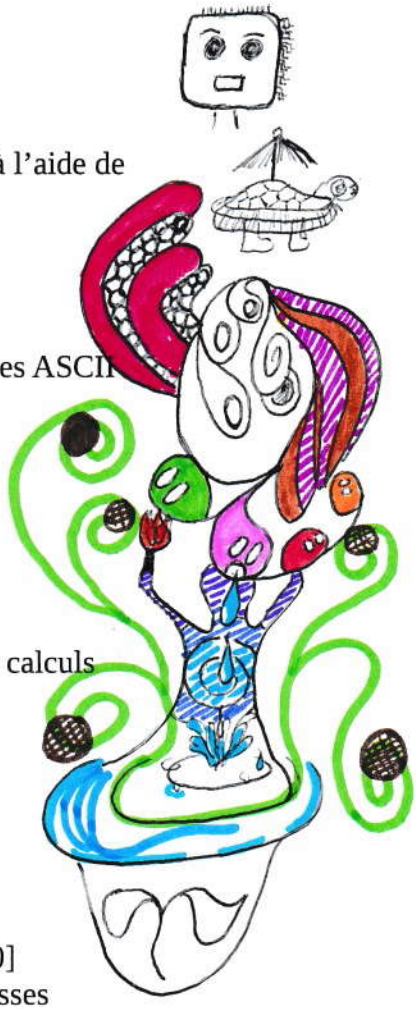
Informatique Industrielle UPSITECH 2023



Organisation :

Volumes horaires : 10h Cours + 20h TD/TP en demi groupes
Évaluation : Note de TP en binôme + 2 notes TD/CM
Malgré le nom du module, nous allons faire de l'informatique EMBARQUÉE à l'aide de microcontrôleur

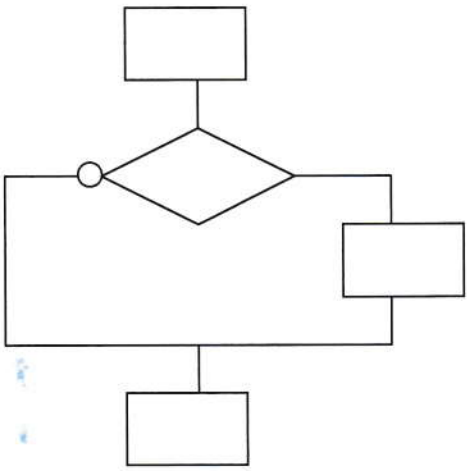
Pré-requis/rappels/bilan initial :

1. Bases de numération et changement de bases:
   Décimal, Binaire, Hexadécimal, BCD (Décimal codé Binaire), caractères ASCII
2. Portes logiques et portes 3 états (Tri state) Notion de haute impédance
3. Mémoires: Dispositif de stockage de l'information
   - Mémoire morte : ROM / PROM / EPROM / EEPROM /Flash
   - Mémoire vive : RAM
   - Bus adresses / Données / Commandes
   - Capacité mémoire: nombre de cases et taille des mots
4. Langage de programmation C/C++
   - Algorithmie: Condition, test, boucle, fonctions, variables, opérations de calculs
   - Opérateurs:
     - booléens (&&, || , !)
     - logiques (&,|,~,^) pour le masquage
     - de décalage (<<,>>)
     - arithmétiques: (+,-,*,/,%,++, --)
     - comparaison (==, !=, >=, <= , >, <)
     - affectation (=)
     - indirection (*) et indiçage ([]) , *ptr_var équivalent à ptr_var[0]
     - accès aux champs et attributs (. et - >) pour les structures et classes
   - Types:
     - entier unsigned/signed, sur différents nombres de bits, pour entier ou en virgule fixe
     - flottants avec différentes précisions
     - tableaux à une dimension et affectation lors de la déclaration
       indice allant de 0 à nombre d'éléments-1
     - pointeurs (type suivi de *), une variable pointeur ne stocke qu'une adresse !
     - structures et extension aux classes
     - vide
   - Variable = instance d'un type,
     - Notion de portée : déclaration locale ou globale,
   - Objet = instance d'une classe
   - Structuration d'un programme:
     - Fonction principale: void main(void), initialisation puis tâche périodique en boucle
     - Fonctions, type retour et paramètres, pointeurs pour les paramètres en E/S (type *)
     - Notion de librairie, organisation .cpp/.h
     - Pseudo parallélisme : fonctions non bloquantes pour les différentes tâches
       while(1){tache1(); tache2(); tache3();}
   - Documentation du code :
     - commentaires // simple ligne et /* multiligne */
     - règles pour la documentation automatique https://www.doxygen.nl/manual/
   - Gestion de version du code : GIT

