**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**Ministère de l'enseignement supérieur et de la recherche scientifique**

**École Nationale Polytechnique**

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

**DEPARTEMENT GENIE ELECTRIQUE**

**OPTION : AUTOMATIQUE**

Thèse de Magister

**Préparée par**

**Mr ALLOU FARID**

Ingénieur d'état en automatique (ENITA)

**Pour l'obtention du grade de Magister en automatique**

Thème :

# Extraction de données radar : Implantation sur carte DSP TMS320C40

Soutenue le,

**Devant le jury composé de :**

| | | | |
|---|---|---|---|
| **Président :** | **Mr Berkani D.** | **Professeur** | **ENP** |
| **Rapporteur :** | **Mr Belouchrani A.** | **Maître de conférence** | **ENP** |
| | **Mr Bellounar M.** | **Docteur d'état** | **CRD/CFDAT** |
| **Examinateur :** | **Mr Tadjine M.** | **Chargé de cours** | **ENP** |
| | **Mr Sadoun R.** | **Chargé de cours** | **ENP** |
| | **Mr Trabelsi M.** | **Chargé de cours** | **ENP** |
| | **Mr Hammadouche M.** | **Docteur d'état** | **CRD/CFDAT** |

**Année universitaire 2001/2002**

**REMERCIEMENTS**

Je tiens à remercier vivement mon directeur de thèse le Mr Belouchrani pour tous les conseils précieux qu'il m'a prodigués, pour sa patience et ses encouragements durant tout le projet.

Je tiens à remercier aussi Mr Bellounar mon co-directeur de thèse pour ces précieux conseils et l'intérêt qu'il a donné à ce travail.

Je remercie Mr Amoura pour son aide précieuse au projet.

Je remercie MM Magaz et Kemouche pour leurs aides et conseils le long de ce projet.

Je remercie le professeur D. BERKANI pour avoir accepté de présider ce jury et je remercie également tous les membres du jury, Mr TRABELSI, chargé de cours à l'ENP, Mr SADOUN, chargé de cours à l'ENP et le Docteur M.Hammadouche.

Que mes amis chercheurs du Centre de Recherche et Développement MM B. ALLAILOU, A. SENOUCI, K. MEBARKI, K. TINE et K. HADDACHE trouvent ici mes vifs remerciements.

# Sommaire

## Table des figures

# Chapitre 1

# Introduction Générale

## 1.1. Introduction aux systèmes de détection radar :

### 1.1.1. Historique :

Les premières applications de la radio - électricité furent les télécommunications, puis la radio - navigation. Le développement des moyens de transports, tant par mer que par air a fait rapidement apparaître le besoin de détecter simultanément la direction et la distance d'un objet non coopératif ( par exemple détection des icebergs ).

En effet, on ne pouvait, par les seuls moyens de la radio navigation, déterminer la position d'un objet, que si celui ci était muni d'un dispositif émetteur ; ce qui nécessitait pour le repérage de points géographiques la présence de balises, et pour le repérage des engins pilotés, la coopération d'un opérateur spécialisé. Ceci limitait les possibilités de la radio navigation face aux problèmes de l'anti - collision et de la détection des objectifs hostiles, ou simplement non coopératifs.

Le radar a été développé pour *« détecter »* la présence d'un objet sans la participation de l'objet lui même. Le mot R.A.D.A.R. qui est universellement adopté pour designer un matériel répondant à cette exigence est une abréviation de l'expression anglo-saxonne [1]: RADIO DETECTION AND RANGING

Les radars émettent des ondes radioélectriques, dont la longueur d'onde varie de quelques centimètres à environ 1 m. Les objets passant dans le faisceau réfléchissent ces ondes et les renvoient à l'émetteur. Les concepts de base du radar sont fondés sur les équations régissant les ondes électromagnétiques, formulées par le physicien britannique James *Clerk Maxwell* en 1864. Ces principes furent vérifiés en 1886 par les expériences du physicien allemand *Heinrich Hertz.* L'ingénieur allemand *Christian Hülsmeyer* fut le premier, en 1904, à suggérer l'utilisation d'échos radio dans un appareil de détection afin d'éviter les collisions en navigation. Un dispositif similaire fut proposé en 1922 par l'inventeur italien *Guglielmo Marconi.* Par la suite, le radar fut développé progressivement, grâce à l'action de nombreux savants, ingénieurs et techniciens [2].

Le radar est apparu simultanément entre 1930 et 1935 aux Etats-Unis, en Grande-Bretagne, en Allemagne et en France.

En France, les travaux de la C.S.F. sur le magnétron (1930) ont permis dès 1935 l'installation d'un Radar sur le paquebot NORMANDIE, ce dispositif ne constituait pas encore un Radar au sens « classique » du terme, mais plutôt un dispositif anti - collision basé sur des mesures de variation de phase en présence d'obstacles.

En 1940, la Grande-Bretagne, et aussi l'Allemagne avaient suffisamment développé la technique des impulsions pour mettre au point des systèmes opérationnels. La Grande Bretagne par exemple, possédait déjà en 1939 un réseau de déclenchement d'alerte presque continu, qui lui permettait de déceler la présence d'avions à 100 km de sa cote Est et Sud. Ces systèmes fonctionnaient sur une longueur d'onde d'une dizaine de mètres. Les antennes se

trouvaient sur des pylônes de 50 à 80 mètres de haut. A cause de la longueur d'onde utilisée, la directivité de ces antennes était très faible. Pour cette raison, la longueur d'onde fut ensuite ramenée aux environs de 1,5 mètre.

En 1940, les Allemands étaient parvenus à peu près au même point que les alliés. Par contre, ils n'arrivaient pratiquement pas à maîtriser des longueurs d'ondes inférieures à 0,5 mètre; aussi furent-ils surclassés en 1945 par les dispositifs développés par les alliés et qui, grâce aux applications du magnétron, fonctionnaient sur des longueurs d'ondes de 10 cm, ce qui leur permettait l'utilisation de dispositifs plus précis, car plus directifs.

Signalons en outre, le développement des radars de bord qui firent leur apparition dès la bataille d'Angleterre ; ils fonctionnaient alors sur des longueurs d'ondes de 30 cm à 1 m. L'abaissement de cette longueur d'onde jusqu'à 1 cm environ en fit un instrument très précis. Des résultats sérieux avaient parallèlement été enregistrés sur les radars de DCA et radars de tir en général.

Par ailleurs des prototypes de radars à impulsions fonctionnant en ondes métriques ont permis d'obtenir dès 1939 des portées supérieures à 100 Km sur aéronef, alors que les travaux de la CSF sur le magnétron avaient abouti à des performances remarquables pour l'époque en ondes décimétriques. Ces débuts prometteurs furent malheureusement interrompus par la guerre.

La technique radar évolua de manière très profonde entre 1940 et 1945, au point qu'une série de volumes édités par MIT (Massachussetts Institute of Technology) a pendant longtemps servi de document de base pour ce domaine [1].

Les développements du radar sont considérables; citons:

- Les radars de couverture plane
- Les radars volumétriques
- Les radars de surveillance secondaires
- Les radars de navigation
- Les radars de tir
- Les radars d'interception
- Les radars Doppler
- Les radars d'atterrissage
- Les radars de surveillance de sol
- Les radars météorologiques
- Les radars de poursuite et de trajectographie.

### 1.1.2. Principe de base :

Les premières expériences exécutées sur la détection des objets non coopératifs furent basées sur le fait que la présence d'un objet dans le champ de rayonnement d'une antenne perturbait ce rayonnement, ce qui provoquait une variation des paramètres radioélectriques de celle-ci. La mesure était faite par exemple en utilisant une antenne d'émission, et une antenne de réception sur laquelle étaient mesurées des variations d'impédance lorsqu'un obstacle extérieur venait modifier le couplage entre les deux antennes. Ce procédé qui permet de déterminer l'existence d'un objet est insuffisant car il ne permet pas de localiser cet objet. Il faut

donc compléter cette technique par une mesure permettant la localisation, c'est ce que peut être réalisé en utilisant des impulsions selon le principe illustré dans la Figure 1.1:

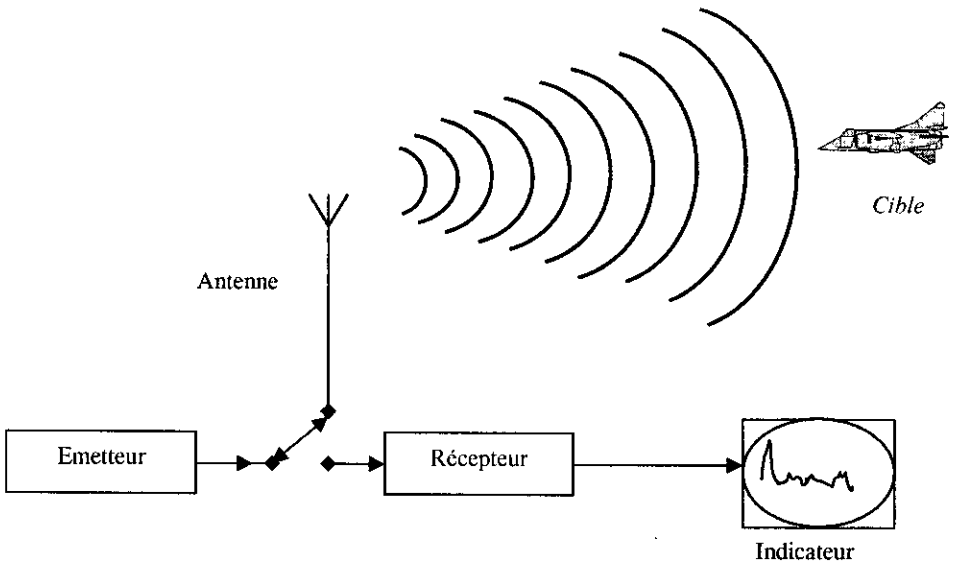



**Figure 1.1: Principe de détection radar**

Soit un signal très bref de durée ζ égale à quelques micro - secondes, engendré dans un émetteur, et dirigé vers une antenne omnidirectionnelle. L'onde ainsi excitée se déplace dans l'atmosphère à la vitesse de la lumière: $C=3.10^8$ m/sec.

Ce signal se propage dans toutes les directions et se trouve à un instant donné $T$ réparti entre deux sphères de rayon $C.T$ et $C(T+\zeta)$. En d'autres termes il y a formation d'une " onde sphérique". Au bout de l'intervalle de temps $\Delta T_1$, le signal atteint la cible; ce temps est proportionnel à la distance antenne - cible $D$ et s'en déduit par la relation:

$$\Delta T_1 = \frac{D}{C}$$

Une partie du signal est réfléchie par la cible; on dit quelquefois que la cible est "illuminée" et "re-rayonne" une partie de l'énergie de manière également omnidirectionnelle. Au bout d'un intervalle de temps $\Delta T_2$ égal à $\Delta T_1$ l'onde réfléchie atteint à nouveau l'antenne qui capte une partie de l'énergie réfléchie par la cible.

Cette antenne a été reliée, à un récepteur très sensible qui amplifie le signal capté et fournit donc une impulsion analogique comme réponse à l'impulsion émise. L'intervalle de temps qui sépare l'instant d'émission de l'instant de réception est alors:

$$\Delta T = \Delta T_1 + \Delta T_2 = 2.\frac{D}{C}$$

La mesure de ΔT permet de connaître la distance à laquelle se trouve la cible. On peut mesurer ce temps de la manière suivante; un tube cathodique analogue à ceux utilisés sur les oscilloscopes classiques appelé "Scope A" est utilisé à cet effet. Sur les plaques de déviation

horizontale, on applique un signal en dents de scie dont le début est synchronisé avec l'impulsion d'émission. Sur les plaques de déviation verticale, on applique le signal reçu par le récepteur, convenablement amplifié. Ainsi, la position horizontale du spot sur le Scope est proportionnelle au temps écoulé après l'émission et une déviation verticale est le signe de présence d'une cible. On peut graduer directement l'axe horizontal du Scope en distance et on possède alors un appareil révélant la présence d'une cible et donnant sa distance. De tels dispositifs peuvent être utilisés dans la construction "d'un radar de barrière. La Figure 1.2 résume la suite des signaux engendrés et perçus par ce type de radar [1].

### 1.1.3. Composition d'un radar:

Le radar est un instrument capable de fournir à l'utilisateur les informations suivantes :

- Existence d'un corps étranger dans l'atmosphère (avion, bateau, nuage, obstacle naturel, etc. ...).
- Position du corps étranger.

Ces deux informations peuvent être complétées suivant les cas, par d'autres portant sur la vitesse, l'étendue, voir la nature du corps détecté par le radar. Le diagramme fonctionnel type d'un radar peut être représenté par le schéma de la Figure 1.3:

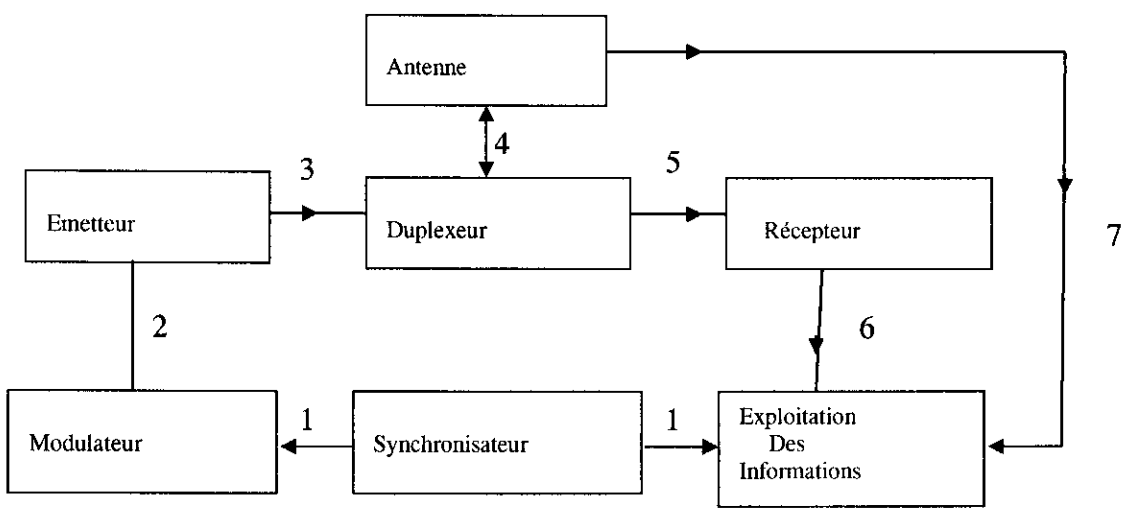



**Figure 1.3 : Diagramme fonctionnel type d'un radar**

1. signaux de synchronisation
2. signaux de commande émission
3. signaux hyperfréquence émis
4. liaison duplexeur Antenne
5. signaux hyperfréquence reçus
6. signaux traités par le récepteur
7. informations angulaires de " pointé ".

