SMMS) as (SPI), (LIN)

واحد يتكلم والتاني بعدين

↳ Multi " Multi slave (MMMS) as (I2C)

يعني شرط ال base يكون فاضي

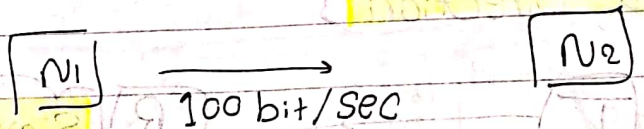
↳ Multi " No Slave (MMNS) as (CAN)

هامة جداً

9 Synchronization

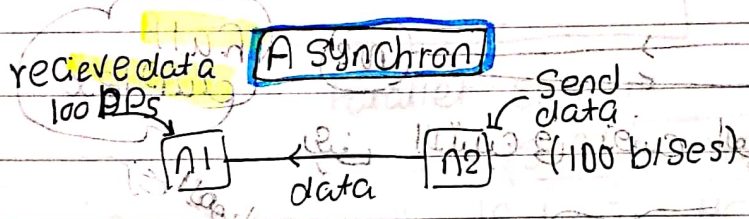
يعني معناها تزامن (يعني احنا الاتنين عندنا نفس الزمن)
يعني لو قولنا Two nodes بيكونوا SYNC
يعني هما الاتنين شايفين نفس الزمن

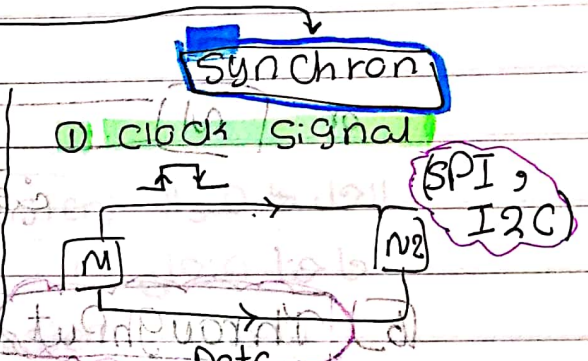
*



لازم N2 تستلم في نفس الزمن (بنفس السرعة بتاعة N1)
علشان يفهموا بعض صح
وده شرط في ال Communication.

* طيب دا ممكن نعمله من خلال ها طريقتين





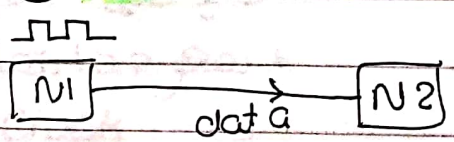
(هنا احنا الاتنين شايفين نفس ال Clock.)

Send → raising edge

recieve → falling edge

فكدا هنفهم بعض

② Frame Synchroization



هيبعت 2 Pulse قبل ما يبعت ال data

as LIN

سلك واحدة و sync

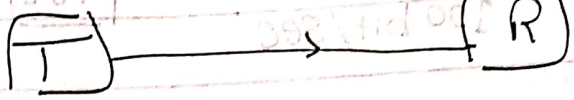
الـ Sync

* فيه

* Dynamic Data Rate ...!

لأن N1 ممكن أبعت بسرعة مختلفة

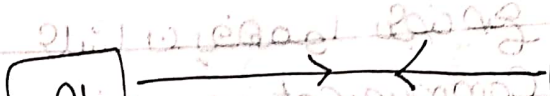
5 Data Direction



Simplex

(يعني data) متحرك في اتجاه واحد بس

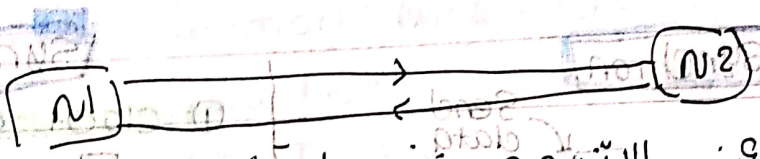
or



Half duplex

الاتنين بيكلموا او بيسمعوا بس مش في نفس الوقت.

or



Full duplex

يعني الاتنين في نفس الوقت

زي الموبايل

6 ThroughPut

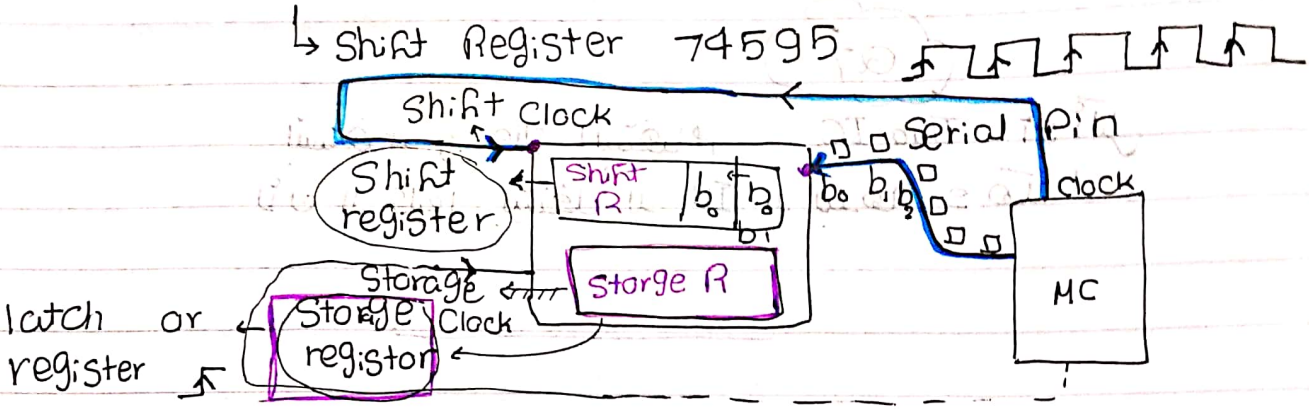
نسبة مئوية بتعبر عن قد ايه من الداتا مهم