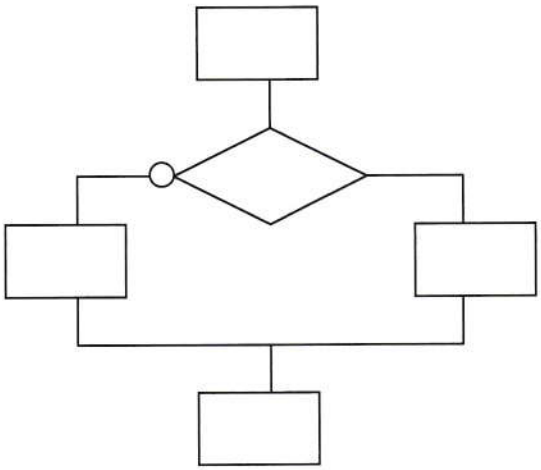
5. Plateforme et environnement de développement :
   - Arduino UNO
   - Simulateur
   - Périphériques (internes et externes)

6. Algorithmie

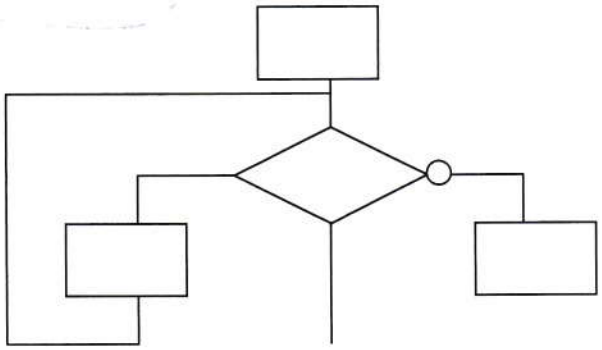
- SI ... ALORS



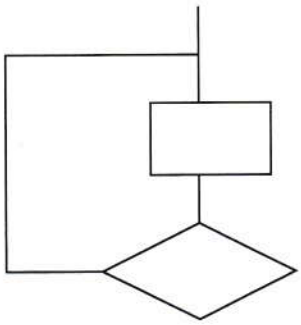
- SI ... ALORS .... SINON



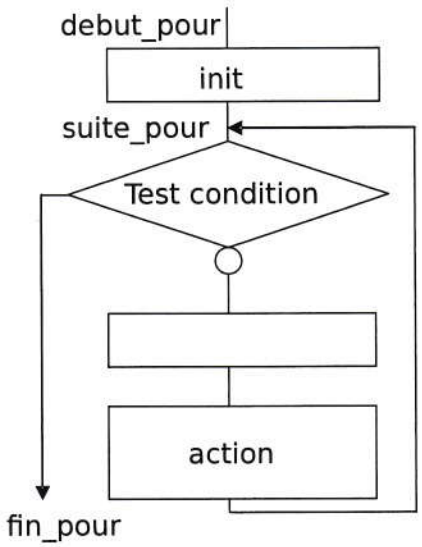
- Tant que .... Fin tant que



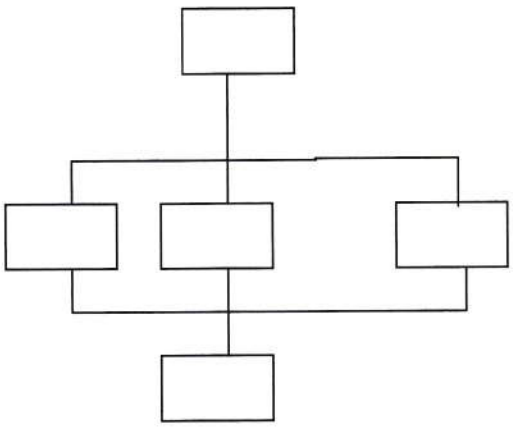
- Répéter ... Jusqu'à



- Pour (init, condition, action) faire...



- SUIVANT cas0 ... cas1 ...casn



Informatique Industrielle UPSITECH 2023

Chapitre 1 : Introduction aux microprocesseurs et microcontrôleurs
Application au ATMEGA328P

Bertrand Vandeportaele et Nicolas Rivière
Basé sur un document d'Alexandre Nketsa

