Le développement des projets

1) Introduction :

Bien que de nombreuses sociétés disposent de leurs propres méthodes de développement, la réussite d'un projet (qu'il soit une conception et une réalisation de type matérielle, logicielle ou les deux) passe par une bonne méthodologie d'analyse et de développement du problème posé. L'usage en vigueur dans le milieu industriel correspond à une méthodologie d'analyse de type « descendante » ou encore appelée « Top-Down » en anglo-saxon. En conséquence le déroulement d'un projet passe souvent par les étapes suivantes :

- élaboration du document de spécifications
- conception préliminaire
- conception détaillée
- implémentation
- tests unitaires
- intégration finale
- recette

1.1) élaboration du document de spécifications

C'est le document obtenu à partir du cahier des charges de l'objet à concevoir et à réaliser. Il s'appuie sur le cahier des charges d'origine et le complète de manière à ce qu'aucun flou ne subsiste quand à la finalité du produit à réaliser. Le document de spécifications est très souvent rédigé par le bureau d'études en charge du projet alors que le cahier des charges est fourni par le client.
Le document de spécifications du produit s'attache donc à décrire de manière exhaustive :

- le nombre et la nature des entrées-sorties (numérique, analogique, bande passante, amplitude des signaux, tolérances....)
- toutes les fonctions supportées par le produit y compris les aspects « temps réel » si ceux –ci ne sont pas explicites. (ex : telle fonction est réalisée en x millisecondes maxi...)
- les contraintes sur la connectique et les câbles utilisés si nécessaire
- le contexte environnemental (gamme de température de fonctionnement, de stockage, présence d'humidité, poussière, vibrations, parasites électromagnétiques (CEM)
- les besoins en énergie (présence ou non du réseau EDF par exemple)
- la signature électromagnétique (normes CEM)
- la durée de vie (MTBF : Mean Time Between Failure)
- ...

Dans l'industrie ce document est contresigné par les différentes parties prenantes du projet (client et bureaux d'études) et fait office de contrat.

1.2) La conception préliminaire

Elle a pour but de définir dans les grandes lignes l'architecture de la solution technologique qui sera retenue :

- réalisation d'un produit compact ou de plusieurs sous-produits interconnectés
- solution logicielle à base de processeurs, solutions câblées ou mixtes

- éventuellement choix des technologies voire des marques des principaux composants (processeurs Motorola, Intel, ...) si un savoir faire existe dans le domaine.
- choix des bus de communication (IEEE-488, VXI, VME, ethernet ...)
- ....

Cette phase doit en premier lieu asseoir la confiance en la finalité et la faisabilité du produit, et, en second lieu, servir de base pour l'estimation de la quantité de travail à fournir pour le réaliser.
La conception préliminaire s'appuie très souvent sur les savoir-faire du bureau d'études. A l'issue de cette phase, les spécifications « grossières » des sous-ensembles qui composent le produit sont rédigées.

1.3) la conception détaillée

C'est la phase correspondant à l'analyse fonctionnelle aussi détaillée que possible du produit ou sous-produit dont on a la charge de la conception. En principe, à ce stade de la conception, les solutions technologiques ne sont pas encore retenues (choix entre implémentation logicielle ou matérielle par exemple, achat ou sous-traitance de certaines parties). Chaque fonction doit avoir un plan de test unitaire, qui fournit aux réalisateurs la liste des tests à effectuer ou des types de scénarios de test a créer afin de vérifier que la fonction répond aux spécifications.
Voir exemple sur l'annexe jointe.

1.4) L'implémentation

Pour les parties réalisées par logiciel (software), c'est le développement des programmes associés aux différentes fonctions. A ce stade du développement on choisit le langage qui sera utilisé et on code les algorithmes et algorigrammes.
Pour les parties réalisées par du matériel (hardware), ce sont les schémas électriques des différentes fonctions, le routage, la réalisation et le câblage des cartes électroniques.

1.5) les tests unitaires

Ce sont les tests de toutes les fonctions, indépendamment les unes des autres, qui constituent l'objet à réaliser. Un soin particulier est apporté au test des modules logiciels et à leur fiabilité de fonctionnement.