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

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 هنخلى 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 ال

ويدخلها عنده فى ال Shift register

Slave

، بمجرد ما يتملى ، هيقوم ال MC يبعت ose

ل Storage Clock (واحدة بس)

SS then Pulse recieved on Storage Clock :

١ Data Copied from Shift register

To Storage Reg.

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

Q7

لتستخدم في حالة استخدام Two IC أو أكثر
لأن الـ enable بينتقل لـ IC الـ بعدها و هكذا

lec 19

Communications

SPI

* TFT
* SD card

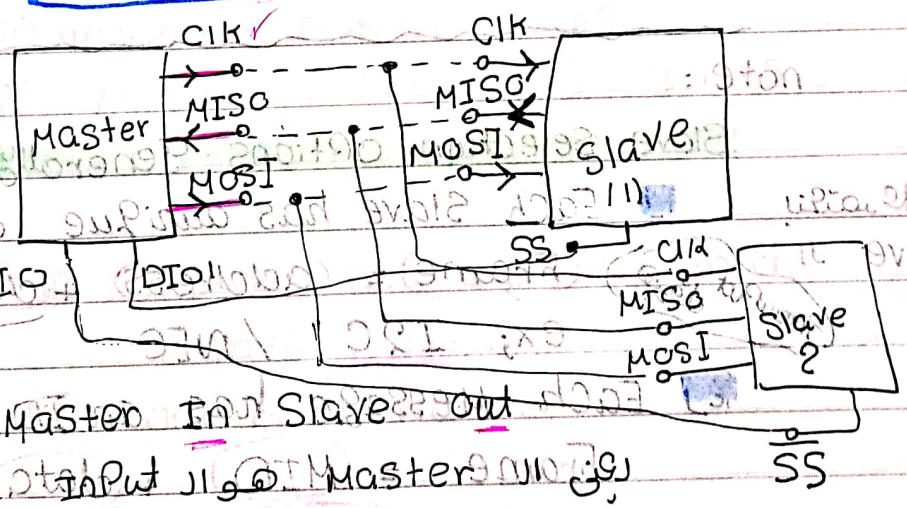
1) Serial Peripheral Interface :

→ 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 ٤ سلكات

SS : Slave Select (active low)

ومن خلالها ال Master بيختار أنه

هيكلم Slave

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

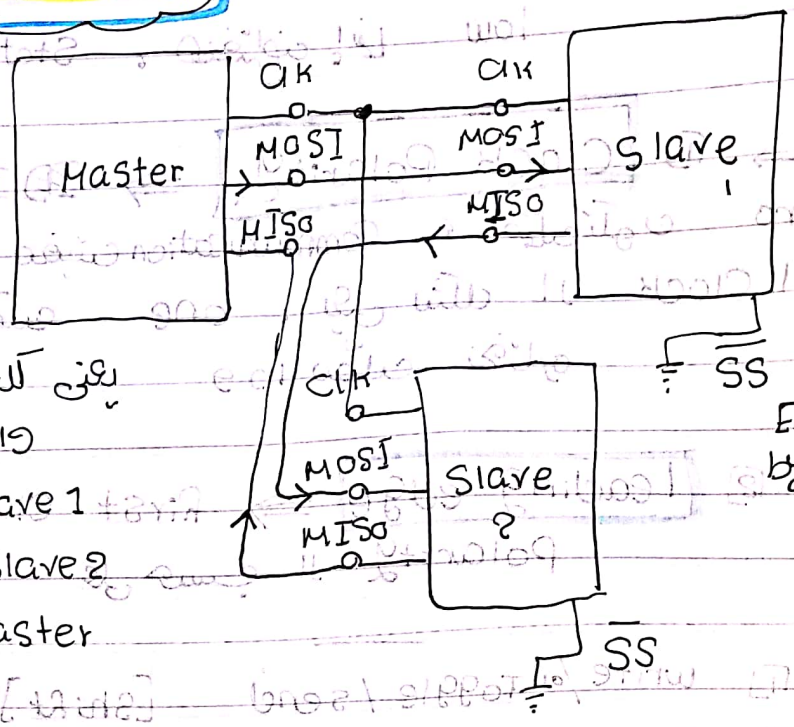
Advan...
1) Simple
2) Direct

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

note :
Slave selection options generally :
1) Each slave has unique address
Frame = address + data
ex; I2C / NEC
بنعتمد على إن ال Slave ده
Throughput هيقل
توفير hardware
2) Each message has an ID
Frame = MID + data
ex; CAN / LIN
3) Hardware selection → Chip select
→ Slave select
Hardware pin for each slave
Throughput أعلى وسرعة أعلى
no broadcasting
Hardware استخدام