## 1.2. Les radars de surveillance secondaires:

### 1.2.1. Introduction :

L'identification des aéronefs s'est présentée très tôt comme une nécessité aux personnes chargées de la surveillance du domaine aérien. Ce besoin est évident pour les applications militaires, où tout organisme chargé de la surveillance d'un territoire doit pouvoir déterminer l'identité des aéronefs qui y pénètrent. Mais c'est aussi une aide commode pour la gestion du trafic aérien civil. Des techniques ont donc été développées pour permettre cette identification. Celles ayant conduit aux applications le plus largement diffusées ont trait au système question- réponse, et ont pris le nom de **radar secondaire** [composante de l'ATC (Air Trafic Control)] en application civil, et d'IFF (Identification Friend or Foe, identification ami ou ennemi) en application militaire. Un même système concerne, en fait, ces deux applications, avec simplement quelques particularités au niveau des modulations utilisées et des réalisations des équipements correspondants [3].

### 1.2.2. Historique :

L'apparition du radar vers la fin des années trente, permettant de détecter des aéronefs au delà de ce que pouvait déceler l'œil ou l'oreille humaine, rendit caduque l'identification a vue utilisée jusque-là.

C'est donc a cette période que remontent les premières études d'un système d'identification dont les performances sont en harmonie avec celles des radars. C'est ainsi que des équipes travaillèrent, simultanément, sur des concepts voisins en Grande-Bretagne, aux États-Unis et en Allemagne. Les premiers concepts visent à moduler l'écho radar renvoyé par l'avion au moyen de dipôles ou d'éléments réfléchissants commutes par relais (travaux de Watson-Watt). Mais l'altitude de l'avion par rapport au radar pouvait sérieusement affecter l'efficacité de ce système. On pensa alors à rendre l'avion actif en lui faisant renvoyer un signal détectable par le radar, donc dans la même bande de fréquence. Ainsi, apparurent en Grande-Bretagne en 1939 les premiers répondeurs MK I (Mark I), fonctionnant dans la bande 20 à 30 MHz des radars Home Chain. Leur emploi fut vital pendant la bataille d'Angleterre.

La seconde guerre mondiale donna un coup de fouet aux radars qui se développent vers des fréquences de plus en plus élevées pour gagner en précision angulaire. Il ne fût plus possible de concevoir des répondeurs efficaces sur toutes les fréquences radars susceptibles d'être utilisées. On en vint alors à affecter une fréquence particulière à la fonction d'identification, tant pour l'interrogation que pour la réponse (répondeurs MK III dans la bande 157-187 MHz). Le radar doit alors être double d'un équipement susceptible d'émettre un signal d'interrogation, et de recevoir la réponse. On arrive ainsi au radar secondaire dont une équipe conjointe États-Unis/Grande-Bretagne fige dès 1945 les grandes lignes (système MK V dans la bande 950-1150 MHz) encore valable a nos jours. La guerre avait vu la mise en place de

plusieurs étapes avant ce concept, tant du côté des alliés que du côté allemand. L'après guerre voit essentiellement s'affiner les techniques de codage, afin de rendre l'identification de plus en plus sûre (MK X). En 1954, le secret militaire qui couvrait ces travaux fut levé, ce qui en permit l'usage civil aux Etats-Unis sous l'appellation ATCRBS (Air Traffic Control Radar Beacon System), pendant que se poursuivait l'évolution du concept militaire par l'application des techniques de chiffrement (MK XII ). Ces systèmes sont ceux utilisés de nos jours, l'évolution se faisant militaires, et vers l'adressage des interrogations pour les systèmes civils.

### 1.2.3. Principe de fonctionnement :

Dans les systèmes actuels, le processus d'identification est donc initialisé par un interrogateur (généralement associé à un radar primaire, mais qui peut être autonome) qui génère un mot d'interrogation. Celui-ci est rayonné par une antenne directive orientée vers l'aéronef à identifier voir la figure.1.4. L'identification devant être obtenue dans tout l'espace aérien entourant le radar, on donne au diagramme de rayonnement de l'antenne une forme d'éventail avec une bonne directivité en azimut et un faisceau large site. L'antenne est animée d'un mouvement de rotation continu autour d'un axe vertical. La totalité de l'espace aérien est donc balayée en un tour d'antenne. Lorsque au cours de la rotation, l'antenne se trouve pointée vers un aéronef ami (ou plus généralement coopératif), le répondeur de ce dernier reçoit le mot d'interrogation et y répond en renvoyant un message de réponse caractéristique de son identité. Ce message est reçu par l'interrogateur, qui le décode et en extrait l'identité de l'aéronef. La position de l'aéronef correspondant est simultanément connue en coordonnées polaires : la distance par mesure du temps aller-retour nécessaire lors de l'échange de signaux, et l'azimut par la direction dans laquelle était pointée l'antenne de l'interrogateur lors de l'identification. L'interrogateur obtient donc une vue panoramique de l'environnement, généralement figuré en superposition sur l'écran panoramique circulaire du radar primaire.

#### 1.2.3.1. Signaux d'interrogation :

C'est un signal hyperfréquence modulé en impulsions. La fréquence adoptée est de 1030 ± 0.2 MHz, la modulation consistant en une paire d'impulsions, appelées P1 et P3 dont l'espacement définit le mode d'interrogation. La durée des impulsions est de 0.8± 0.1 ps avec des temps de montée compris entre 50 et 500 ns, des temps de descente compris entre 50 et 200 ns (cette raideur des flancs est limitée en valeur maximale pour avoir une bonne définition temporelle des impulsions, et en valeur minimale pour limiter l'occupation spectrale du signal émis).

Pour les applications civile du radar secondaire, il y a 4 modes d'interrogation, désignés par A, B, C et D, aux quels correspondent les espacements suivants de la paire d'impulsion voir figure.1.5.

| Mode d'interrogation | Espacement entre PI et P3 en ps | Utilité |
|---|---|---|
| A | 8 ± 0.1 | Identification |
| B | 17± 0.1 | Identification (GB) |
| C | 21+ 0 1 | Altitude |
| | 25± 0 1 | Non utilisé |

Les applications militaires ont 3 modes conventionnels non sécurisés. Désignés par 1, 2 et 3, ils sont caractérisés par les espacements suivants :

| Mode d'interrogation | Espacement entre P1 et P3 en ps | Utilité |
|---|---|---|
| 1 | 3 ± 0.1 | Varie d'une force aérienne à une autre |
| 2 | 5± 0.1 | |
| 3 | 8± 0.1 | |

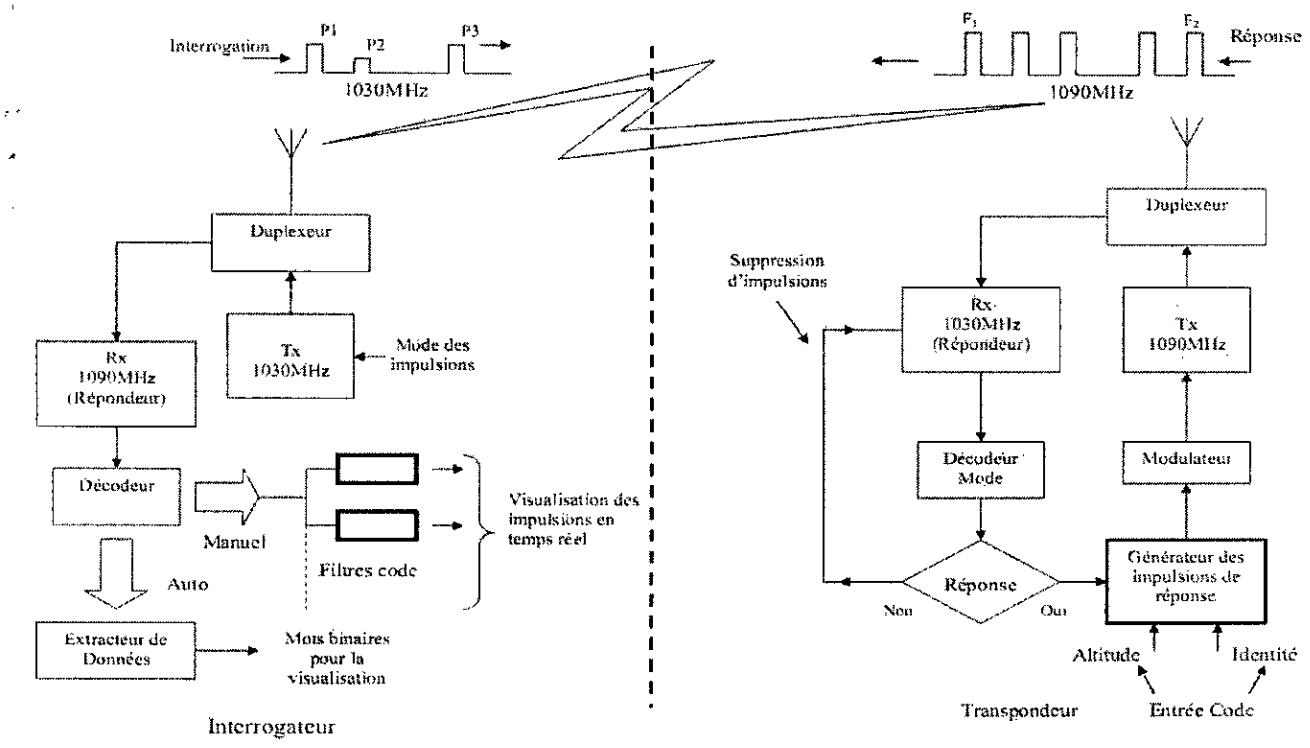



**Figure 1.4: Schéma bloc du SSR**

### 1.2.3.2. Signaux de réponse:

C'est un signal hyperfréquence modulé en impulsions. La fréquence adoptée est de 1090 ± 3 MHz, la modulation comporte toujours une paire d'impulsions dites d'encadrement, appelées FI et F2 espacées de 20.3ps. Entre ces impulsions existent 13 positions possibles d'impulsions au pas de 1.45ps. La position centrale appelée X n'est pas utilisée. Ainsi les 12 positions permettent de coder un mot de réponse à 12 bits (présence d'impulsion c à d 1, absence c à d 0) soit 4096 réponses différentes voir figure.1.5.

### 1.2.3.3. Description des principaux Modes :

**Mode 1:**

Ce mode, à usage militaire, permet l'identification tactique ami/ennemi d'un aéronef, d'une unité aérienne ou d'un dispositif.

**Mode 2 :**

Ce mode est aussi à usage militaire seulement, il est utilisé une identification plus détaillée de l'aéronef.

**Mode 31A:**

C'est un mode à utilisation militaire/civil, permet le contrôle de tout les aéronefs qui l'environnent.

**Mode C:**

Ce mode est utilisé par l'aviation militaire et civile en même temps. Il est destiné à fournir le niveau de vol (altitude) des aéronefs à chaque interrogation. Le code de réponse n'est donc pas affiché par le pilote mais fourni par un *alticodeur* (capsule barométrique).

Le contenu de la réponse voir la figure.1.5 est limité aux impulsions Al, A2, A4, B1, B2, B4, Cl, C2, **C4,** D2 et D4, qui permettent de coder une altitude de -1000 à 126700 pieds (1 pied = 30,48 cm).

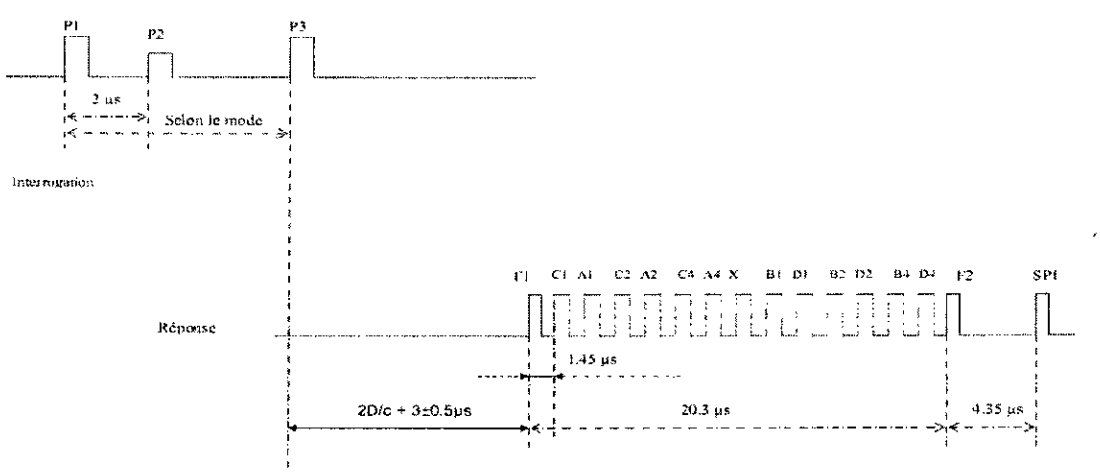



**Figure 1.5: Signaux Echangés entre l'interrogateur et le transpondeur**

## 1.3. Cahier des Charges :

### 1.3.1. Introduction :

Dans ce chapitre nous allons présenter les actions à entreprendre pour la réalisation de ce projet pour les différentes parties du système, en l'occurrence le hardware, le software, comme nous le montre la figure .1.6 ; le schéma synoptique général du nouveau système montre la partie radar et la partie extracteur de plots ; la carte d'interface, la carte DSP de l'IFF et le système de visualisation. Les signaux qui sont nécessaires au fonctionnement qui sont les signaux principaux de synchronisation ainsi que le signal commande de mise en marche de l'émetteur récepteur de l'interrogateur au moment voulu avec la souris.