1.6) l'intégration

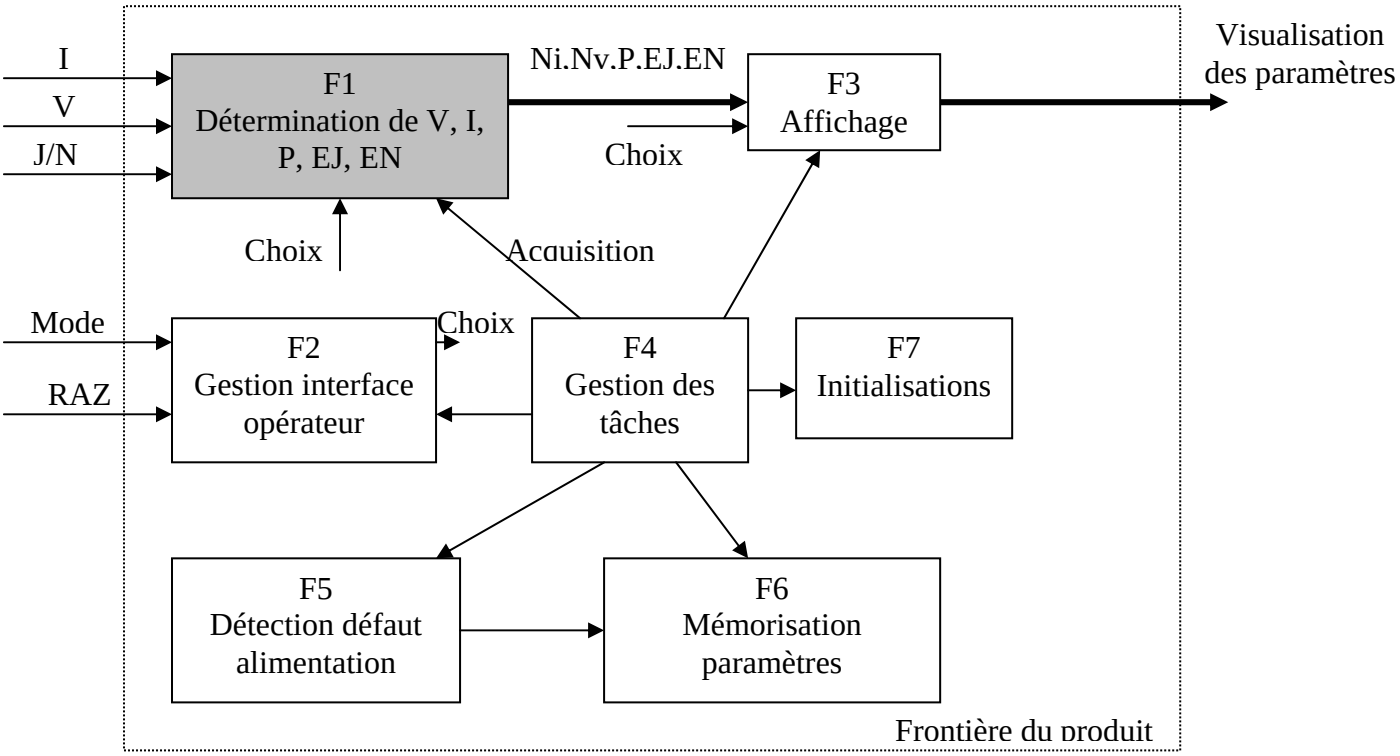
l'intégration correspond, dans un premier temps, au test du produit complet c'est à dire lorsque tous les modules sont connectés entre eux. Dans un deuxième temps le produit est testé dans son environnement simulé ou réel si possible.

1.7) la recette

La recette fait l'objet d'un document rédigé en début de projet, avalisé par les parties prenantes et qui s'appuie fortement sur le document de spécifications du produit. Le document de recette décrit de manière exhaustive dans quelles conditions et comment sera testé le produit fini. La signature du procés verbal de recettes par les différentes parties clôt le contrat (à l'exception des aspects de maintenance) et entraine la facturation de la prestation.

