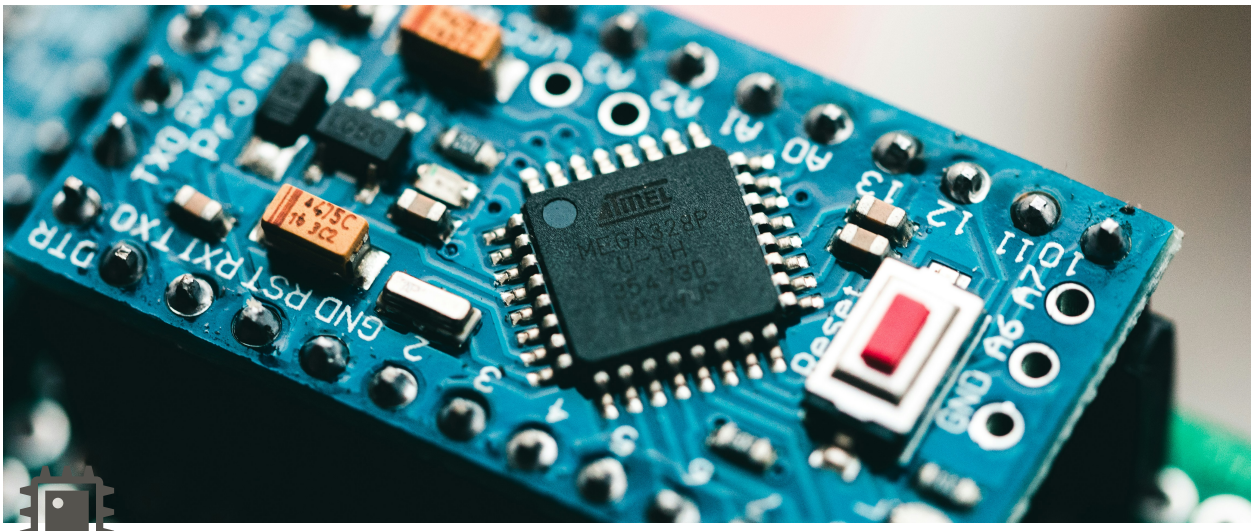

Prog Sys Embarqué

https://rex.plil.fr/Enseignement/Systeme/Systeme.PSE/

On utilise des µC à 2ko de mémoire

1ere semaie plus théorie

2eme plus programateur

3eme plus mannette

Quand un µC a un sys de protection de la mémoire c'est un µP.

En facile quand ça fait tourner linux correctement c'est un µP :)

cf : "3 Microcontrôleurs AVR : fonctionnalités" sur le link

pr une correction plutôt ok des tp µP

amr (pr ex rasberry) → catégorie m c'est des µC et autre µP

avr → principalement µC 8bits

On est sur des µC STM32F0

☝️voir les spec sur le link

Conception et real on commence au .4 car routage déjà fait

Mémoire flash → mémoire de masse, c'est là où est le prog

von nueman mémoire flash et mémoire de donnée dans la même mémoire

quand pas mm endroit harvard

Ds un flash à la base de la mémoire on a le prog

- le boot loader qui est un prog qui sert à en intégrer un autres. ex : sur arduio quand on flash il voit une activité sur son port spi, le bootloader (chargeur) sert à lui dire de checker ses port séries pour accérire le prog (cf "cas arduino")
- au dessus tout les vecteur d'interuption (par ex le reset, cad quand tu pousse le btn reset ça créer une interupt et ça exé le sous prog reset)

autre intéruption : port série (prêt à lire ou pas) etc

Petit point sur les bus

différent bus

- bus série : com asyncrone (pas d'orloge que des donnée) sur trois fils (rx tx et gnd pour accordé les masses) ⚠ un trouple de fil pour un sens de com

- bus spi ; un bus maitre et plusieurs bus esclave (syncrone)
  - ligne mosi : parallèle avec en dessous
  - ligne miso : master in slave out
  - ligne sck : clock
  - ligne SS : ship select

  ☝ Le master envoie un bail sur SS pour dire avec quelle slave il veut parler et ce slave discute via le miso/mosi

  ⚠ une ligne de selection par périphérique

- bus i2c : quand un esclave répond il rep sur la même ligne que celle qui l'a appelé

  ça implique de faire une diff entre les information (température ou info mes couilles) et les donnée de contrôle (hola les gars je veux parler au capteur de temp)