### 1.3.2. Hardware :

Cette partie qui est la première dans le projet consiste à l'étude du système existant qui est la station radar ASRx et le choix de la solution hardware pour adapter cette station radar au système d'extraction de plots numérique. Les tâches à entreprendre sont les suivantes :

- L'étude détaillée du fonctionnement de la station radar qui comprend l'émetteur, le récepteur, le déchiffreur, les indicateurs panoramiques et principalement le bloc de synchronisation qui représente le cœur de la station.
- Identification des signaux nécessaires ainsi que le meilleur emplacement pour un éventuel piquage de ces signaux pour les utiliser dans le système d'extraction de plots.
- Interfaçage des signaux pour le conditionnement ; le filtrage, adaptation et la séparation galvanique s'il y a lieu pour quelque signaux.
- Le choix de la carte DSP qui va répondre aux exigences matérielles et la programmation temps réel.

### 1.3.3. Software :

Au niveau du software deux parties importantes seront à étudier :

- La première partie comprend le choix de l'extracteur de plots secondaires à implanter dans le DSP avec les exigences temps réel. Cette partie du software sera écrite en assembleur et le compilateur C de Texas Instruments.
- Les résultats de l'extraction au niveau du DSP seront communiqués vers une unité visualisation et d'exploitation ; la visualisation aura pour fonction la visualisation des cibles amies interrogées par la station ainsi que leurs coordonnées, pour l'exploitation ; sa fonction reste la prise en charge de ces cibles, (interrogation automatique) l'enregistrement des situations aériennes.

Ce programme représente une application écrite avec un langage évolué qui ne demande pas un travail temps réel sauf dans la récupération des données envoyées par la carte DSP à qui il faut donner la priorité pour éviter la perte d'information ou une erreur de protocole qui peut provoquer un blocage de l'unité de visualisation, cette application se

déroule sur un Pentium II 200MHz avec 64 Motets de RAM monté sur une carte mère avec bus VME et l'application sera écrite avec le Borland C++Builder version 4.

Les bancs d'essai comprennent l'ensemble des programmes soit écrits en assembleur c'est à dire à charger sur dans la carte DSP ou avec le langage évolué Borland C++Builder version 4 qui se déroulent dans le carte Pentium II pour le test des points suivant :

- L'arrivée des signaux de synchronisation (PRFs, CPIs et Nord) vers la carte DSP TMS320C40 sans impulsions parasitaires.
- Le bon déroulement du programme d'extraction de plots dont les coordonnées des cibles soient hardware ; avec un signal synthétique, simulateur au niveau de la station ou par une interruption interne au DSP (calcul de la distance et de l'azimut) et la conversion des coordonnées polaires vers les coordonnées cartésiennes.
- La récupération des données de la carte DSP TMS320C40 sans erreur et sans perte d'information.
- L'envoie des données de la visualisation vers la carte DSP TMS320C40.
- Blocage du balayage du programme de visualisation.

## 1.4. Le but de cette thèse :

Le travail présenté dans cette thèse rentre dans le cadre d'un projet de modernisation de la station radar ASRx. Ce travail a été réalisé au centre de recherche et développement Réghaïa. Cette modernisation a pour but d'automatiser toute la chaîne de traitement radar en l'occurrence le traitement du signal numérique, visualisation numérique et la communication série des données radar soit par ligne spécialisée ou ligne téléphonique normale. Le système d'identification amie ou ennemi existant est analogique, le signal reçu est déchiffré puis envoyé directement vers un indicateur panoramique à rotation mécanique. L'intégration des plots des cibles amies se fait à l'œil nu, ainsi que la détermination de leurs coordonnées qui se fait manuellement et qui présent essentiellement un problème de transfert des données vers un centre de décision lointain.

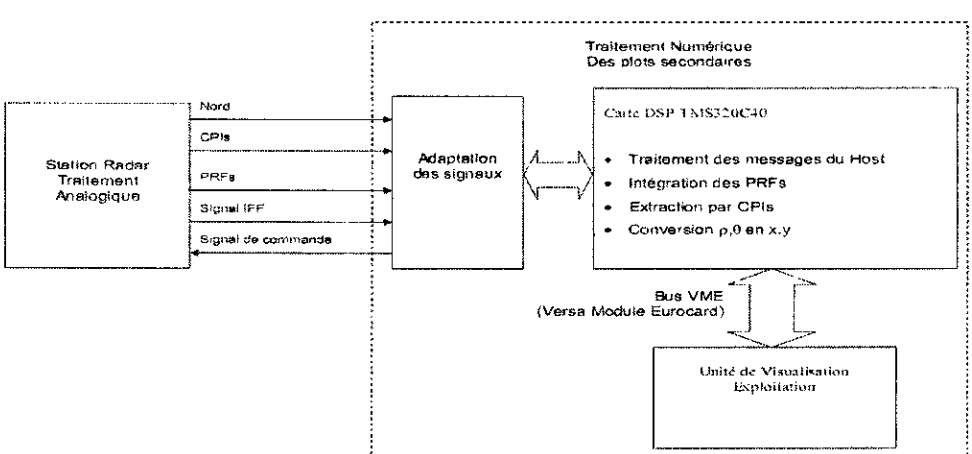



**Figure 1.6: Schéma synoptique général du système modifié**

# Chapitre 2

# Etude de la station radar utilisée

## 2.1. Présentation de la station radar:

la station radar utilisée dans ce projet est une station radar de surveillance bidimensionnel à balayage mécanique dont la portée pratique en distance dépasse les 300 Km et inférieure à la portée théorique qui est de 500 Km pour une PRF de 300 Hz en appliquant la formule (2.1) suivante [5] :

$$\text{Portée en Distance} = (1/\text{PRF}) \times C/2 \qquad (2.1)$$

$C = 3 \times 10^8$ est la vitesse de la lumière.

Cette station comprend un radar primaire et un radar secondaire. Le radar primaire est un radar à plusieurs canaux superposés. Le radar secondaire qui est solidaire au radar primaire car les signaux de synchronisation de l'émetteur et le récepteur du primaire sont les même que pour le radar secondaire, les aériennes pour la formation du lobe secondaire utilise l'aérienne supérieur du radar primaire et enfin la visualisation se fait sur les mêmes indicateurs panoramiques. La station est équipé d'un système pour déporter par câble l'image radar sur une portée ne dépassant pas les 300 m, et aussi un déport par radio dont la portée théorique est d'une dizaine de kilomètres mais qui pose beaucoup de problème pour le faire fonctionné, ce qui présente un problème au niveau du déport de l'information soit cible amie ou ennemie vers le poste de prise de décision ou de contrôle de la circulation aérienne qui se trouve très loin de la station radar.

## 2.2. Interrogateur:

Le radar ASRx est équipé d'un système IFF qui fait que la cible amie soit représentée sur l'indicateur par un spot sous forme d'une banane à côté de l'écho primaire si ce dernier existe, et les deux échos primaire et secondaire apparaissent sur le même azimut alors que en distance l'écho secondaire est un peu plus loin de l'écho primaire pour éviter la confusion dans l'affichage de ses deux échos sur l'indicateur monochrome [6]. L'interrogateur de la station radar ASRx, indiqué dans la Figure.2.1, est constitué d'un duplexeur, d'un émetteur, d'un récepteur, d'un déchiffreur, d'un bloc de corrélation et d'un simulateur des signaux de code. Sa portée est supérieure à la portée du radar primaire. Les angles limites; en gisement et en site sont de 0° à 360° et de 0.75° à l'angle maximum; respectivement. Le pouvoir séparateur en distance est de 1500 mètres. L'affichage de la banane de l'avion ami peut se faire de deux manières ; soit avec en corrélation avec l'écho de la cible elle même ce qui veut dire que s'il n'y a pas de détection alors il n'y aura pas de réponse de la cible aussi même si cette dernière a le code exacte, alors la présence de l'écho radar primaire de la cible est nécessite. Cependant, la deuxième manière est sans corrélation avec l'écho primaire, généralement ce cas de figure est très souvent utilisé dans l'opérationnel en raison de mauvaise détection ou si la cible à interroger est très éloignée. Le signal d'identification qui est la réponse reçue passe à travers le déchiffreur

qui génère à sa sortie une impulsion si la réponse correspond exactement au code qui se trouve au niveau du déchiffreur. Cette impulsion subis un retard de 4 ps pour ne pas écraser l'écho primaire au niveau de l'affichage. Ces 4ps est l'équivalent de 600 m en appliquant la même formule (2.1), ce retard est provoqué au niveau du bloc de corrélation en utilisant une ligne à retard.

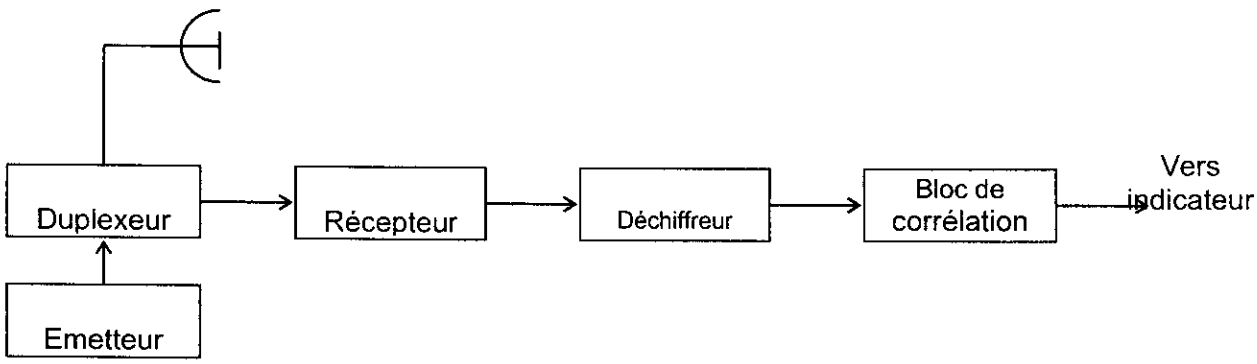



**Figure 2.1: Schéma synoptique de l'interrogateur du système IFF**

## 2.3. Composition :

### 2.3.1. Emetteur

L'émetteur, indiqué dans la Figure.2.2, se compose d'un pré-modulateur, d'un modulateur et d'un oscillateur HF (Haute Fréquence). Le pré-modulateur génère des impulsions nécessaires pour le déclenchement du modulateur. Le signal à la sortie de l'émetteur est un signal impulsionnel de haute tension, ayant pour porteuse une fréquence UHF (Ultra high-frequency) dans la gamme des ondes décimétriques de 750MHz *(2.= f le :* la fréquence et la longueur d'onde sont inversement proportionnelles dont c est la vitesse de la lumière.) comme illustré sur la Figure.2.3. La PRF du radar secondaire est la même que celle du radar primaire vu le fait que le signal de synchronisation de toute la station radar est l même dans le cas où l'affichage des échos soient primaires ou secondaires se fait sur le même indicateur panoramique qui nécessite un seul signal de référence pour le radar primaire que secondaire. La figure 2.3 montre les signaux de sortie des différents étages de l'émetteur.

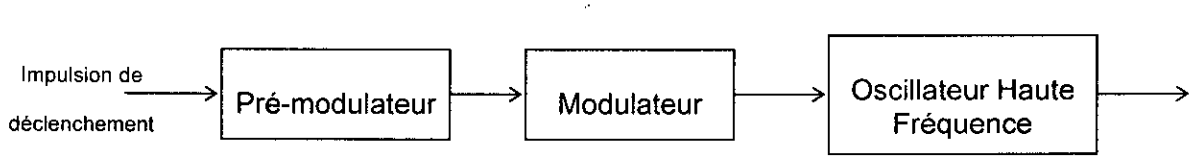



**Figure 2.2: Schéma synoptique de l'émetteur**

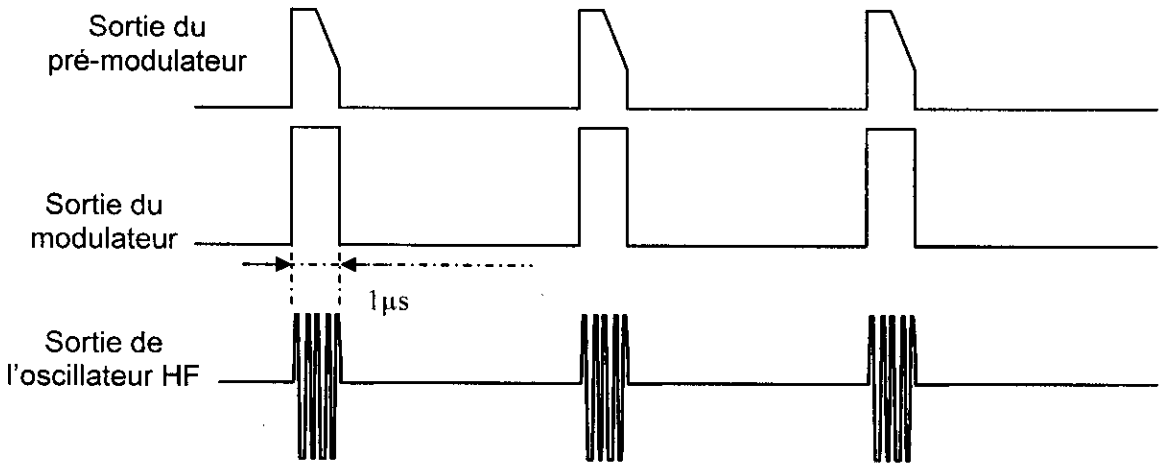



**Figure 2.3: Les signaux de sortie des différents étages de l'émetteur**

### 2.3.2. Récepteur

Le récepteur, indiqué dans la Figure.2.4, se compose d'un AHF, Amplificateur à Haute Fréquence, d'un mélangeur et d'un oscillateur local pour la conversion du signal reçu en signal à Fréquence Intermédiaire (FI), d'un amplificateur à fréquence intermédiaire, et d'un détecteur d'enveloppe pour convertir le signal FI en *signal vidéo. La* sensibilité du *récepteur est* de *2 x* $10^{-}$ *w. La bande* passante *est* de 27 ± 3 MHz à -3 dB avec une fréquence centrale (fréquence intermédiaire) de 60 MHz.