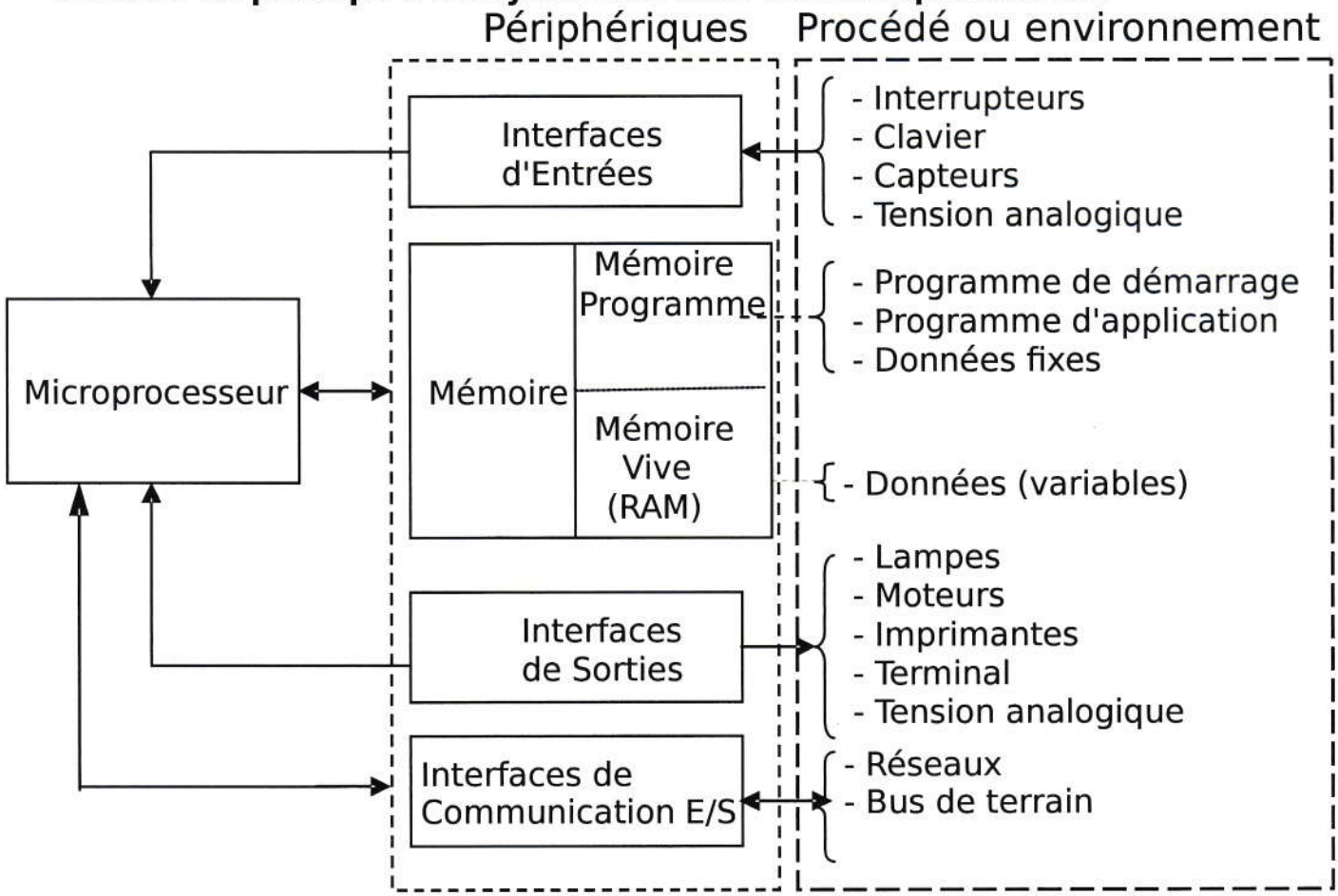
Partie 1 – Système à base de microprocesseur

Un système à base de microprocesseur est constitué de 2 blocs :
- un microprocesseur
- des périphériques que l'on peut organiser en 2 groupes :
    - la mémoire (morte et vive)
    - les interfaces d'entrée, de sortie, d'entrée-sortie (numériques ou fonctionnelles)

Cette architecture est aussi celle des ordinateurs, des systèmes embarqués, des systèmes de contrôle commande (par exemple des Automates Programmable Industriels).

Les domaines d'application des **systèmes embarqués** sont très nombreux:
- Avionique : exemple AIRBUS A380
- Automobile : ABS, contrôle des Airbag, véhicule électrique
- Ferroviaire, Métro, Bateau
- Médecine : appareils de mesure et de surveillance (monotoring) - implants
- Militaire
- Systèmes industriels, Robotique, vision par ordinateur
- Objets connectés
- Télécommunications, Télévision numérique
- Jeux, Téléphone, Internet mobile
- Domotique, Électroménager, Appareils photo
- Villes et bâtiments intelligents
- Vêtements connectés
-….

Schéma de principe d'un système à base de microprocesseur:



Partie 2 – Microprocesseur

Le microprocesseur est le chef d'orchestre.
La (ou les) mémoire contient :

- le programme (la suite d'instructions (ordres élémentaires) que le microprocesseur doit lire et doit exécuter en séquence.
- les données (les variables)

Cette exécution consiste à :

- prendre connaissance de l'environnement par l'intermédiaire des interfaces d'entrée (contrôle)
- effectuer des traitements (calculs et prises de décision)
- utiliser la mémoire pour stocker les informations permanentes ou temporaires
- influencer l'environnement en affichant les résultats ou en commandant des dispositifs (commande).