[2] * Daisy Chain

الطريقة التانية ل SPI



Enable by default

يعني كل واحد بيكلم node واحدة بس

Master → Slave 1

Slave 1 → Slave 2

Slave 2 → Master

Adv

[1] unlimited num of slaves

[2] Master consume only 3 lines

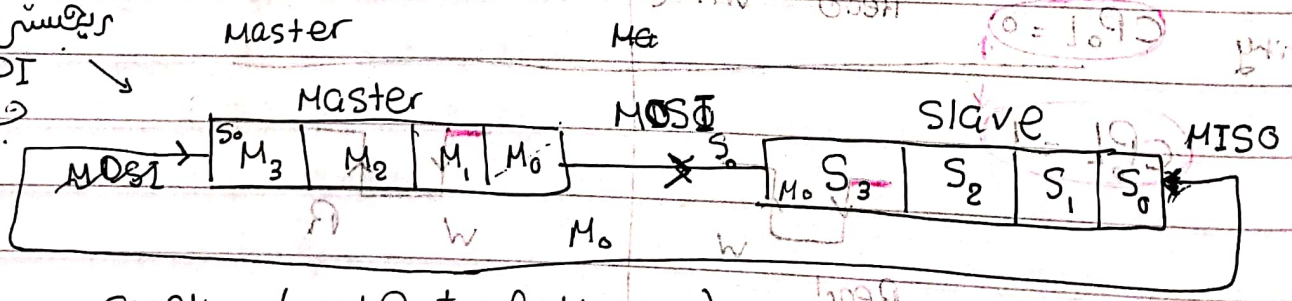
Dis

[1] Transfer Time depends on the slave position

[2] one point failure

[3] Frame Format

ريجستر بتاع SPI جوا M



Clock (output of Master)

Clock

في الأول ايوا State ، هنفترض إنها low

→ ① **Clock Polarity** ⇒ ID idle State of clock

الحالة ال او مفيش Communication ، ممكن تكون Zero و

ممكن one ، يعني شكل ال Clock ايه

و دا ممكن نختاره

→ ② **leading edge** ⇒ first clock edge ↑ or ↓

على حسب ال Polarity

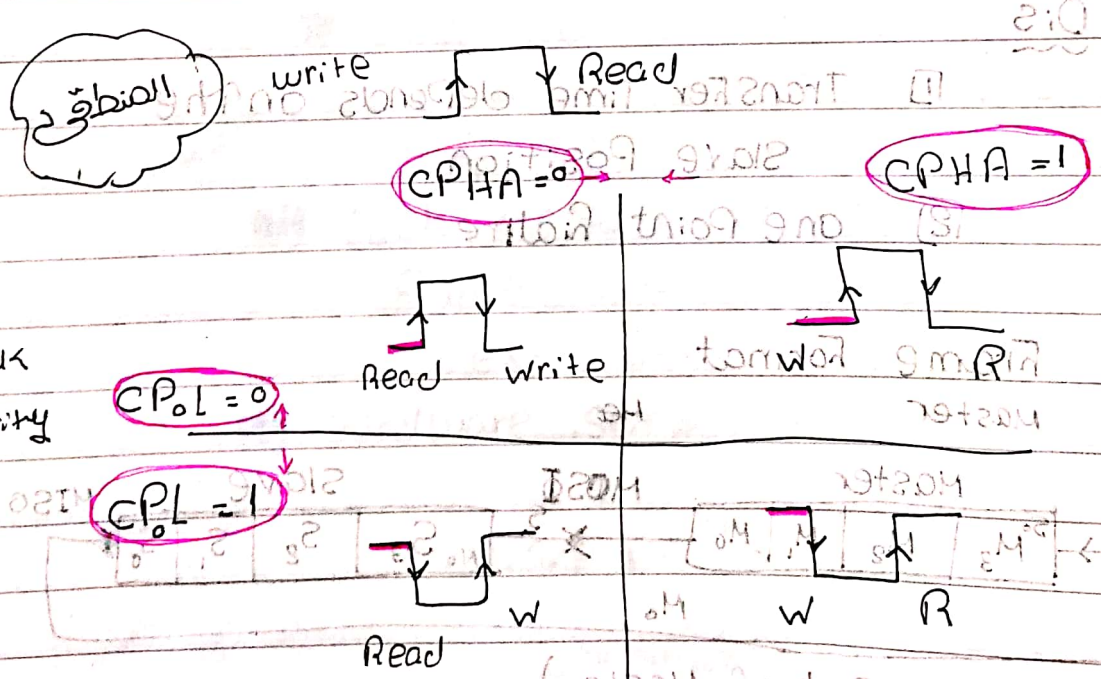
---

1. write / or Toggle / send [Shift]
2. Read [from Slave]