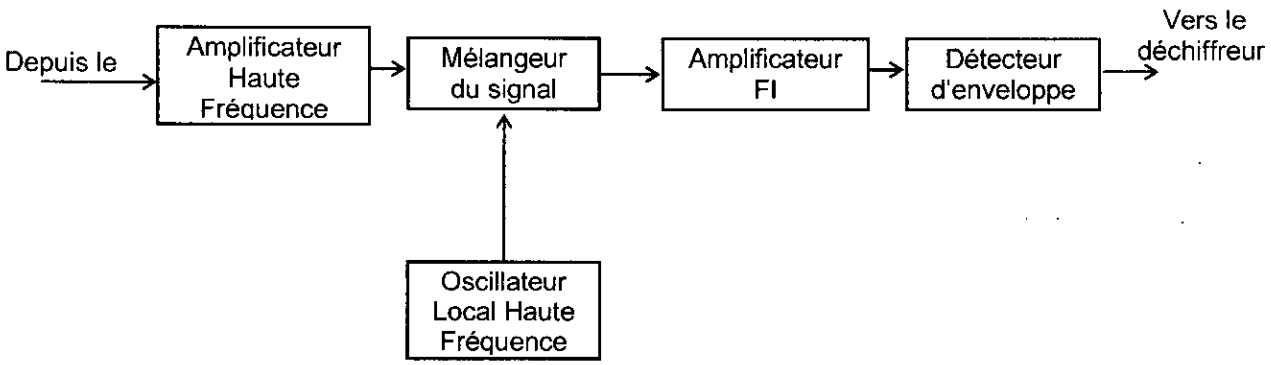



**Figure 2.4: Schéma synoptique du récepteur**

### 2.3.3. Déchiffreur

Le déchiffreur, indiqué dans la Figure.2.5, a pour rôle l'extraction de la fréquence code qui se trouve à l'intérieur des impulsions renvoyées par l'aéronef. Ces fréquences codes dont les fréquences sont gardées secrètes et au nombre 32 mais qu'on peut décodées avec un analyseur de spectre de 1 GHz pour avoir deux pics ; le plus petit va être la fréquence code et le deuxième pics ne peut être que la porteuse de 750 MHz. Le déchiffreur se compose d'un amplificateur vidéo à large bande

passante, d'un limiteur, d'un filtre à bande passante étroite de code amovible, d'un détecteur d'enveloppe, d'un relais électronique, et d'un simulateur intérieur du déchiffreur. L'amplificateur vidéo à large bande passante amplifie les signaux acheminés de la sortie du récepteur qui sont des impulsions vidéos modulées en amplitude dont la durée est de l'ordre de 3 ps comme illustré dans la Figure.2.5(a). Les signaux à la sortie de l'amplificateur vidéo sont limités en amplitude pour le filtre de code amovible à bande passante étroite qui ne laisse passer que les signaux dont la fréquence de modulation est comprise dans la bande passante du filtre de code pour donner les signaux illustrés dans la Figure.2.5(b) qui passent par le détecteur d'enveloppe. A la sortie du détecteur d'enveloppe nous avons une impulsion indiquée dans la Figure 2.5(c) qui excite le relais électronique et qui à son tour forme une impulsion vidéo de 20 V et d'une durée de 3 ps comme illustrée dans la Figure 2.5(d).

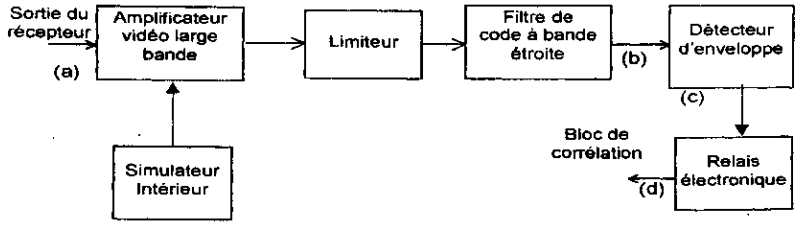



Figure 2.5: Schéma synoptique du déchiffreur

Le simulateur intérieur est un bloc qui génère des impulsions avec la fréquence code puisqu'un même code est placer dans ce bloc de simulation. Alors le simulateur fournit à l'entrée du déchiffreur les signaux dont la fréquence de modulation se place dans la bande passante du filtre de code N° x (parmi les filtres de code) pour contrôler le fonctionnement du déchiffreur et le réglage du seuil pour le déclenchement du relais électronique.

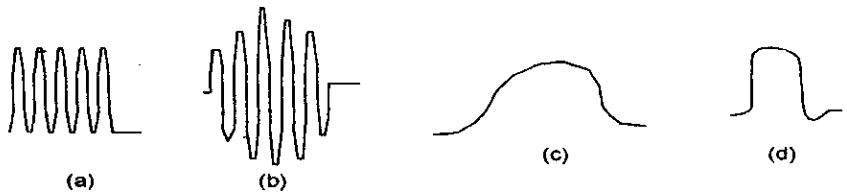



Figure 2.6: Chronogramme des sorties des différentes parties du déchiffreur

### 2.3.4. Bloc de corrélation :

Le bloc de corrélation accomplit les fonctions suivantes:

• Retarder les impulsions à la sortie du déchiffreur par rapport à l'écho primaire.

• Élargissement des ces impulsions pour avoir la forme de banane affichée sur les indicateurs panoramique du radar; le bloc de corrélation reçoit à son entrée le signal d'identification (IFF) comme le montre la figure.2.7; la durée des impulsions à l'entrée est 3 ps, à la sortie la durée des impulsion est de 5ps. L'amplitude du signal peut être ajusté entre 5 et 2OVolts avec un potentiomètre qui se trouve dans le bloc de corrélation.

• Corrélation de l'écho secondaire avec l'écho primaire; l'écho secondaire est affiché si l'écho primaire est présent.

• Corrélation des impulsions dans le cas de détresse et générer une impulsion de durée 20ps.

### 2.3.5. Bloc de synchronisation :

Le bloc de synchronisation est le cœur du système radar. Il délivre les signaux de base qui définissent les instants d'émission, et divers signaux annexes nécessaires à des opérations en temps réel. Son élément de base est une horloge de très grande stabilité ($10^{4}$ à $10^{-5}$) à partir de laquelle sont engendrés les signaux de synchronisation. Ces signaux sont distribués aux différents éléments à piloter. Leur distribution doit être assurée avec une grande reproductibilité sur chaque voie, de manière à ne pas fausser les mesures effectuées.

#### 2.3.4.1. Composition :

Le bloc de synchronisation est constitué des cartes suivantes :

- Carte de formation des signaux de synchronisation PRFs (Pulse Repetition Frequency) : cette carte génère les signaux de synchronisation en fonction du régime de travail choisi (changement de fréquence, contre mesure électronique), ces signaux sont les suivants [5] [6]:

***PRF-1:*** c'est le signal de synchronisation principal. Ce signal commande les émetteurs récepteurs des **4** étages du radar primaire et l'émetteur récepteur du radar secondaire. La durée du signal est de 1.5 à 2.5 ps et la période de répétition est de 2.5 à 4 ms (400 à 250 Hz) comme le montre la figure.2.8. L'amplitude est de 5V.

***PRF-2:*** retardé de quelques microsecondes par rapport au signal PRF-1. Il est utilisé dans le déport de la situation radar vers la salle des opérations.

***PRF-3*** *:* ce signal est aussi retardé par rapport au PRF-1 et il va vers les indicateurs panoramiques pour la visualisation.

• . Carte de formation du signal nord et ACP : cette carte forme les signaux suivants :

***Signal Nord:*** le signal nord a pour rôle de donner la référence zéro par rapport au nord

magnétique. Ce signal est impulsionnel; la durée de l'impulsion est de 9ms, la période est de 10 à 20s (10 s pour une vitesse de rotation de 6 tr/mn, 20 s pour une vitesse de rotation de 3 tr/mn) et l'amplitude est de 5 V comme le montre la figure.2.9.

***Signal ACP(azimut pulse change):*** le nombre d'impulsions générées par tour d'antenne est de 4096; la position de l'antenne du radar est codée sur un mot 12-bit (voir la figure.2.10).

Les signaux Nord et ACPs sont obtenus à partir de signaux lumineux délivrés par des disques tournants entraînés par des selsyns (un selsyn récepteur entraîne les disques, reçoit une tension envoyée par un selsyn transmetteur entraîné par l'antenne, et cette tension traduit l'angle de rotation de l'antenne) synchronisés sur la rotation de l'antenne. Ces disques comportent des fentes, à travers lesquelles passe un rayon lumineux qui frappe une cellule photoélectrique, celle-ci délivre des impulsions qui vont déclencher des bascules permettant d'obtenir l'illumination de l'indicateur pendant toute la durée de balayage. Le disque Nord contient une fente qui correspond avec le passage par le nord. Le deuxième disque comprend 64 fentes plus un système réducteur de 1:64, ainsi pour un tour d'antenne on aura 64*64 (4096) ACPs.

- *Carte de formation des marques de gisement :* cette carte génère les marques de gisement 30° et 5° de l'indicateur panoramique à partir des signaux ACPs et Nord.
- *Carte de formation des marques distance : cette carte génère les marques distance 10km* à partir du signal PRF-3 de l'indicateur panoramique.
- Carte de suppression des impulsions asynchrones : cette carte a pour fonction de filtrer le signal d'identification (IFF) des impulsions asynchrones du au rayonnement d'autre radars.
- Carte d'amplification : cette carte amplifie les signal PRF-1, PRF-2 et PRF-3 du niveau 5V à 30V pour la commande des émetteurs récepteurs des radars primaire et secondaire, le déport et les indicateurs panoramiques.

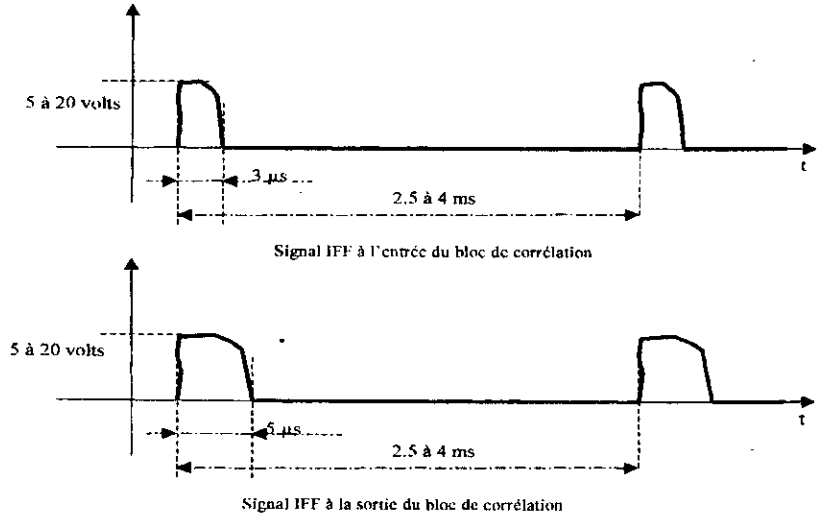



**Figure 2.7 : La forme du signal IFF**

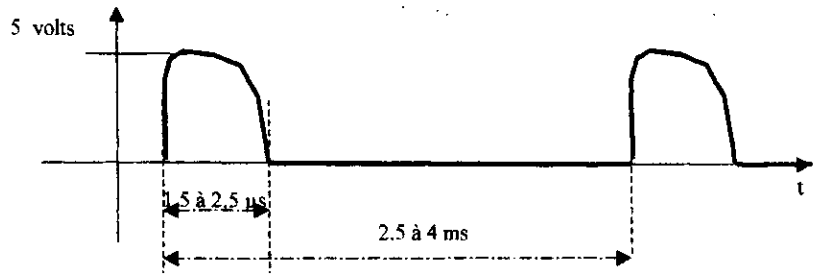



Figure 2.8: Signal PRFs

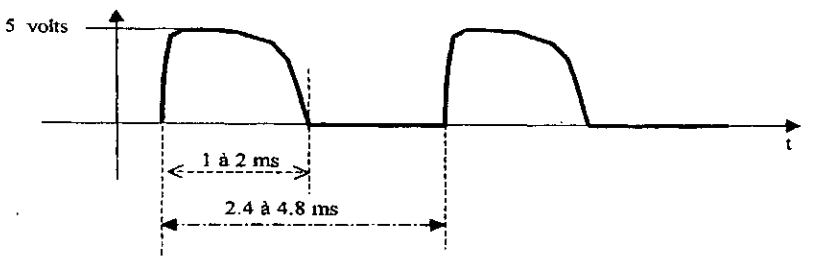



Figure 2.9: Signal ACPs

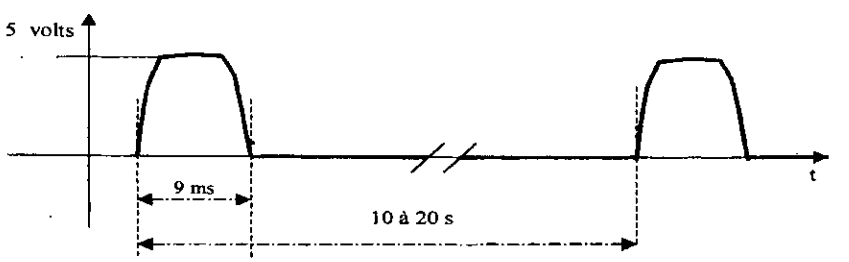



Figure 2.10: Signal Nord

## 2.4. Conclusion

Après l'étude de la station radars utilisée les signaux de synchronisation principaux pour le fonctionnement de notre système d'extraction de plots radar secondaire sont comme suit :

1. Le signal PRF-1 commande les émetteurs récepteurs de la station radar, et il sera utilisé comme signal de référence (t-zéro) en distance. Le signal sera utilisé pour remettre à zéro le compteur des cellules en distance.
2. Le signal ACPs (azimut pulse change) fournit la position instantanée de l'antenne par la génération de 4096 impulsions par tour d'antenne. Ces impulsions passent par un compteur modulo 16 pour générer le signal CPI (Coherent Pulse Interval).
3. Le signal nord fournit une impulsion à chaque passage de l'antenne par le nord magnétique.

# Chapitre 3

# Système d'extraction de plots

## 3.1. Introduction :

Dans ce chapitre, on présente la solution pratique adoptée pour la réalisation du nouveau système d'extraction de plots. Avant de présenter cette solution les points suivant seront présentés :

- Description de l'interfaçage des signaux radar au nouveau système.
- Carte DSP TMS320C40 Bus VME présenté dans l'annexe B.
- Générateur des signaux de synchronisation et de l'IFF pour la simulation à base FPGA.
- Extracteur.