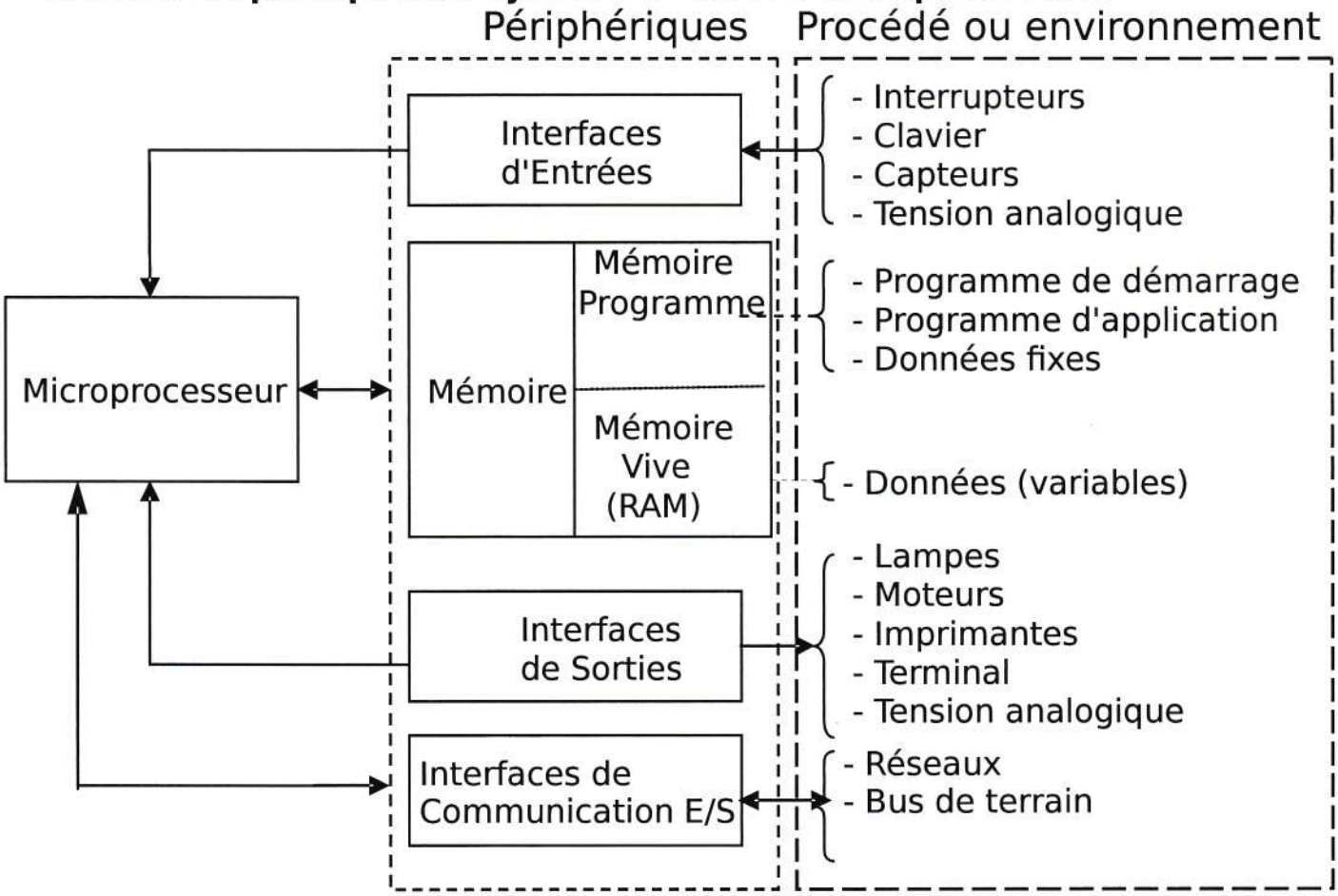
Schéma simplifié du microprocesseur

En interne, le microprocesseur est organisé comme les systèmes que nous avons vus en électronique numérique :

* une partie commande (que nous appellerons **séquenceur**)
* une partie opérative comportant :

- un **registre d'instruction** qui mémorise le numéro de l'instruction pendant son exécution
- un **compteur programme** qui contient l'adresse de la case mémoire contenant le numéro de la prochaine instruction à exécuter.
- une **unité de calcul** qui permet d'effectuer tous les calculs

Schéma de principe d'un système à base de microprocesseur:



Partie 2 – Microprocesseur

Le microprocesseur est le chef d'orchestre.
La (ou les) mémoire contient :

- le programme (la suite d'instructions (ordres élémentaires) que le microprocesseur doit lire et doit exécuter en séquence.
- les données (les variables)

Cette exécution consiste à :

- prendre connaissance de l'environnement par l'intermédiaire des interfaces d'entrée (contrôle)
- effectuer des traitements (calculs et prises de décision)
- utiliser la mémoire pour stocker les informations permanentes ou temporaires
- influencer l'environnement en affichant les résultats ou en commandant des dispositifs (commande).