→ ③ **Clock Phase :**

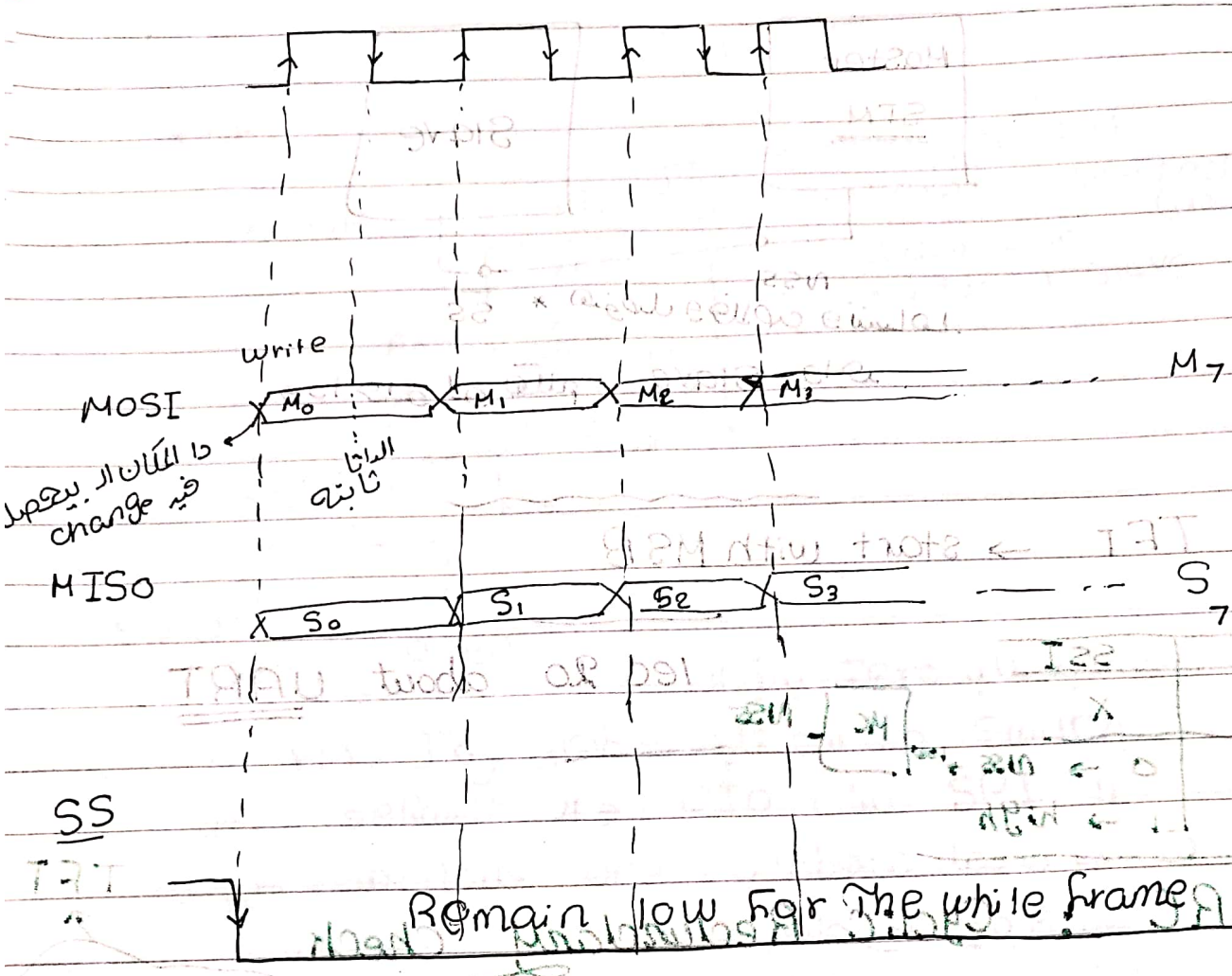
what to do first @ leading edge.

المنطق



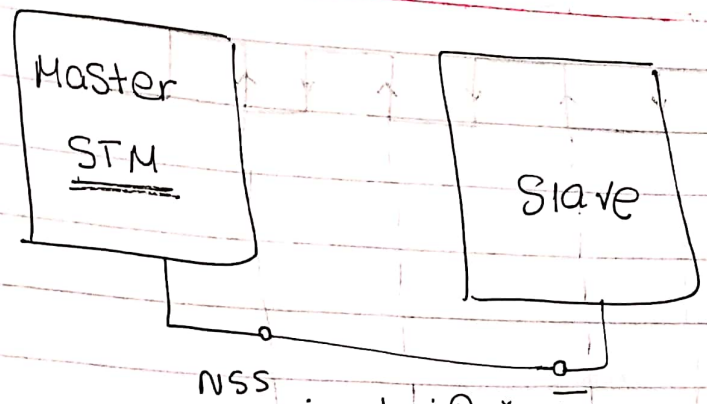
Slave وال Master لازم يكون ال

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 ونساها.