## 3.2. Hardware :

### 3.2.1. Présentation de la carte DSP utilisée :

#### 3.2.1.1. Introduction :

La carte DSP IFF est constituée d'un système de processeur de signal numérique (DSP : Digital signal processor) avec une interface bus VME. Quand la carte DSP TMS320C40 fonctionne dans un rack VME le processeur maître (dans le cas de ce système le processeur est un Pentium II 200Mhz) peut avoir accès à la carte DSP via l'interface bus VME, qui permet de commander et de visualiser les tâches du système DSP voir le schéma synoptique de la carte dans la figure 3.1. [17][18].

#### 3.2.1.2. Environnement TMS320C40 :

le module TMS320C40 est à base d'un DSP de Texas Instruments avec une fréquence d'horloge de 50MHz. Le bus local est interfacé avec une RAM zéro état d'attente (zero wait-state) de 4Moctets de 32-bit de largeur (2 Moctets pour LSTRBO et 2 Moctets pour LSTRBI ). Le bus global du TMS320C40 est interfacé aux supports d'EPROM de 28-pin ,(avec GSTRBO) arrangés en bus de 16-bit de largeur, et 2Moctets de RAM zéro état d'attente (avec GSTRBI ). Le bus local du TMS320C40 est aussi interfacé (avec LSTRBI) à un relais qui commande le relais de la carte.

Les ports de communication 2, 4 et 5 du TMS320C40 sont présents à l'avant du panneau de configuration des connecteurs. Ces connecteurs permettent à plusieurs cartes de se connecter entre elles pour avoir des réseaux de TMS320C40.

Le port de communication 1 de la carte TMS320C40 est interfacé avec de VMEbus via deux (02) FIFOs de 2048-mot —une FIFO pour les données d'entrée du port de communication 1, et une pour la sortie des données du port de communication 1. Ce port permet au maître VMEbus d communiquer avec le DSP.

Le DSP TMS320C40 peut être chargé (Bootloaded) des deux supports des EPROM du bus global, d'un autre DSP TMS320C40 via des connecteurs externes des ports

de communication 2, 4, ou 5, ou du VMEbus maître via la FIFO en entrée du port de communication 1.

Les signaux IIOFO..IIOF3 de TMS320C40 peuvent être conduits (quand ils sont configurés en signaux d'entrées) par quatre signaux externes de niveau TTL se qui permet à la carte DSP d'être interrompue par des sources externes.

L'émulateur standard XDS510 de Texas Instruments connecté au connecteur de l'émulateur standard de la carte DSP TMS320C40 peut être utilisé pour débuguer les applications TMS320C40. La carte DSP TMS320C40 comprend aussi le contrôleur JTAG 74ACT8990 qui peut être interfacé avec les signaux JTAG du TMS320C40. Ce qui permet aux applications du TMS320C40 d'être débuguer directement de l'application du VMEbus maître. Cela nécessite l'emulation Porting Kit de Texas Instruments, TMDX3240040 qui n'est pas inclus dans cette carte.

### 3.2.1.3. Interface bus VME :

L'interface VMEbus travaille comme VMEbus esclave de 16-bit et permet aux applications de la carte VMEbus maîtresse (le processeur Pentium) de contrôler et de visualiser les résultats des applications des TMS320C40s. L'interface se constitue de quatre parties qui sont : le registre d'état, le registre de configuration, l'entrée sortie avec le port de communication N° 1 des FIFOs et le contrôleur du JTAG.

Le registre d'état permet au VMEbus maître de lire les états des flag de vide, demi-plein et plein des FIFOs d'entrée et de sortie du port de communication n°1, lire l'état de la sortie de demande d'interruption VMEbus de la carte DSP.

Le registre de configuration permet au VMEbus maître d'installer l'interrupteur VMEbus de la carte DSP. Le VMEbus maître peut sélectionner une parmi 16 sources d'interruptions de la carte DSP possible, sélectionner les modes d'interruption ; front ou niveau trigger, et peut aussi réinitialiser le TMS320C40 via le registre de configuration, et allumer la LED rouge du panneau avant de la carte.

Les FIFOs d'entrée sortie sont du port de communication n°1 du DSP TMS320C40 sont interfacées au VMEbus, les cycles d'écriture du VMEbus dans la FIFO écrit 32-bit mots dans la FIFO en entée du port de communication 1(en utilisant 2 cycles d'écriture VMEbus de 16-bit) et automatiquement la carte DSP essaye de transférer des mots de l'entrée de la FIFO vers le DSP TMS320C44 via son interface le port de communication 1.

L'interface VMEbus peut être configurée pour répondre aux cycles VMEbus de transfert par block et simple A32 et/ou A24. Les cycles de transfert par block peut être utilisés pour le transfert rapide de larges blocks de données de ou vers les FIFOs de port de communication.

### 3.2.2. Emplacement mémoire :

#### 3.2.2.1. Emplacement Mémoire du TMS320C44 :

L'emplacement mémoire du TMS320C44 est défini dans les publications de Texas Instruments TMS320C44 User's Guide (numéro de la littérature SPRU0663B).

L'emplacement et la dimension du bus Global et du bus Local du TMS320C44 à l'intérieur de la carte mémoire sont programmables et sont configurables par l'initialisation des valeurs des registres de contrôle de l'interface de la mémoire Global et de la mémoire Local.

#### 3.2.2.2. Emplacement Mémoire du VMEbus :

Les 16 bits poids fort de l'adresse de base de la carte DSP dans l'espace d'adresse de VMEbus sont initialisés par les switchs SW3(poids fort), SW4, SW5 et SW6(poids faible).Chaque carte DSP occupe 64 KOctets dans l'espace d'adresse du VMEbus. Les différents circuits de la carte DSP ont leur emplacement mémoire dans les 64Koctets dont BASE indique l'adresse de base de la carte DSP dans l'espace d'adressage de VMEbus comme suit :

| | |
|---|---|
| Registre de configuration : | BASE à BASE+0xff |
| Registre d'état : | BASE+ 0x0100 à BASE+0x1ff |
| Contrôleur de JTAG : | BASE+0x200 à BASE+0x2ff |
| FIFOs du port Communication : | BASE+0x300 à BASE+0xffff. |

### 3.2.3. Outils de développement.

Les outils de développement qui seront présentés dan cette partie sont essentiellement les outils logiciels utilisés pour générer le programme ALL.BTL qui est sera chargé dans la carte DSP.

Le programme principal ou le noyau est écrit avec le C compiler de Texas Instruments, il fait appel à tous les sous programmes, soit écrits avec le C compiler aussi pour le cas de la routine de corrélation et l'installation des interruptions ou avec l'assembleur pour le cas des interruptions et l'installation du TIMER0, des DMA1 et 4. L'assembleur pour le cas des interruptions et le setup du TIMER0, des DMA1 et 4.

Installation de DSPTOOLS:

Installer la version 4.70 copyright (c) 1987-1996 qui se trouve sur le CD de Texas Instruments DSPTOOLS.

Copier tous les fichiers dans le répertoire DOS ou WIN du CD dans un répertoire de la racine de votre disque dur, nommé de préférence DSPTOOLS. Modifier le fichier AUTOEXEC.BAT en rajoutant les chemins suivants:

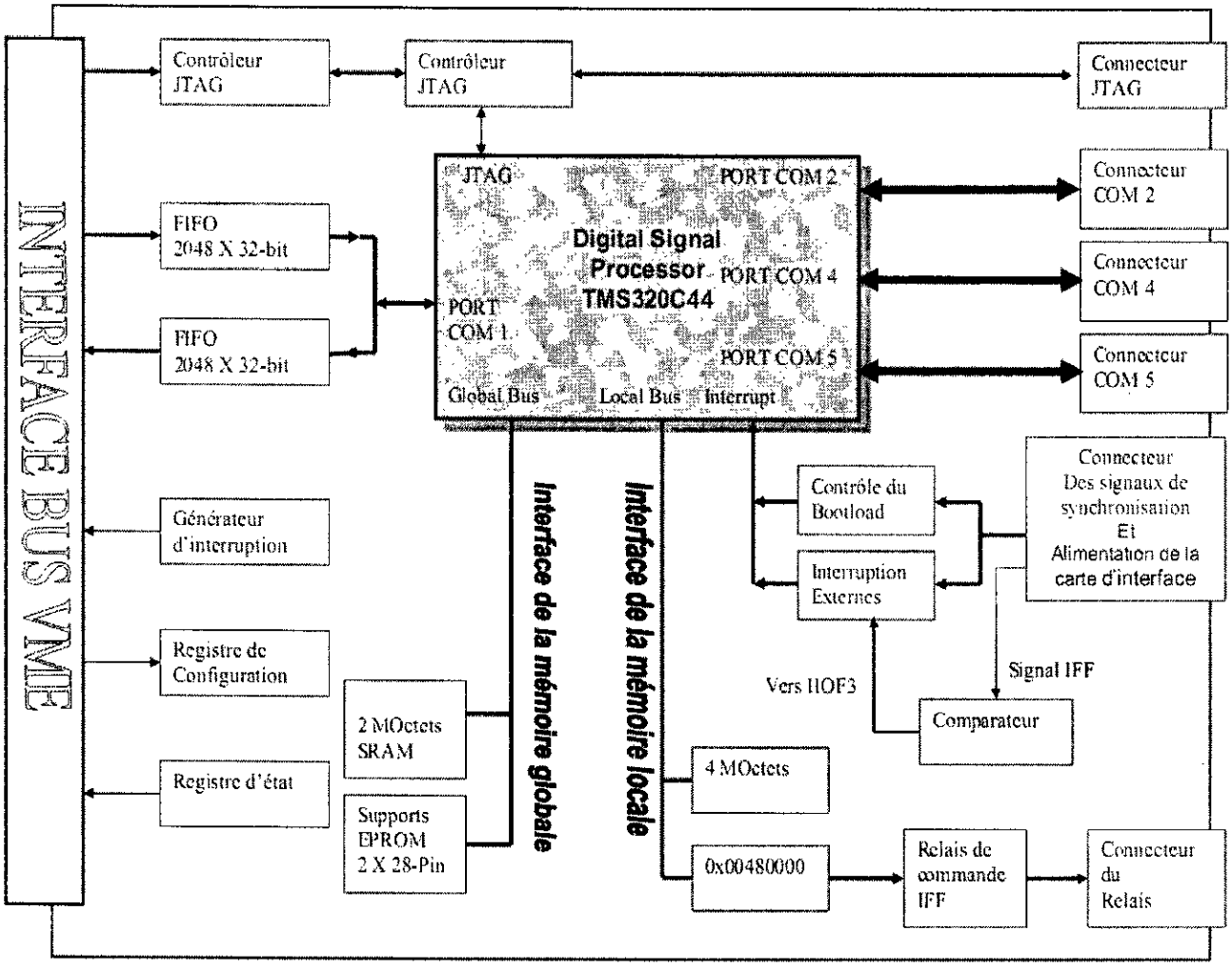



**Figure 3.1 : Schéma bloc de la carte DSP TMS320C44 d'IFF**

SET PATH=C:\DSPTOOLS;%PATH%

SET A_DIR=C:\DSPTOOLS

SET C_DIR=C:\DSPTOOLS.

Important:

Il faut exécuter la commande suivante:

mk30 -v40 --h -o2 -mn prts40.scr

pour créer les librairies pour les TIMERs, les DMAs, les COMPORTs et les INTERRUPTIONS.

Présentation du Programme d'IFF:

Le programme principal est le fichier: ALL.C.

Les sous programmes sont les suivants:

1-var.h: ce fichier contient toutes les variables utilisées dans le programme ALL.C et le reste des procédure.

2-tri.h: ce programme est la procédure de triage des cibles à interroger.

3-process.h: procédure de validation et de convertion.

4-init.h: procédure d'installation des interruptions IIOF0..3 et DMA1, DMA4.

5-comm.h: procédure d'envoie des résultats d'extraction vers le host.

6-cor.h: procédure de corrélation des plots pour l'unité de poursuite.

7-LastCpi.h: procédure du dernier CPI ou passage du nord.

Génération du fichier exécutable :

La commande de compilation pour générer le fichier all.obj est la suivante:

**cl30 -v40 -k -g -as all.c**

Le seul fichier écrits en assembleur est le fichier assem.asm, l'assemblage de ce fichier avec l'option "-v40" pour le l'objet 'C40 pour générer en sortie a comme suit :

**asm30 -v40 assem.asm**

Les fichiers à inclure pour assem.asm sont:

makectl.asm: ce fichier permet de calculer les mot de contrôle des DMA.

dma1.asm: permet de définir le mot de contrôle de DMA1.

dma4.asm : permet de définir le mot de contrôle de DMA1.

La commande de linkage est la suivante: **lnk30 all.cmd**

Les outils les plus importants utilisés dans ce travail sont :

| | |
|---|---|
| ASM30 | assemblage. |
| CL30 | Compilation. |
| Link30 | fait le lien entre les fichiers .Obj qui sont le résultat ; soit de l'assemblage |

(ASSEM.OBJ) soit la compilation(ALL.OBJ) pour avoir en sortie un seul fichier exécutable.