Schéma simplifié du microprocesseur

En interne, le microprocesseur est organisé comme les systèmes que nous avons vus en électronique numérique :

* une partie commande (que nous appellerons **séquenceur**)
* une partie opérative comportant :

- un **registre d'instruction** qui mémorise le numéro de l'instruction pendant son exécution
- un **compteur programme** qui contient l'adresse de la case mémoire contenant le numéro de la prochaine instruction à exécuter.
- une **unité de calcul** qui permet d'effectuer tous les calculs

- des **registres de travail** :

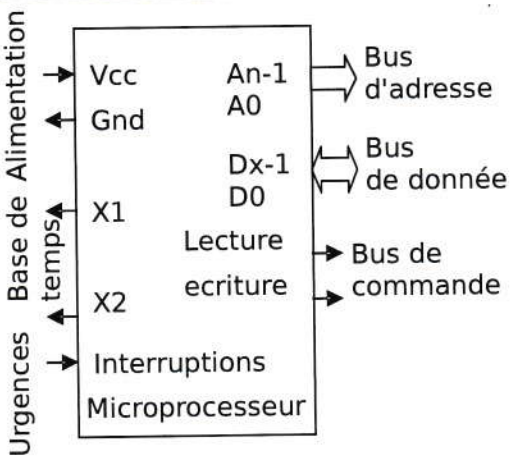
* les General Purpose Registers (registres à usage général) qui servent pour le stockage temporaire des données.

* Les Special Function Registers (registres spéciaux) qui servent à piloter et à connaître l'état de périphériques.

- un **pointeur de pile** (à titre indicatif ici) pour la gestion des appels de fonctions

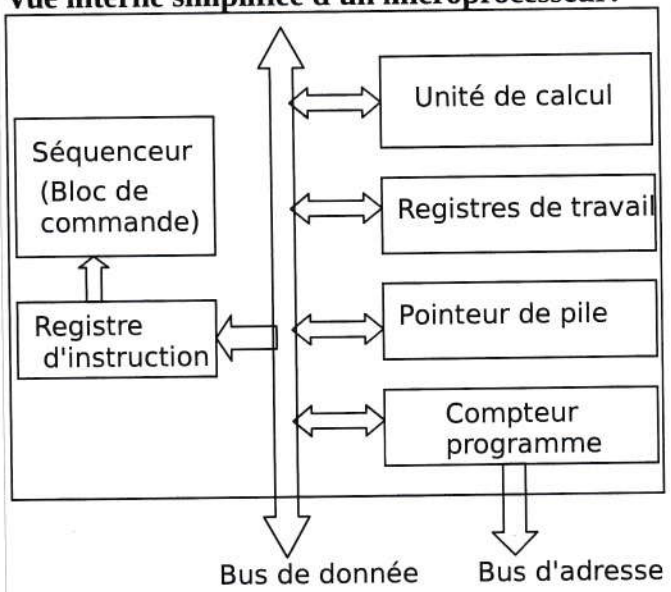
Définition :Un **bus** est un ensemble de fils qui:
- peuvent prendre des valeurs binaires
- peuvent être électroniquement déconnectés

Le microprocesseur est un circuit électronique ayant :
- un bus d'adresse pour choisir le périphérique avec lequel il doit échanger des données
- un bus de donnée pour véhiculer les données à échanger avec le périphérique choisi
- un bus de commande pour indiquer le sens de l'échange
  * lecture : périphérique vers microprocesseur
  * écriture : microprocesseur vers périphérique
- une base de temps (Quartz) (cadencement des activités)
- un bus pour les urgences (interruptions)
- une alimentation



La taille du bus de donnée et l'unité de calcul déterminent l'**architecture 8/16/32/64 bits** du processeur. La **fréquence de la base de temps** influe sur le nombre d'instructions exécutées par unité de temps (**MIPS**). Le jeu d'instructions est soit **RISC** (instructions simples et exécutées rapidement) soit **CISC** (instructions complexes mais plus lentes).

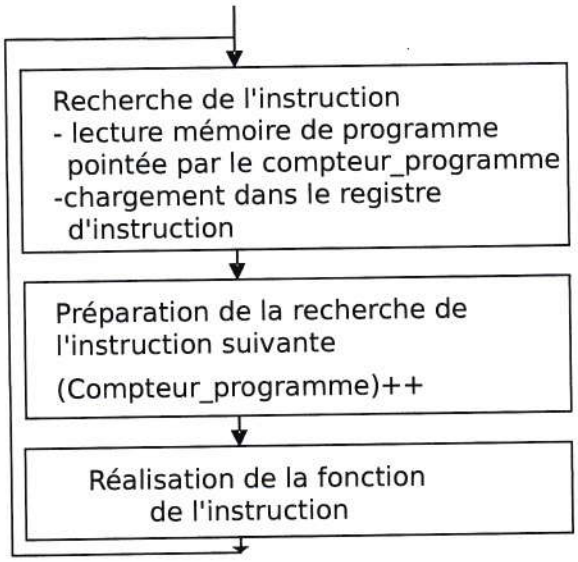
Vue interne simplifiée d'un microprocesseur:



https://fr.wikipedia.org/wiki/Architecture_de_von_Neumann (même bus pour les instructions et les données)

Organigramme du fonctionnement interne d'un microprocesseur:

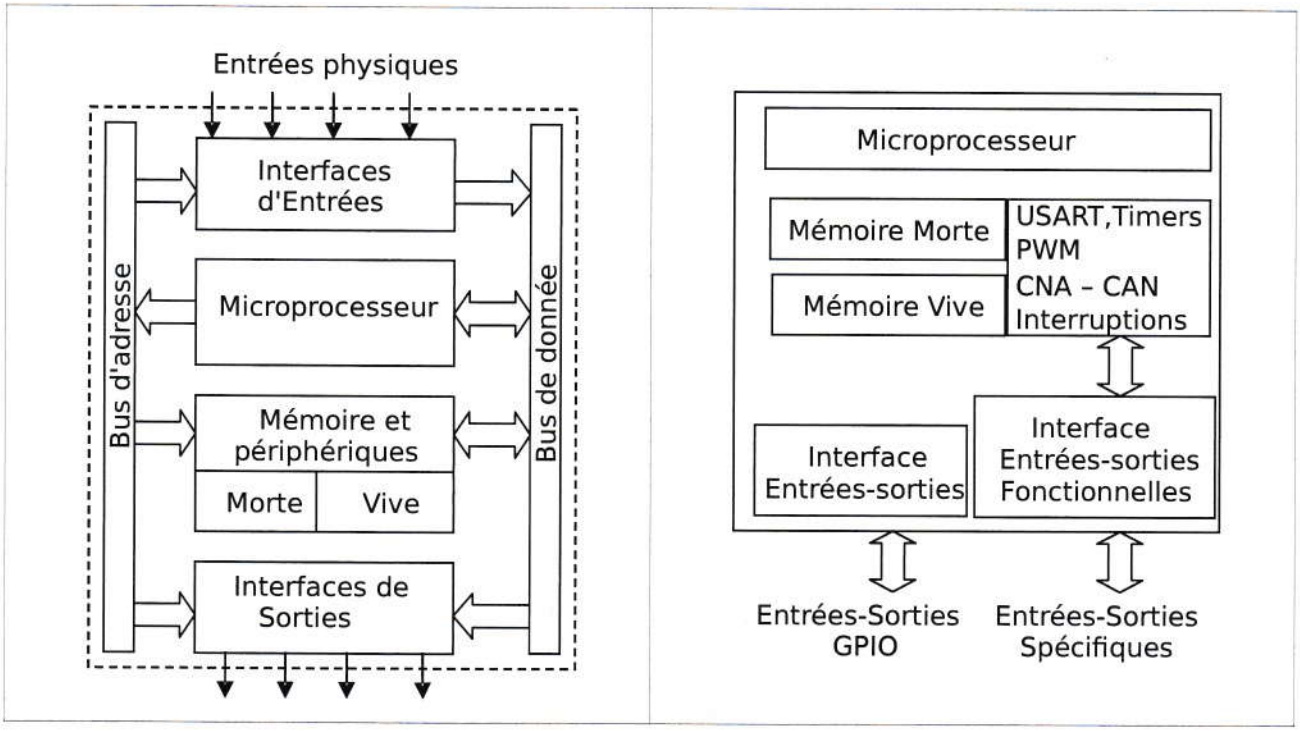
Il est basé sur la lecture et l'exécution des instructions stockées en séquence dans une mémoire (mémoire de programme)



Partie 3 – Microcontrôleur

Définition

Un microcontrôleur est une intégration dans le même circuit intégré :
- d'un microprocesseur
- d'une mémoire vive statique (SRAM)
- de la mémoire morte (EEPROM et plus souvent flash EPROM)
- des interfaces d'entrée-sortie numériques multifonctions (GPIO pour General Purpose Input Output)
- des interfaces fonctionnelles d'entrée-sortie (USART, TIMERS, PWM, CAN/CNA, demandes d'interruptions...)



Partie 4 – Application au microcontrôleur ATMEGA328P

Extraits de la documentation du ATMEGA328P.
Version complète sur: https://bvdp.inetdoc.net/files/cesi/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