ودا حلو لو هتكلم Slave واحد

TFT → start with MSB

	SSM	SSI
H.W	0	X
SW	1	0 → NSS → low (MC) NSS 1 → high

lec 20 about UART

CRC : Cyclic Redundancy Check

SSM : Slave select Mode

0 → H.W

1 → SW → Portable

or → Gnd / DIO Pins

LSB : 0 → MSB → TFT

1 → LSB

SSI : هي قيمة ال Slave Select في حالة انها

Controling

~~H.W~~ SW

0 3 4 7

0000 0011 0100 0111

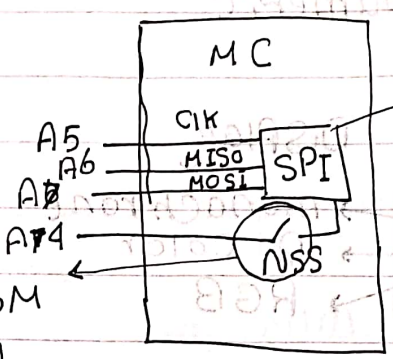
Lec 21

TFT

* SPI

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

ممكن يشتغل
Master or Slave



هو SSM

0 → Switch closed

1 → open

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

كدا الـ NSS قيمتها مش طالع برو ، وقيمتها هنا على حسب قيمة الـ SSI فهي.

لو احنا شغالين slave وهنبقى SSI بـ high لازم Master

علشان NSS (active Low)

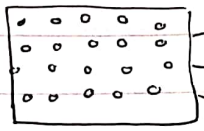
as Master

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

Display

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

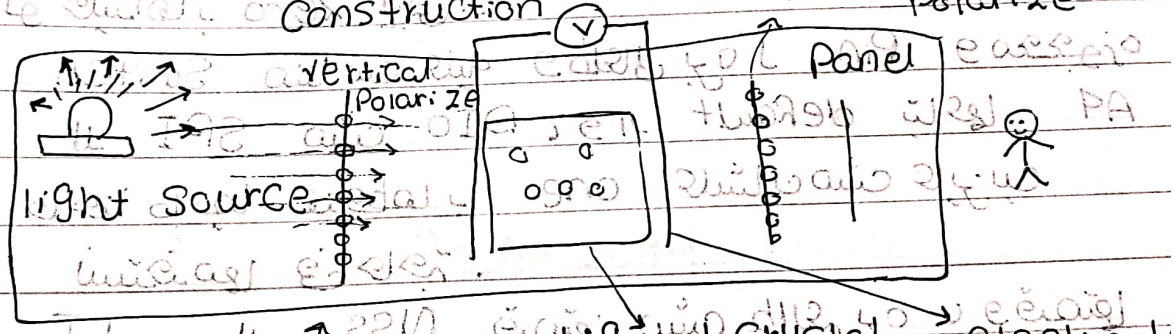
2) LED Matrix Display



→ monochrone
→ Bi color
→ RGB

3) LCD

Construction



Polarize

عبارة عن سائل فيه جزيئات لمادة اسمها Crystal (طبيعية)، لما يعملوا عليها Apply volt بتبدأ 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.

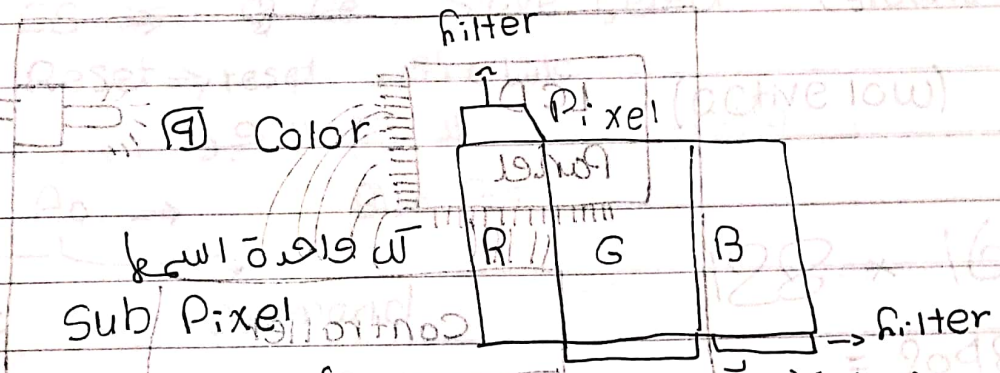
عبارة عن System كل لون يوحد اتجاه اللون الخارج

light Source → Halge
→ LED

(2) Contrast التباين (vs) Intensity

(3) Thin film Transistor

(1) Color

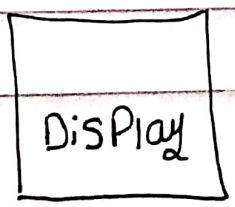


كل واحدة اسمها
Sub Pixel

أي لون في الدنيا يمكن تحويله لـ 3 ألوان دول بس بنسب مختلفة (Red, Green, Blue) لو كلهم صفر يبقى اسود
لو كلهم 1 يبقى أبيض