Le fichier de commande de notre application est ALL.CMD voir la table 3.1:

Table.3.1. Fichier des commandes ALL.CMD

```
-c
all.obj

-m all.map
-o all.out
-e c_int00

-l c:\dsptools\prts40.lib                    /* GET RUN-TIME SUPPORT        */
-l c:\dsptools\rts40.lib

-stack 0x400                    /* 1K STACK                */
-heap  0x400                    /* 1K HEAP                 */

assem.obj
roth2xy.obj

/* SPECIFY THE SYSTEM MEMORY MAP */
MEMORY
{
    ROM:   org = 0x00000000  len = 0x800    /* on-chip rom */
    RAM0:  org = 0x002ff800  len = 0x400    /* RAM block 0 */
    RAM1:  org = 0x002ffc00  len = 0x400    /* RAM block 1 */
    LOCAL1:  org = 0x00300000  len = 0x80000  /* local bus external memory */
    LOCAL2: org = 0x00400000  len = 0x80000  /* local bus external memory */
    GLOBL1: org = 0x80000000  len = 0x10000  /* global bus external memory*/
    GLOBL2: org = 0x80400000  len = 0x80000  /* global bus external memory*/
    GLBLPRTCTL: org = 0x00100004 len = 0x1
    LCLPRTCTL:  org = 0x00100000 len = 0x1
    TIMER0: org = 0x00100020  len = 0x9
    TIMER1: org = 0x00100030  len = 0x9
    DMA0:  org = 0x001000a0  len = 0x9
    DMA1   org = 0x001000b0  len = 0x9
    DMA2   org = 0x001000c0  len = 0x9
    DMA3   org = 0x001000d0  len = 0x9
    DMA4   org = 0x001000e0  len = 0x9
    DMA5   org = 0x001000f0  len = 0x9
    COM1   org = 0x00100050  len = 0x4
}
/* SPECIFY THE SECTIONS ALLOCATION INTO MEMORY */
SECTIONS
{
  ".vector" > RAM0
  .text:   > LOCAL1              /* CODE                          */
  .cinit:  > LOCAL1              /* INITIALIZATION TABLES         */
  .const:  > LOCAL1              /* CONSTANTS                     */
  .stack:  > RAM1                /* SYSTEM STACK                  */
  .data:     > LOCAL1            /* Assembler data                */
  .sysmem: > RAM1                /* DYNAMIC MEMORY (HEAP)         */
  .bss:    > LOCAL1, block 0x10000  /* VARIABLES                   */
}
```

Makeboot:

Après compilation et linkage le fichier **ALL.OUT** est généré. Pour MAKEBOOT les fichiers utiles sont:

1-ALL.OUT: résultat de la compilation et du linkage.

2-ALL.BTM: fichier à ouvrir en précisant la dresse du début du programme qui se trouve dans le fichier ALL.MAP.

exemple:

all.out
-o all.btl
-map all.mxp
-e 003004dfh (adresse à changer s'il y a lieu.)

La commande est la suivante:

**MAKEBOOT ALL**

Le résultat est ALL.BTL sous le format Tektronix, et c'est ce fichier qui sera chargé dans le DSP TMS320C44.

### 3.2.4. Présentation de la solution hardware :

Les principaux signaux pour le fonctionnement d'un radar sont comme cité dans le chapitre 2, le signal ACPs, le signal nord, le signal PRF et le signal IFF à la sortie du déchiffreur. La figure 3.2 montre le schéma synoptique de la solution électronique retenue en premier lieu ; La figure montre le compteur de portes distance qui est remis à zéro à chaque PRF pour la référence distance 0 Km avec un signal d'échantillonnage de 500KHz pour résolution de calcul de 2ps, le compteur azimut dont les entrées sont ; le signal ACPs (4096 impulsions par tour d'antenne) pour déterminer la position instantanée de l'antenne et le signal nord qui remis le compteur à zéro aupassage de l'antenne par le nord, les détecteurs de front montant et descendant pour le signal de réponse IFF, les buffers pour le verrouillage des valeurs des compteurs distance et azimut commandés par le front montant du signal d'identification et un système de commande par relais de l'émetteur récepteur de l'interrogateur.

Les données distance et azimut seront récupérées via un bus de données une fois que le processeur (Microprocesseur ou Microcontrôleur) ait reçu une interruption via un bus de commande, ces données seront traitées par le processeur pour les envoyer à nouveau pour exploitation et visualisation par une autre unité.

Après l'étude du système existant et la détermination des signaux nécessaires pour la réalisation du projet de modernisation de la station radar ASRx dans ses différentes parties pour l'implantation du MTI (Moving Target Indicator), du détecteur CFAR et de IFF (l'étude et l'implantation du MTI et du détecteur CFAR ne sont pas traités dans cette thèse), la première

carte réalisée pour ce projet est la carte DSP TMS320C40 avec deux convertisseurs analogiques numériques pour l'acquisition des signaux échos cohérent et amplitude (voir figure 3.3) et l'entrée de l'interruption IIOF03 est une entrée numérique de niveau TTL (voir figure 3.4).

Ainsi dans la partie IFF des modification au niveau des cartes DSP TMS320C40 sont obligées. Comme le montre la figure 3.5 le signal IFF qui a un niveau variant entre 5 et 20V doit passer par des étages d'adaptation et d'amplification pour ainsi maintenir le signal à l'entrée du comparateur de la carte DSP TMS320C40 entre 5 et 6V qui à sont tour attaque l'entrée de l'interruption externe IIOFO3 du processeur TMS320C40.