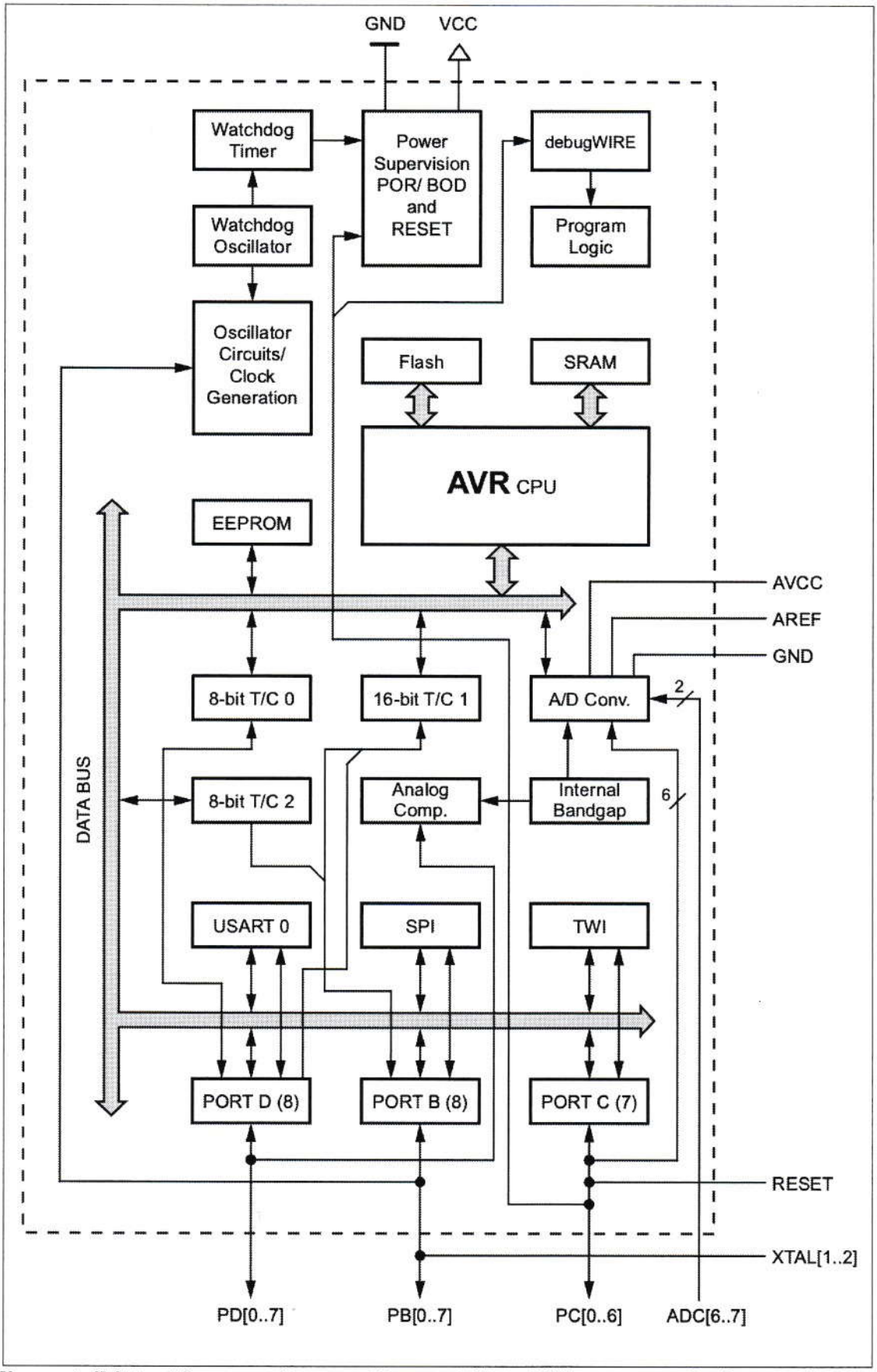
8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash

Features

DATASHEET

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32K bytes of in-system self-programmable flash program memory
  - 1Kbytes EEPROM
  - 2Kbytes internal SRAM
  - Write/erase cycles: 10,000 flash/100,000 EEPROM
  - Optional boot code section with independent lock bits
    - In-system programming by on-chip boot program
    - True read-while-write operation
  - Programming lock for software security
- Peripheral features
  - Two 8-bit Timer/Counters with separate prescaler and compare mode
  - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
  - Real time counter with separate oscillator
  - Six PWM channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature measurement
  - Programmable serial USART
  - Master/slave SPI serial interface
  - Byte-oriented 2-wire serial interface (Phillips I^2C compatible)
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
  - Interrupt and wake-up on pin change
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated oscillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby
- I/O and packages
  - 23 programmable I/O lines
  - 32-lead TQFP, and 32-pad QFN/MLF
- Operating voltage:
  - 2.7V to 5.5V for ATmega328P
- Temperature range:
  - Automotive temperature range: –40°C to +125°C
- Speed grade:
  - 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
  - 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)
- Low power consumption
  - Active mode: 1.5mA at 3V - 4MHz
  - Power-down mode: 1µA at 3V