Exemple de conception détaillée :
le compteur-totalisateur d'énergie électrique

Schéma fonctionnel[1] associé à l'objet technique :



Détail des différentes fonctions :

Fonction F1 « détermination de V, I, P, EJ, EN »

Entrées :

- I(t) : signal de tension analogique pseudo-sinusoïdal de fréquence 50 Hz image du courant Ie(t) consommé par l'installation électrique.
  Relation : I(t)=k*Ie(t)
  Icrête=0V quand Iecrête=0 A
  Icrête=2,5V quand Iecrête=70 A
  Valeur moyenne de I(t)= 2,5V

- V(t) : : signal de tension analogique pseudo-sinusoïdal de fréquence 50 Hz image de la tension Ve(t) du réseau électrique.
  Relation : V(t)=k*Ve(t)
  Vcrête=0V quand Vecrête=0
  Vcrête=2,5V quand Vec= 340V
  Valeur moyenne de V(t)= 2,5V

[1] Il peut exister plusieurs schémas fonctionnels possibles pour un même produit

- J/N : signal logique indiquant le mode jour/nuit.
  J/N=0 => mode jour
  J/N=1 (+5V) => mode nuit.

- Choix : mot binaire prenant les valeurs :
  - 0 si le calcul de V efficace est demandé
  - 1 si le calcul de I efficace est demandé
  - 2 concerne l'affichage de P (non exploité par F1)
  - 3 concerne l'affichage de EN (non exploité par F1)
  - 4 concerne l'affichage de EJ (non exploité par F1)
- acquisition :
  signal logique de période 1s permettant l'exécution de la fonction F1.

Sorties :

- Ni : mot binaire codé sur 8 bits correspondant à la valeur efficace de I(t).
  1 LSB correspond à 1A efficace.

- Nv : mot binaire codé sur 8 bits correspondant à la valeur efficace de V(t).
  1 LSB correspond à 1V efficace.

- P : mot binaire 16 bits image de la puissance efficace en W
  P= 0 pour 0W et 1 LSB correspond à 1W

- EN : mot binaire 16 bits image de l'énergie-nuit en KWH
  1 LSB correspond à 1 KWH.

- EJ : mot binaire 16 bits image de l'énergie-jour en KWH
  1 LSB correspond à 1 KWH.

Rôle :

La fonction F1 a pour rôle, lors de l'apparition du signal « acquisition », de :

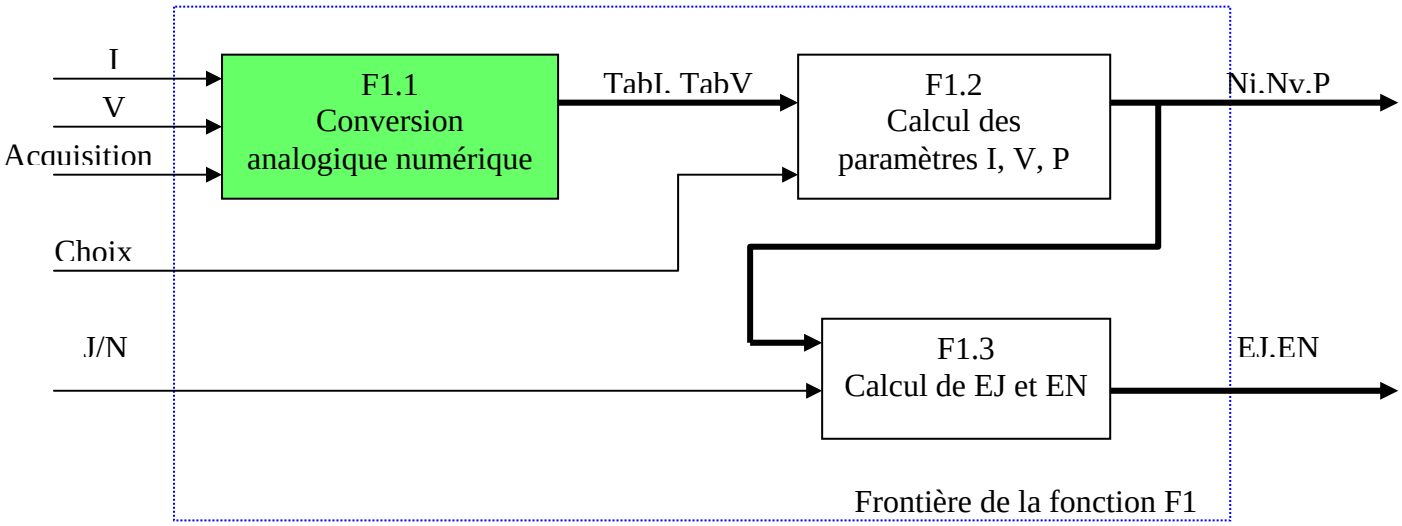
- calculer les valeurs efficaces de I ou de V si l'affichage a été demandé par l'utilisateur (choix = 0 ou 1)
- calculer la puissance efficace en W
- mettre à jour les compteurs EJ et EN.