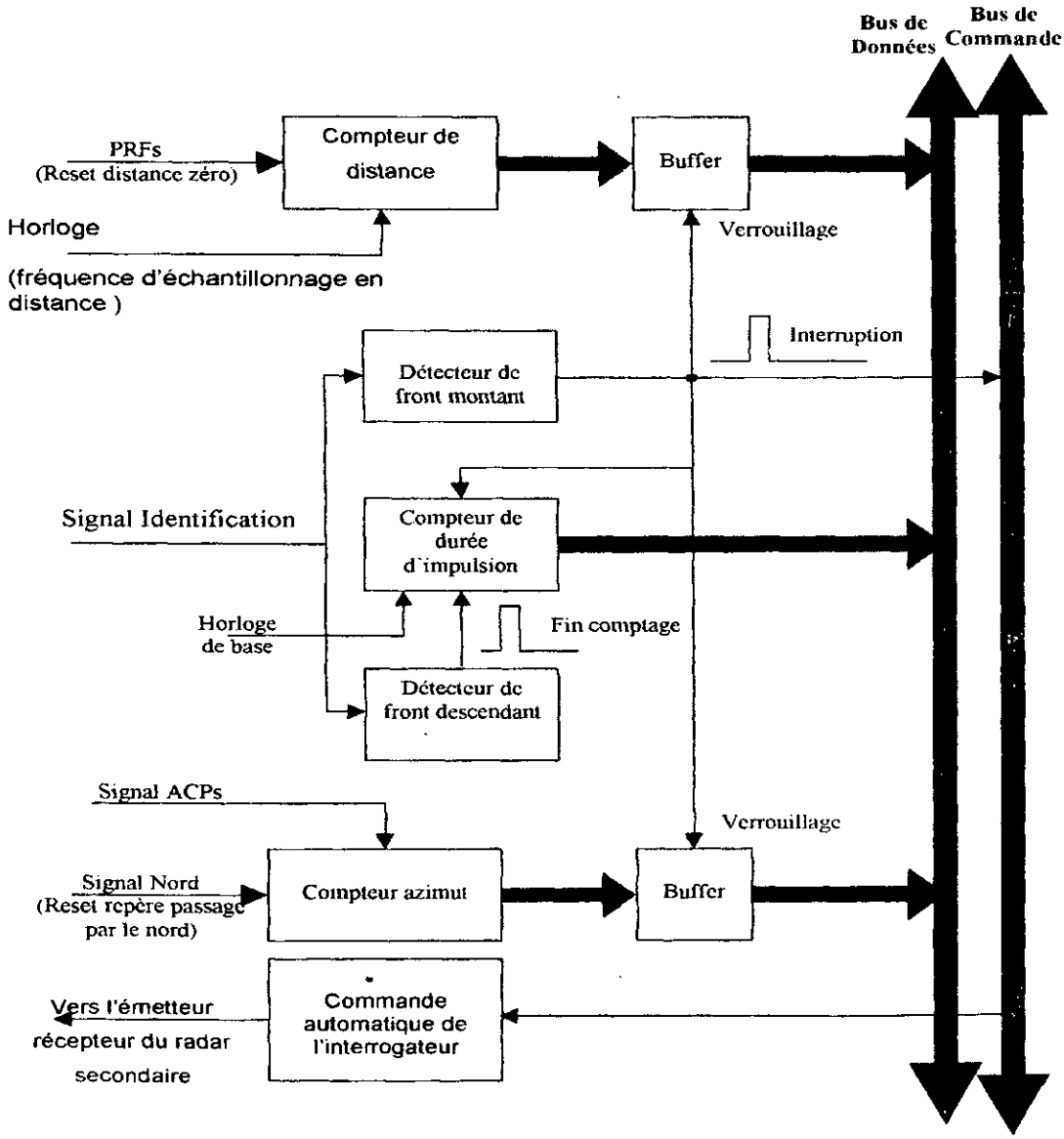



Figure 3.2: Le schéma synoptique de la solution de base électronique

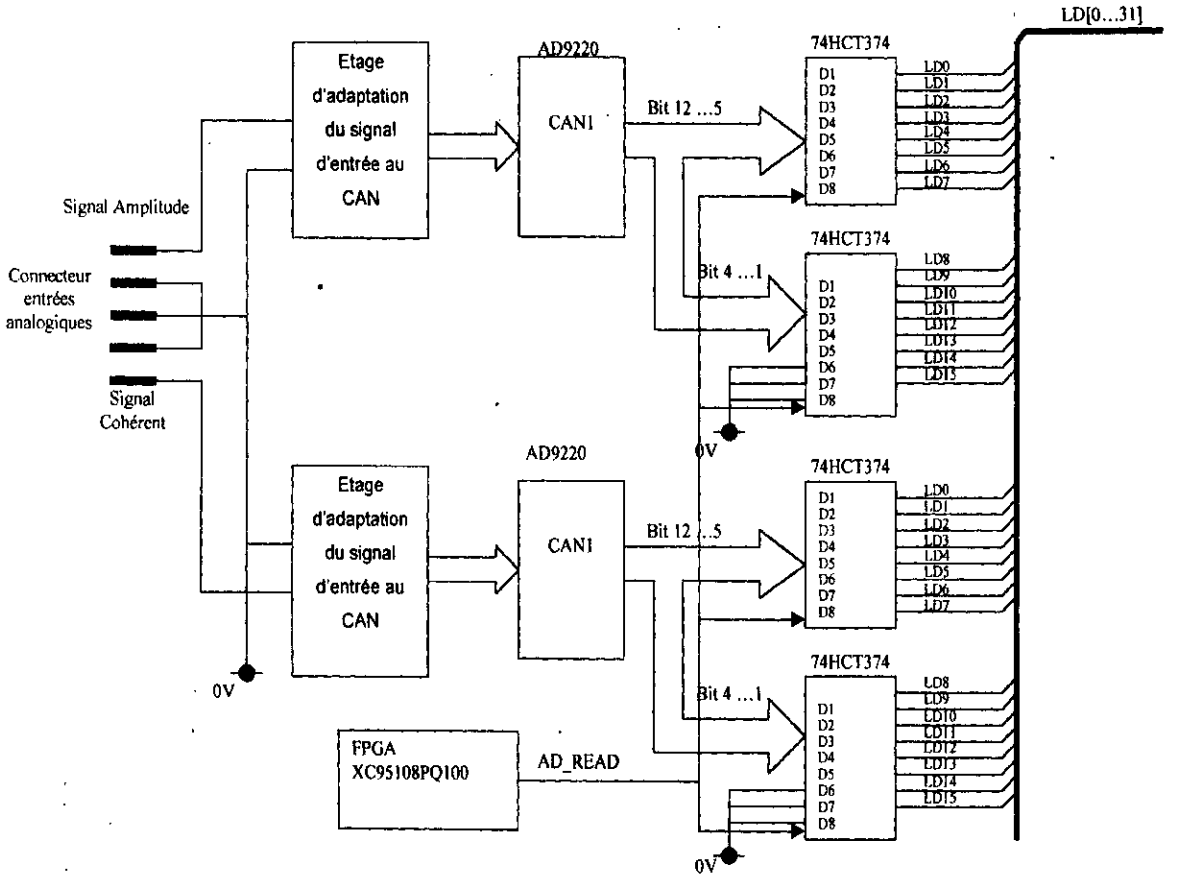



Figure 3.3: Carte DSP TMS320C40 avec convertisseurs analogiques numériques

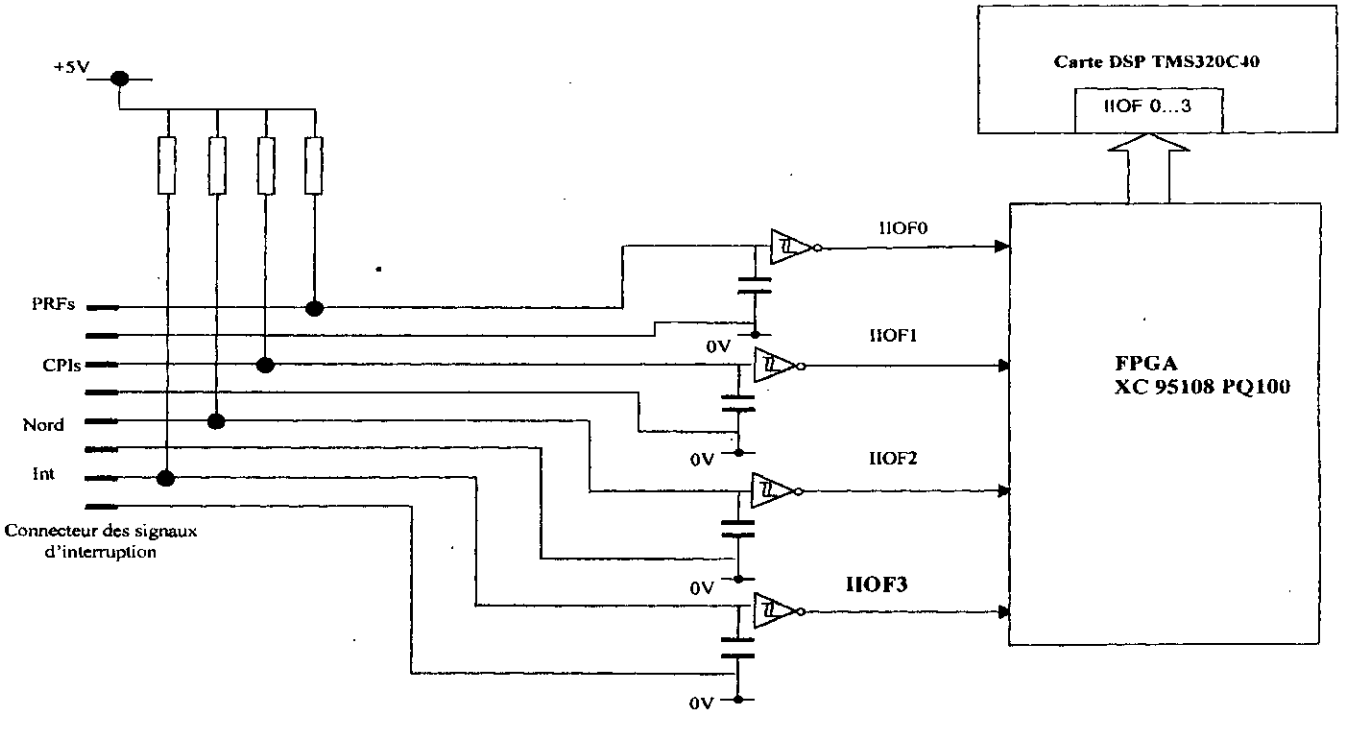



Figure 3.4:Cheminement du signal d'interruption IIOF0-3

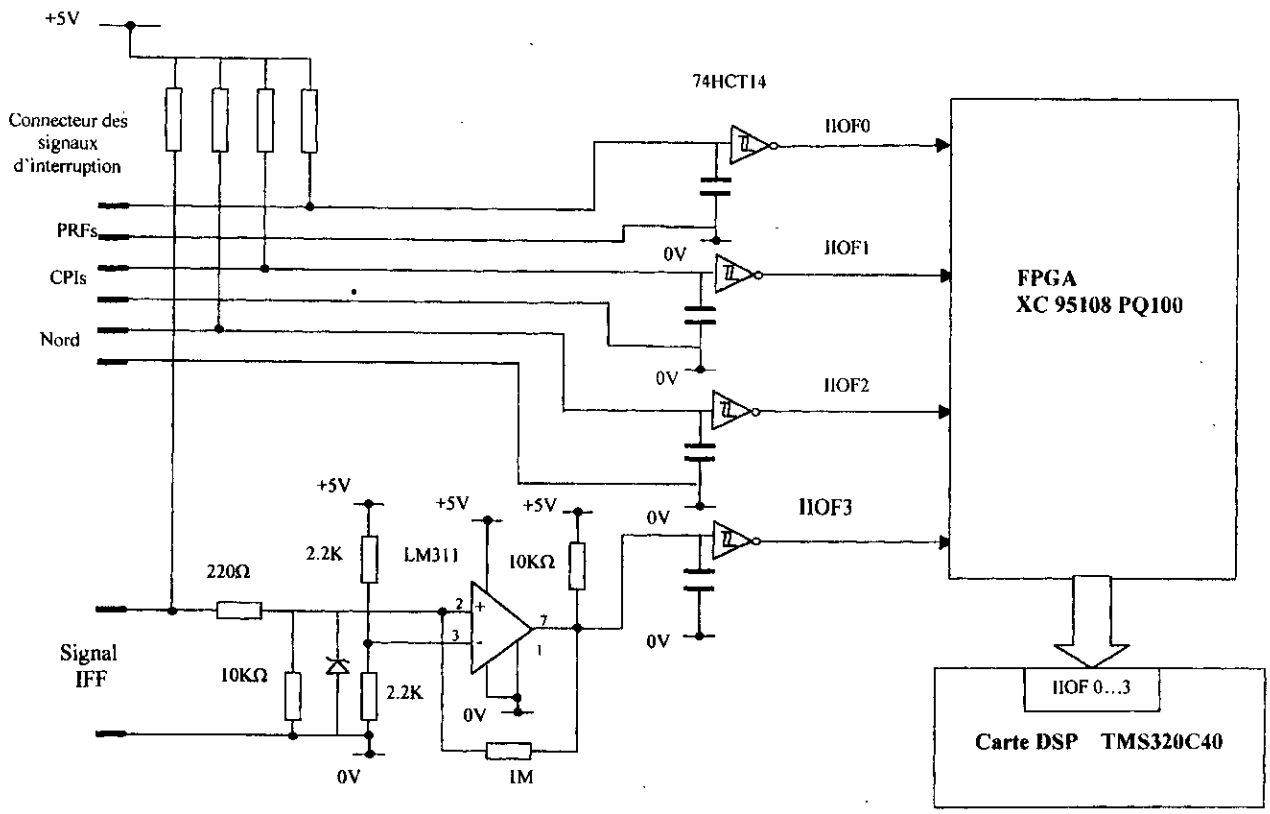



Figure 3.5:Cheminement du signal d'interruption IIOF0-3 dans la carte DSP IFF

Pour la commande du relais, la solution consistait à utiliser l'adresse mémoire 0x480000 des circuits 74HCT374 de la figure 3.3 pour les remplacés par un seul circuit 74HCT174 pour mettre le bit 0 du registre à 5V ou OV pour commander l'ouverture ou la fermeture du relais voir la figure 3.6.

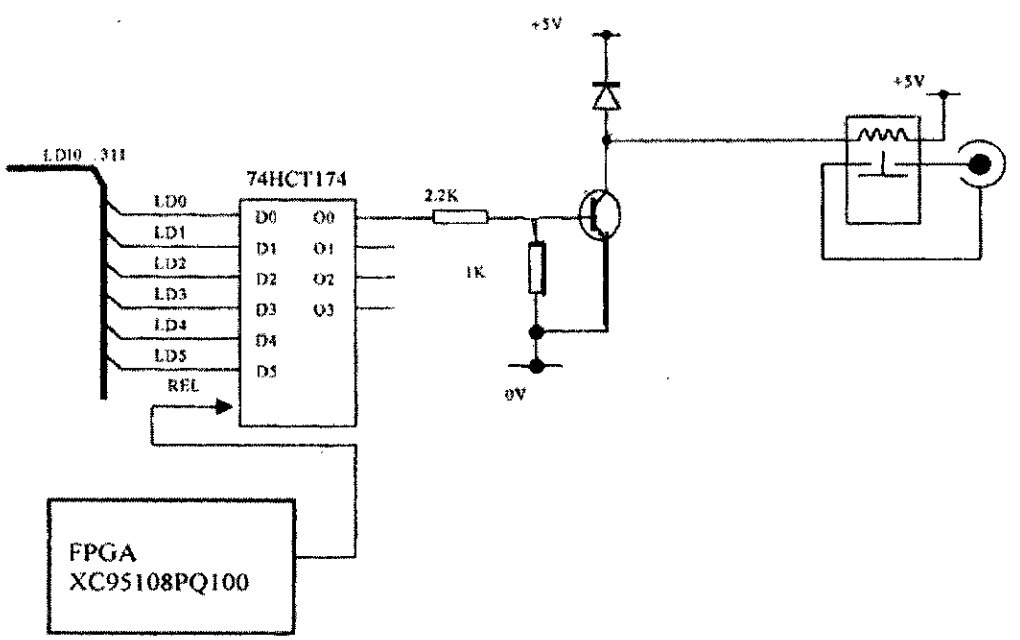



Figure 3.6: Commande du relais d'interrogation

Après les modifications de la carte DSP TMS320C40 d' IFF on obtient la solution hardware de la figure 3.9. Le signal IFF est piqué à la sortie du déchiffreur pour être envoyé vers la carte d'interface pour filtrage et adaptation, il arrive à l'entrée de la carte DSP avec un niveau allant de 5 à 6 V. A la sortie du comparateur, on aura un niveau logique TTL qui va attaquer l'entée IIOF3 du processeur TMS320C40.

Le relais est commandé par la mis d'une valeur impaire dans l'adresse mémoire 0x48000 qui est l'adresse des deux convertisseurs analogiques numériques.

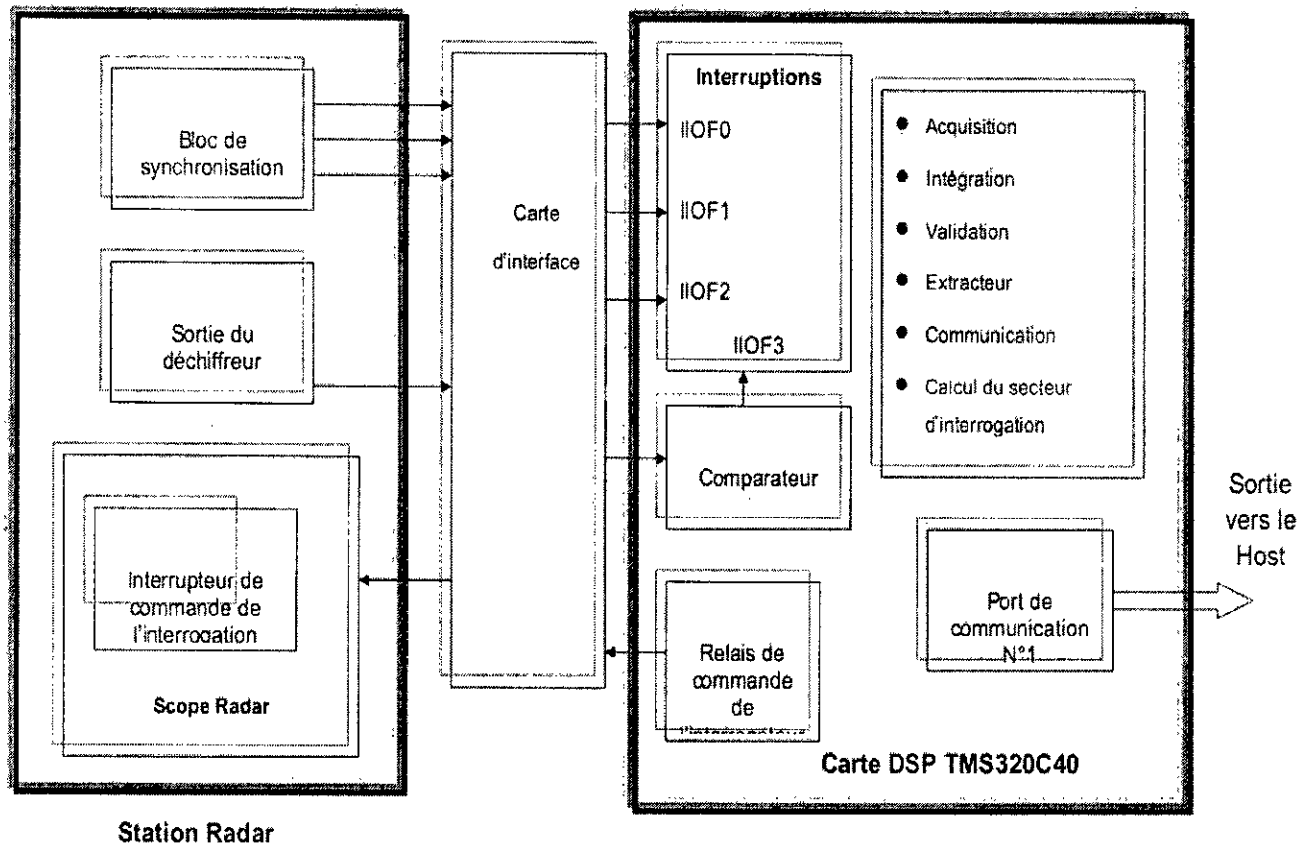



**Figure 3.7 : le Schéma synoptique général du nouveau système**

## 3.3. Software :

### 3.3.1. Notion des portes distance et azimut dans une implantation :

L'implantation d'un algorithme d'extraction de plots radar sur carte DSP avec la contrainte temps réel dépend principalement de la façon dont les portes distances et azimuts sont choisies et organisées dans la mémoire du processeur. Un de ces choix est la résolution du radar, c'est à dire la résolution en distance et la résolution en azimut. Dans ce travail le radar a pour résolution en distance 750 m ; 5 ps en appliquant la formule pour le calcul de la distance parcourue par une impulsion exploratrice:**D=(T x C)12 ;**

***D** : la distance en mètre parcourue par l'impulsion exploratrice.*

*T : la durée parcourue par l'impulsion exploratrice.*

*C : la vitesse de la lumière.*

Ainsi : ***(5x10"⁶ x 3x10⁸)/2=750 m.***

La résolution en azimut est de **4°** qui est liée à l'ouverture du lobe principal du radar secondaire voir figure 3.8.

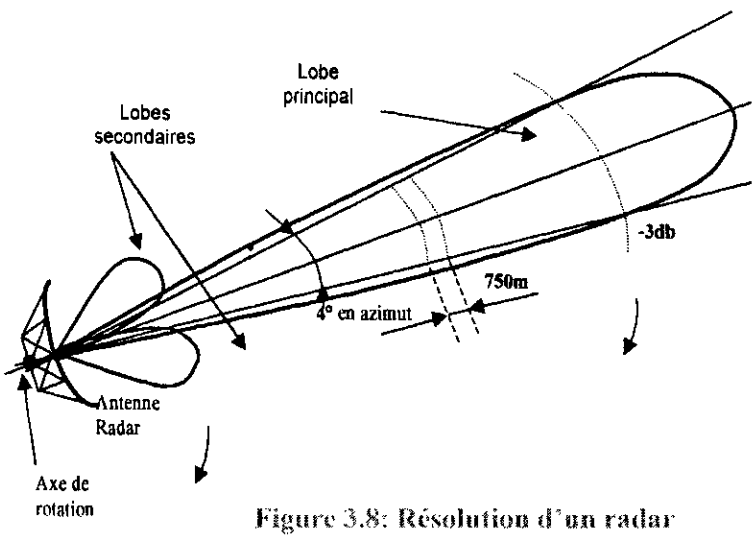



Figure 3.8: Résolution d'un radar

D'après le théorème Shannon la fréquence d'échantillonnage doit être supérieure ou égale à deux fois la fréquence maximale du signal à échantillonner. Le signal IFF à la sortie du bloc de corrélation est de 5 ps, ainsi la dimension de la porte en distance est de 2ps de cela le nombre de portes en distance dépend aussi de la portée du radar qui est de 300km ; La portée radar d'une seule porte distance en mètre est : Dist= (T*c)/2, soit Dist = 2ps*3*1 $0^8$/2=300m

D'où le nombre de portes en distance est donné par:

La portée maximale/Dist = 300000/300=1000.

Le nombre de porte azimut est codé sur 8 bits:360° / 256 = 1.40625°.

Ainsi, notre espace aérien sera présenté dans notre algorithme par une matrice à 02 dimensions 256x1000 voir la figure 3.9.

### 3.3.2. Extracteur de données radar secondaire (Plot Extractor):.

#### 3.3.2.1. *Introduction :*

Tous les radars. primaires et secondaires modernes sont équipés extracteurs dans leurs systèmes du traitement du signal pour les raisons suivantes [4][36] :

- Le transfère des données de la cible du radar de surveillance vers des centres des opérations aériennes lointains pour l'exploitation avec des débits très faibles par rapport au transfère du signal analogique.