https://



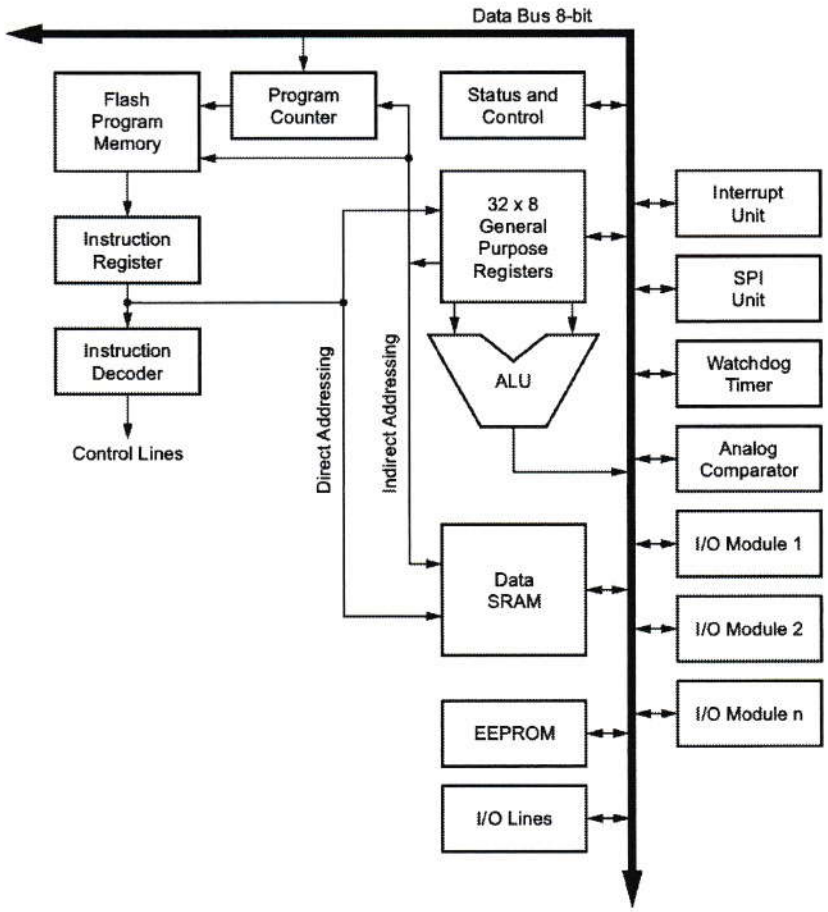
fr.wikipedia.org/wiki/Architecture_de_type_Harvard (bus différents pour les instructions et les données)

AVR CPU Core

Overview

This section discusses the AVR® core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Figure 6-1. Block Diagram of the AVR Architecture



In order to maximize performance and parallelism, the AVR uses a harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is in-system reprogrammable flash memory.

In-System Reprogrammable Flash Program Memory

The ATmega328P contains 32Kbytes on-chip in-system reprogrammable flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the flash is organized as 16K ×16. For software security, the flash program memory space is divided into two sections, boot loader section and application program section in ATmega328P. See SELFPRGEN description in Section 25.3.1 "SPMCSR – Store Program Memory Control and Status Register" on page 228 and Section 26.9.1 "SPMCSR – Store Program Memory Control and Status Register" on page 239 for more details.

The flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega328P program counter (PC) is 14 bits wide, thus addressing the 16K program memory locations. The operation of boot program section and associated boot lock bits for software protection are described in detail in Section 25. "Self-Programming the Flash, ATmega328P" on page 223 and Section 26. "Boot Loader Support – Read-While-Write Self-Programming" on page 229. Section 27. "Memory Programming" on page 241 contains a detailed description on flash programming in SPI- or parallel programming mode.

Constant tables can be allocated within the entire program memory address space (see the LPM – load program memory instruction description).

Timing diagrams for instruction fetch and execution are presented in Section 6.6 "Instruction Execution Timing" on page 14.

SRAM Data Memory

Figure 7-2 shows how the ATmega328P SRAM memory is organized.

The ATmega328P is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The lower 2303 data memory locations address both the register file, the I/O memory, extended I/O memory, and the internal data SRAM. The first 32 locations address the register file, the next 64 location the standard I/O memory, then 160 locations of extended I/O memory, and the next 2048 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, indirect with displacement, indirect, indirect with pre-decrement, and indirect with post-increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space.

The indirect with displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O registers, 160 extended I/O registers, and the 2048 bytes of internal data SRAM in the ATmega328P are all accessible through all these addressing modes. The register file is described in Section 6.4 "General Purpose Register File" on page 12.

Figure 7-1. Program Memory Map ATmega328P

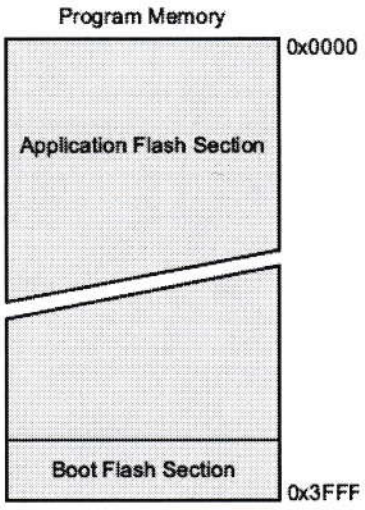
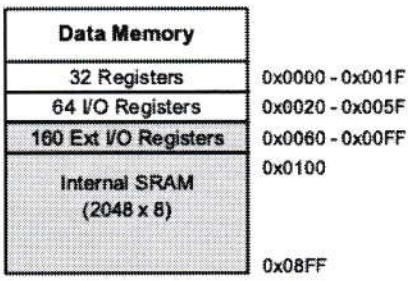


Figure 7-2. Data Memory Map



EEPROM Data Memory

The Atmel® ATmega328P contains 1Kbyte of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM address registers, the EEPROM data register, and the EEPROM control register.

Section 27. "Memory Programming" on page 241 contains a detailed description on EEPROM programming in SPI or parallel programming mode.