Nota : à la mise sous tension les valeurs de EJ et EN sont celles sauvegardées en mémoire non volatile.

La description ci-dessus est répétée pour toutes les fonctions « Fi » qui composent l'objet technique.

Si les fonctions décrites précédemment sont trop complexes à réaliser alors on refait un shéma fonctionnel de niveau inférieur associé à la fonction « Fi ».

Exemple : Schéma fonctionnel associé à F1 :



Fonction F1.1 :

Entrées :
I, V, Acquisition : Idem que celles de F1.

Sorties :
TabI, TabV : tableaux de 64 mots de 8 bits rangés en mémoire Ram.

Rôle :

Sur réception d'un top d'acquisition, la fonction F1.1 échantillonne simultanément 64 fois les signaux I(t) et V(t) avec une période d'échantillonnage de 312,5 µs.

Cette description est effectuée pour toutes les fonctions Fi.j . Si les fonctions décrites précédemment sont trop complexes à réaliser alors on refait un shéma fonctionnel de niveau inférieur associé à la fonction « Fi.j » et ainsi de suite jusqu'à ce qu'on ait des fonctions simples à implémenter.

Directives pour l'écriture des programmes

Un programme applicatif est composé d'un programme principal faisant appel à d'autres programmes ou sous-programmes. Afin d'avoir une organisation structurée permettant de se retrouver plus facilement lors de la mise au point il est conseillé de :

- avoir une en-tête pour chaque programme (sous-programme, module ...).
- avoir un point d'entrée et un point de sortie
- avoir des programmes courts bien commentés
- utiliser des labels facilement identifiables.

1) l'en-tête du programme ou du sous-programme :

L'en-tête doit comprendre les informations suivantes :

- le nom de la fonction réalisée
- les paramètres d'entrée
- le rôle de la fonction
- les paramètres retournés par la fonction
- les registres utilisés
- les registres touchés
- parfois on indique aussi s'il est fait appel à la pile
- le nom du rédacteur et la date de la conception
- la version du logiciel utilisé ainsi que la version du système d'exploitation

Nota :
Les deux derniers points peuvent n'apparaître que dans l'en-tête du programme principal s'il n'y a qu'un auteur et si l'environnement de développement n'a pas évolué au cours du projet.

Exemple :

```
;***************** ****************************************
;                          FONCTION conv_7s
; entrées : registres 8 bits nx_h et nx_l
;cette fonction convertit nx_h et nx_l en un code pour affichage 7 segments
;résultats dans af_h et af_l
;registres utilisés : R22, R23, R24, R25
;registres touchés : R24, R25
;auteur :
;date :
;systèmes de développement : AVRStudio version 4.02, Windows 98 SE
;*********************************************************
```

2) Un point d'entrée, un point de sortie :

- éviter d'avoir dans le corps du sous-programme plusieurs points de sortie (on ne doit avoir qu'une seule instruction de « retour » du sous-programme et celle-ci doit se trouver à la fin.

```
conv_7s:
        out eear,nx_l     <============    Point  d'entrée
```

```
        sbi eecr,eere
        in nx_l,eedr
        out eear, nx_h
        sbi eecr,eere
        in nx_h,eedr
        mov af_h,nx_h
        mov af_l,nx_l
        ret             <=============== Point de sortie
```

3) Utilisation de labels identifiables

On peut utiliser des noms explicites pour les labels ou se servir de l'analyse fonctionnelle pour donner un nom aux différentes fonctions et étiquettes de branchement.