(1) Touch

علمياً مفيش حاجة اسمها Touch Screen (دا اسم تجاري)
الـ Touch هو عبارة عن Display أياً كان نوعها (CD, TFT)
بنحط عليها حاجة اسمها Touch Panel وهو عبارة عن
لوح شفاف معمول فيزكلي بشكل معين



+ 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 ⇒ ~~Se~~ Slave Select (Ground)

→ Reset ⇒ reset علشان أعمل لل IC ال جوا (active low)

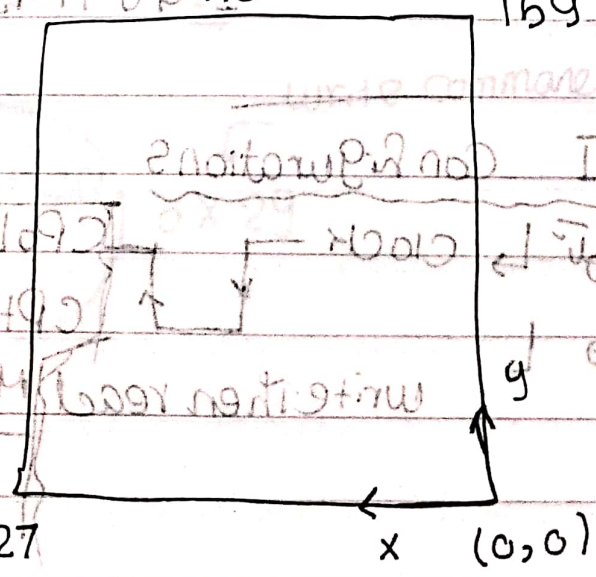
→ A0 ⇒ Dio Pin

Data — Command

1 — 0

128 * 160 = 20480

LCD = 1.8 inch

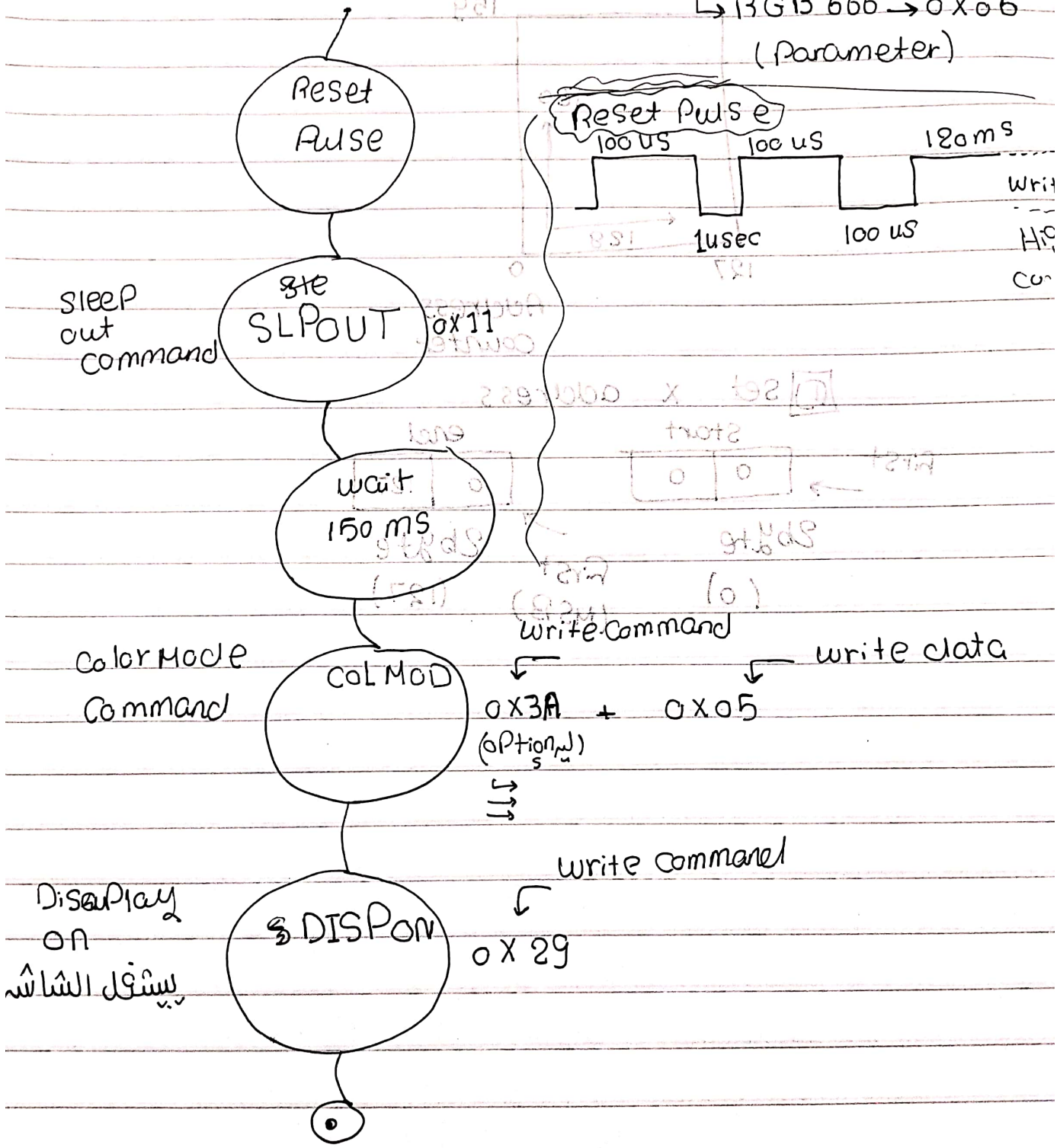


1 Pixel = R G

Init sequence :