- reste bus usb avec à balle de donnée de contrôle

> 💡 Cas arduino
>
> Les microP, d'usine se programe en SPI. L'arduino lui, et ça a fait sont succès, est programmée en liaison série. Il faut donc un conv 🧠

Mémoire RAM :

- PIle d'execution → c'est la fameuse pile vu en cours (⚠ si trop grande peut empieter sur donnée global)
- Donnée global → c'est là ou les variable sont stock et tout
- Registre CPU → les registre de l'avr sont répliqué ici (de l'octet 0 à 32) on peut donc retrouvé le DDRD, PORTB, PINB etc
- Registre étendu → les registre pour la lecture de donnée analogique par ex

---

USB

Controleur USB → gérer par un ensemble de puce anexe au P. (intel incapable de faire de l'usb [acheck])

👉 btw AVR ça passe. (acheck aussi)

(cf parti USB du cours déjà bien complet) 👍

> 💡 Sur usb c une puce gère l'alim et une puce gère les infos

😎 fun fact c'est le pc qui fait la requette au clavier pour savoir ce qu'il a écrit

Architecture Périphérique USB

Les périphérique ont des descripteur (comparable à une SD en C avec different champ changeable)

```
lsusb
```

☝️énnumère les bus usb connecté à l'ordi

```
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 8087:0aaa Intel Corp.
Bus 001 Device 002: ID 1bcf:2cdb Sunplus Innovation Technology I
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

☝️

Num du bus utilisé	affecté à la connex c'est ce chant qui sert à diff les periphérique	id vendeur:id produit	Blaze ou version j'ai zap

Numéro device ne dépend pas de l'id il est négocier à la connexion

👉 le num/descripteur de configuration décrit l'alimentation requise pr fonctionner (peut en avoir plusieurs mais rarisime

👉 descripteur de sous périphérique/de classe : décrit les utilisation pr exemple clavier ou souris mais encore plus la qualité désiré d'une webcam ou la nature de ce qui est capté

Un périphérique peut avoir plusieurs interface (clavier souris intégrer ou justement webcam qui prend le son)

On branche un port usb :

- le contrôleur (ici pc) aloue une adresse
- recupère le descripteur de config pour savoir comment alim etc
- recupère le descripteur de classe pour savoir comment l'utiliser (par ex : eh g c'est un uid → un clavier → qui fonctionne comme ça → 💥 bam **endpoint**)

  Imaginon pour une wecam on a autant de descripteur d'interface de sous interface que de résolution séléctionnable

💡 Parfois une webcam demande une certaine bande passante assuré. C'est ce qui peut parfois ralentir la clef usb.

Voir 5.8 pour les endpoint

```
lsusb -d 1bcf:2cdb -v
```

☝️ affiche le descripteur d'un périphérique

Un périphérique utilise un endpoint pour communiquer soit en IN soit en OUT.

Si le périphérique subit des entrée **et** des sortie, alors il utilise plusieurs endpoint.

rav : Préscalaire, val au bout de laquelle le compteur se décrémente

Cross compil → compiler un truc sur autre machine que celle qui exec le prog

Compilation pour avr :

cf 6.2

1) Préprocessing

Petite commande :

```
avr-gcc -E -I. -DF_CPU=16000000 -mmcu=atmega328p -Wp,-P -Os time
```

☝️ **-P** sert à afficher les erreur et leur num de ligne

(btw préprocecing = enlève les #define et autres fiuriture) on a du c bien bas niveau

2) Génération de l'arbre syntaxyque abstrait (AST) :

Pour un processeur il est plus facile de taffer sur un arbre que sur un texte 🤷

Via les arbre il y a une optimisation

👉 cf 6.5 pour voir plus exactement ce que fait le préprocessing

Il existe plusieurs niveau d'optimisation, le -O sert à le choisir dans la commande avr-gcc

3) Création de l'assembleur intermédiaire

Créer un code assembleur pour un µP virtuelle qui aurait un nb infinie de registre.

4) Création du réel assembleur

5) Boom baby l'executable est là 😎

Il se peut qu'on ait un objet d'extention **.o**, la diff est qu'un objet n'a pas le nécessaire pour fonctionner seul (par ex pas de main)

.data → variable global hormis celle def à 0 (ça part dans la RAM)

.text → c'est le code généré qui rentre dans la machine (ça part dans la flash)

à check qu'est que .elf (je crois que c'est le .text collé au .data en hexa ascii)

Edition des lien :

Symbole → variabl ou fonction

Bibliothéque → ensemble d'objet