Exemple :

```
;***********************************************************
;                       Fonction F12
;.....en-tête de description ....
;
;***********************************************************
F12:
        subi nx_l,$0A
        brmi  F12_1
        inc nx_h
        rjmp F12
F12_1:
        ldi temp,$0A
        add nx_l,temp
        ret
```

La gestion des tâches

Une fois que le système technique est découpé en fonctions et sous-fonctions, il est nécessaire de concevoir un programme de gestion des tâches afin d'activer celles-ci aux moments opportuns. Pour des applications conséquentes, il est conseillé d'utiliser un logiciel industriel appelé « **moniteur multi-tâches temps réel ou RTOS** (Real Time Operating System)**»**. Celui-ci assure la synchronisation entre les tâches et peut également prendre en compte les problèmes de priorité lors d'éventuels conflits. Concernant les petites applications la méthode présentée ci-dessous est simple mais reste néanmoins suffisante.

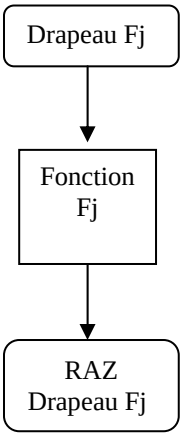
1) On associe à chaque fonction (ou sous-programme) un drapeau (bit de la mémoire RAM ou d'un registre). Ce drapeau, marqué à 1 par exemple, signifie que la tâche doit être exécutée).

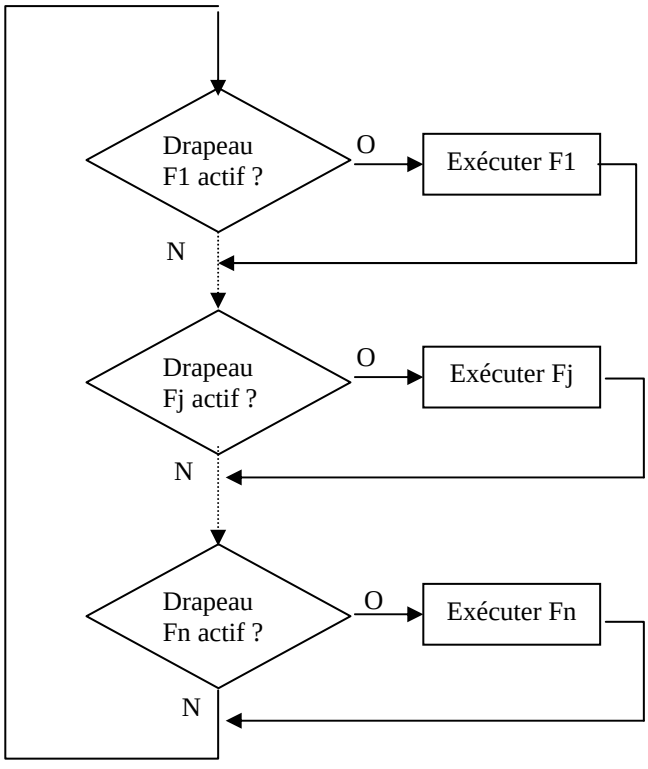
2) On crée un programme de gestion dont le rôle est de scruter séquentiellement les drapeaux associés aux différentes tâches et d'en lancer l'exécution si ceux-ci sont actifs. Une fois la tâche exécutée, le drapeau (bit) associé est remis à 0 (soit en fin de tâche, soit dans le programme de gestion).

La mise à 1 d'un drapeau se fait à partir :

- d'un programme d'interruption (horloge temps réel ou tout autre ressource du µC)
- d'un autre sous-programme.

Exemple sur le compteur d'énergie :





Gestionnaire des tâches