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

(Parameter)



Lec 22

IoT

* web application

Networking

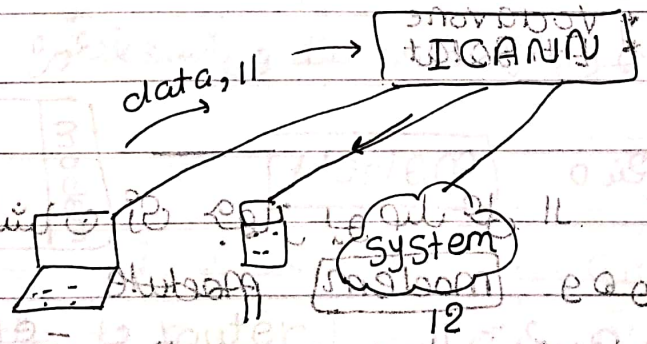
" Internet "

→ 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

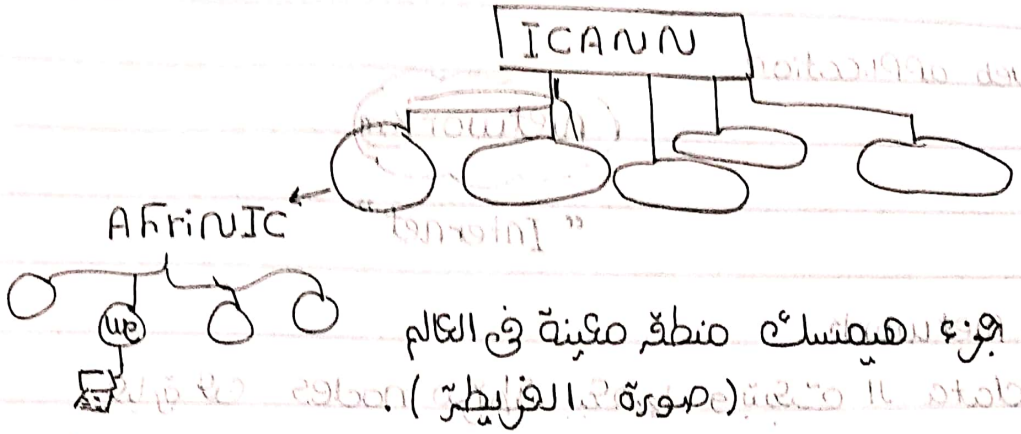


المشكلة إن العالم كله متوصل على ال internet وكلهم المفروض معب

هنا بقى ال ICANN قالت احنا هنجزء العالم بتاعنا لـ 5 أجزاء

ICANN هي الي ذكرتها وتحتها 5 دول الناس الي بتوزع ال IPs للعالم كله

Regional ISP (Internet Service Provider)



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

⇐ احنا مثلاً في مصر ال AfriNIC هي الا بتبعت لـ ICANN

وبعدين كل جوا كل دولة فيه

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

ex
* WE
* orange
* vodafone
* Etisalat

وعلشان أي جهاز يوصل على ال internet ، لازم يكون عنده Module modem وهو جهاز بياخد ال data من الكمبيوتر ويرميها على Physical medium

علشان يبعتها على ISP بتاعك

modem

wired → Telephone (بنستغل فكرة وجود التليفونات)

wireless (3G, 4G)

Each modem has IP → internet Protocol address

IP

هو عنوانك وهو unige ومحدش تانى ياخده

طيب ازاى هيكون ال IP بالعدد دا كله وهو unige ؟

هو ال IP

↓

4 Byte

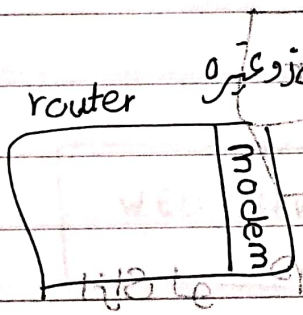
2^{32} = 4 مليار وشوية مثلا

والناس أساساً حوالى 7 مليار ؟

قالو هنعمل حاجة اسمها

router

هو جهاز بيتحط فى أى مكان مثلا فى البيت زو غيره



Modem

ال router دا عنده

وال device هو هيبعت لل router فى الجهاز هيبقى

فيه NIC

(network interface controller)

wired

wireless (wifi)

ethernet فى كابل ال

B.B.B.B

0 : 255

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

[2] تانى حل لتوفير ال IPs

Dynamic IP

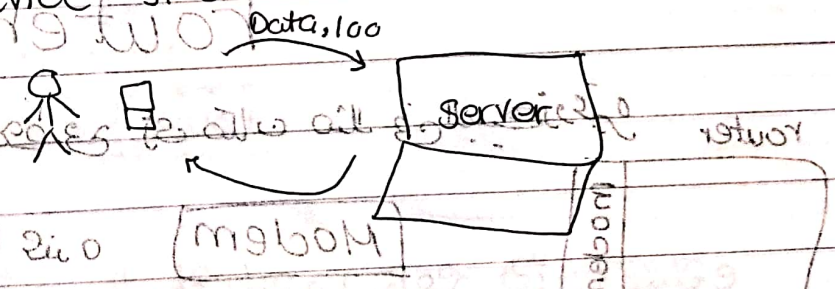
لو قفلنا ال router وفتحناه تانى هنلاقى ال router واخد IP Global تانى

Client Server

تشغيل mod

ازاى

ال server عبارة computer متوصل بال internet و الانسان ال مكان ال device هو Client



ازاى لو ال IP ال Dynamic يعنى هيتغير متواصل مكان ال Client

غيبه ناس قالوا خلاص بعض ال nodes هنحليلهم Static IP

ويدفع فلوس ال ISP علشان ياخد ال Static.

DNS (Domain Name Server)

Domain	IP
facebook	---

Modem شبه

أشهر الـ Protocols اللي هنتكلم بيها مع الـ Server ←
HTTP
وفي كمان FTP

في عالم الـ web دا اسمه Front End الحاجة اللي بشوفها Front end
والعكس 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
دول بقلوس
↳ Microsoft Azure
↳ CISCO
↓
open Source

2) Native
دا الهنشتغله

website ← هنجيب developer يعملهولي
↳ simple APP

Connect ⟷ Embeded ~~device~~ device

HTML & PHP

← هنعمل website بسيط علشان نتبسط بيها *

Website Implementation

علشان نعمل website

↳ Server
↳ hosting servies

Static IP
PC

↳ ② Domain

Freewha.com

webseeka

pw * ayaramzy

www.webseeka.freevar.com

HTTP → Server

← ودا علشان أستعرض الداتا ال على server

بيجي من ال Browser
هنستخدم (Chrome)

FTP (file zila)

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 لها أدوس على اللينك مثلا هى تنادى على الـ Script الفلانى

```
<? PhP

    echo "      ";

?>
```

PhP variables

$ X = ---

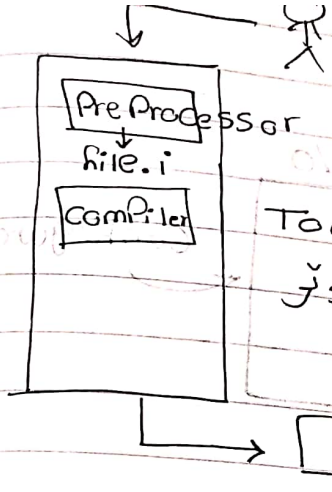
array

$ arr ['X']

$ arr ['y']

element		assesiation
X	→	0
y	→	10
Z	→	30

$_GET ['status']



high level language
يعني قريبة من لغة الانسان

Tool Chain
كلها بترن على الكمبيوتر
(SDK)

machine langu...

: Tool Chain

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

Tool Chain *

ال C بتتكون على الأقل 4 من ال Tool

1 PreProcessor

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

وهي Tool ←

* Search for PreProcessor Directive

* واللي فيها 11 ← PreProcessor بيبدأ ب #

#include #define #undef

#ifndef #elif #else #if

#ifndef #endif

#warning #error

بيدور عليهم ويعمل TEXT replacement ويحط محتواه

* Programa ← ملها ش علاقة بال PreProcessor

بيعمل TextReplacement ويشيل ال * ، لو * Pragma هيسيبها زي ماهي.

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

* البريبروسيسور هيطلع file.i (intermeduate file)
مكتوب بالسي ، لكن مفيهوش *
والباقي موجود زي ماهو 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 حاجة مهمة في ال الكود بيسميها Token
والباقي هيتمسح زي المسافات الفاضية والكومنت

[2] Syntax Analysis
بيشوف ال Syntax صح ولا لا

[3] Parsing int x = 10;
هو عملية تحويل سطر السي ال مكتوب لسطر ال Assemply ال بيقابله
Push

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 يعني أنا بقدمه لوحده جايز

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

{

x++;

}

logical Address at stack

push 0

compare i,5

jump

وهكذا

رغم إن هملته تكمل x += 5

هنا بقى الـ compiler هيستبدل الكود دا بدا

وهنا مرحلة الـ optimizer (وهو مراحل ودي صورة منه)

optional

يعني بيعمل بيحاول يعمل نفس الـ يتعمل

لكن بطريقة أفضل والـ optimizing

(optimization levels)

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

* عايز أشوف بقى file ← ال .ASSEMPLY file ال
* بالكود عند ال في السي.

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