- Visualiser l'historiques des cibles radars.
- Enregistrer les situations radar sur disque magnétique et restituer ces situations à tout moment.
- Faciliter l'exploitation de la situation radar en introduisant des informations supplémentaires sur le scope radar comme les caractères alphanumériques, les symboles...
- Former des pistes avec le choix de l'historique.

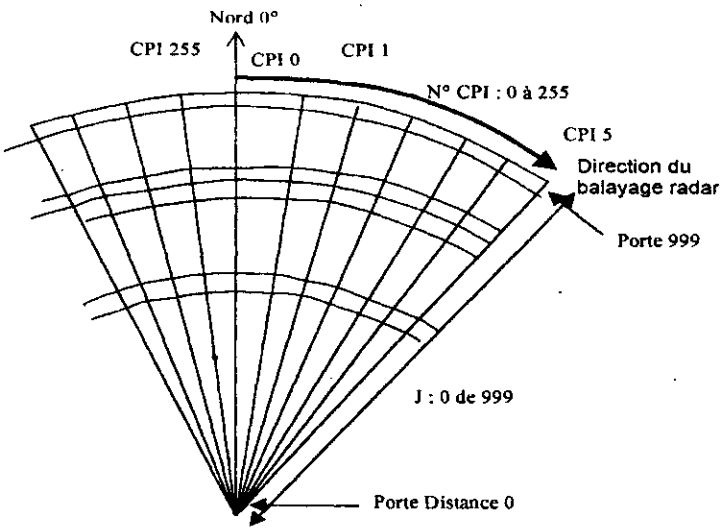



**Figure 3.9: Portes distance et Portes azimut**

### 3.3.2.2. *Le principe de fonctionnement :*

L'extracteur de plot est nécessaire pour imiter la visualisation des PPIs traditionnels, et prendre des décisions sur les spots claires s'ils constituent un signal désiré ou une fausse alarme. La figure 3.10(a) montre une portion d'un PPI avec l'éclatement d'un espace pris pour exemple, dans la figure 3.10(b). Selon la forme des échos, deux cibles sont représentées comme elles le sont visualisées habituellement. Dans la figure éclatée, il est présenté le signal échos radar pour chaque PRF avec une moyenne de 10 impulsions pour l'ouverture du lobe ou par CPI. La cible RI est clairement définie, en fournissent une impulsion pour chaque transmission consécutive durant tout le balayage du lobe de l'antenne de cette cible. La seconde cible R2 n'est pas clairement définie. Cela est du soit à le cible dont la surface échos de fluctuation diminue au dessous de la moyenne du temps dont doit durer lobe de l'antenne, soit les échos sont ceux de deux petites cibles qui se chevauchent, soit un petit avion avec des fausses détections à la même distance et enfin soit tous les échos sont de fausses alarmes. Généralement les extracteurs de plot à double seuils sont utilisés. Le premier seuil est de voir si le seuil de détection est dépassé ; c'est à dire, voir si l'amplitude de la sortie du radar dépasse un niveau qui constitue un événement autre que le bruit. Beaucoup de

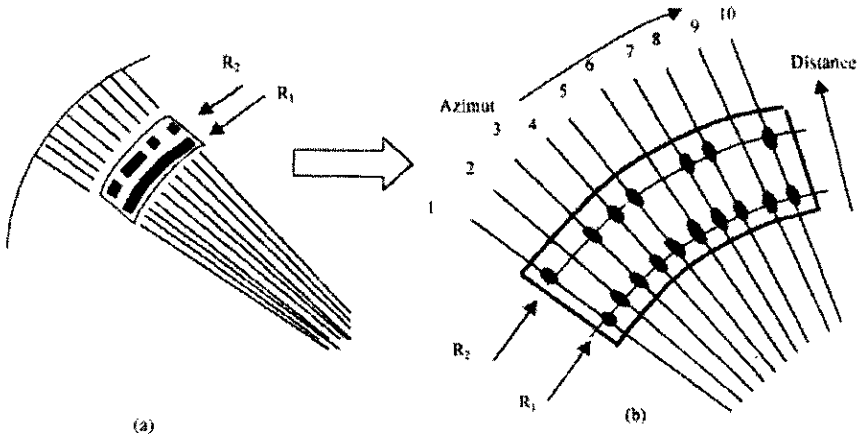



**Figure 3.10: (a). Apparence de deux cibles sur PPI. $R_1$ est forte alors que $R_2$ est faible. (b). Vue agrandie des deux cibles. La plus forte produit des signaux détectables pour chacun des 10 échantillons successifs montrés ; la faible produit des absences intermittentes en détection.**

processeurs de signal incluent ce circuit de la prise de décision mais, s'il n'y est pas il doit être incorporé dans l'extracteur de plot. Ces événements sont recherchés dans chaque cellule de résolution en temps réel. Maintenant une autre décision doit être prise – est ce que chacun de ces événements forme le modèle attendu d'un ensemble de détections d'une cible quand lobe de l'antenne la balaye? Cette seconde décision est faisable par plusieurs manières ; la plus commune est la technique de la fenêtre glissante (Sliding Windows).

Dans la technique de la fenêtre glissante (Sliding Windows), des arrangements sont faits pour stoker la décision si le premier seuil a été dépassé ou pas pour chaque incrément en distance d'une période de répétition de l'impulsion. Généralement il est de fournir au moins deux et il peut y avoir plus de quatre incréments distance par cellule de la résolution, mais la décision est exprimée quant à elle par 1 si une cellule de la résolution contient un premier seuil dépassé et un 0 s'il ne fait pas. Pour la prochaine période un même incrément en résolution en distance est vérifié et son état 1 ou 0 est stocké avec son prédécesseur. Le stockage est équivalent au nombre d'impulsions par angle de résolution qui est toujours donné voir la figure. 3.11.

Une autre raison d'utilisation d'extracteur qui n'a aucun lien avec la numérisation de la situation radar et le déport de ces informations vers un centre de traitement lointain, est présentée ci-dessous. Malgré les algorithmes puissants développés dans le domaine de la détection automatique radar pour réduire la probabilité de fausse alarme, la même cible peut être détecter dans plusieurs cellules adjacentes en distance, en azimut et en site. Ainsi l'unité de poursuite sera appelée à traiter plusieurs plots primaires et secondaires causés par ces multiples détections pour une même cible; ce qui présente un inconvénient important.

Pour remédier à ce problème, les plots secondaires subissent en premier lieu un pré traitement à la sortie du bloc de validation avant leur transfert vers l'unité de poursuite. Ce pré traitement est réalisé par cette extracteur. L'extracteur est placé à la sortie du détecteur pour le cas du radar primaire et à la sortie du déchiffreur pour le cas du radar secondaire (voir la figure

3.12) qui consiste à réunir tous les plots primaires élémentaires et les plots secondaires élémentaires d'une même cible pour former des classes. Une estimation de la distance et de l'azimut (pour le cas d'un radar à deux dimensions) ainsi que le reste de données délivrées par le détecteur et la sortie du décodeur du radar secondaire pour chaque classe afin de fournir un plot centré. Ainsi l'unité de poursuite aura à traiter uniquement des données d'une seule cible soit à traiter un plot primaire et un plot secondaire, ce qui lui fait un gain de temps important en traitement de données radar en plus du problème de communication qui surgie au niveau des cartes DSP TMS320C40 et la carte host le Pentium II qui cause beaucoup de problème pour la récupération de tous les flux de données des cartes DSPs et afficher toutes ces données en temps réel sans qu'il y ait perte d'information ou blocage du programme de visualisation et de tout le système.

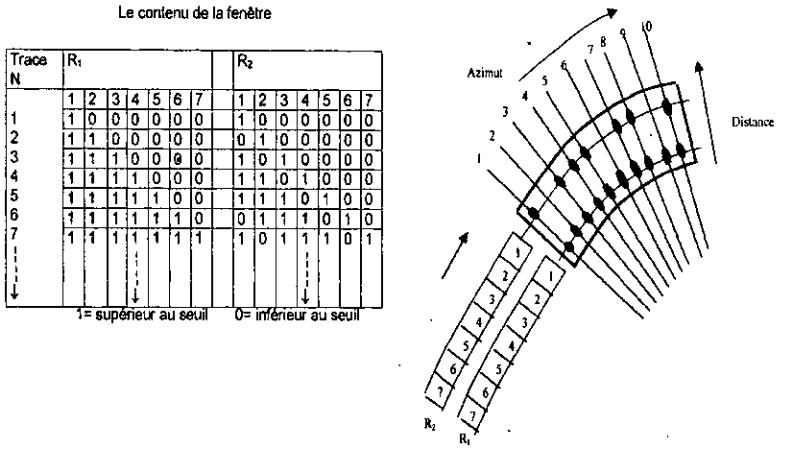



**Figure 3.11: Détecteur a fenêtre glissante. Le résultat de la détection des deux cibles est présenté dans le tableau. Les cellules des cibles avec le balayage sont remplies de 1s ou de 0s en fonction du niveau du signal (ou bruit). Différents algorithmes peuvent être utilisés pour déterminer s'il y a cible ou non.**

Ces détections multiples sont dues à un problème fondamental d'ordre pratique, qui est lié à la forme de l'impulsion de l'écho de cible qui est rarement connue dans un environnement radar réel, comme le montre la figure 3.13 selon l'échantillonnage de signal écho, plusieurs cas se présentent. Ces détections multiples sont dues aussi, en premier lieu, au fait que les cibles ne sont pas des réflecteurs ponctuels mais au contraire ils ne sont que des réflecteurs complexes. En second lieu, la forme de l'impulsion de l'écho de cible n'est pas connue en raison des incertitudes dans le comportement de l'impulsion transmise et la bande passante du récepteur qui n'est pas parfaite. Enfin, l'étalement de l'impulsion écho du à la réflexion des objets très proches tel que les constructions, les tours, les pylônes ...etc.

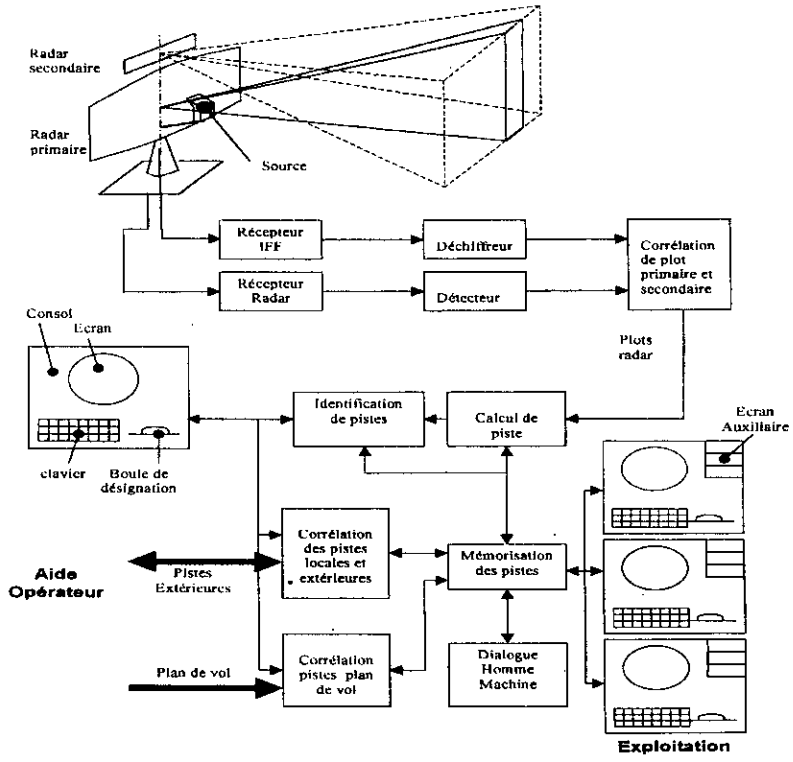



Figure 3.12: Chaîne de traitement radar automatique

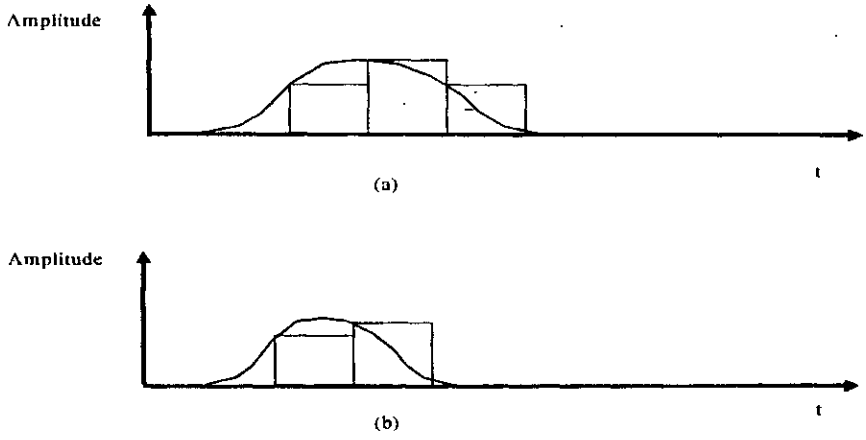



Figure 3.13: Effet de l'échantillonnage, (a) 3 plots primaires pour une même cible (b) 2 plots primaires pour une même cible

Dans le cadre du projet qui consiste en la modernisation de la station radar de technologie analogique, l'utilisation d'une unité de poursuite pour la corrélation des plots primaires avec les plots secondaires ainsi que la formation des pistes est nécessaire. Cette unité de poursuite a pour tâche le traitement de milliers de plots primaires et secondaires par tour d'antenne délivrés par le détecteur et le déchiffreur parmi les quels on retrouve des plots utiles et de faux plots.

Vu la contrainte temps de traitement de toutes les données délivrées par le détecteur, l'utilisation d'un extracteur évitera à l'unité de poursuite de faire le même traitement plusieurs fois pour une même cible, et faire uniquement le traitement d'un seul plot centré en distance et en azimut.

### 3.3.3. Algorithme:

Le choix de l'algorithme est tiré de celui du MTD-II (Moving Target Detector); les règles de formation des classes et le calcul du centroïde [37].
Les positions du lobe de l'antenne pour un radar de surveillance de type tridimensionnel sont montrées dans la Figure 3.14. En conséquence, une cible peut être détectée dans plusieurs positions azimut élévation du lobe en fonction de sa dimension et le lieu où se trouve la cible. La cible peut être aussi détectée dans les cellules distances adjacentes [35]. Ainsi la notion de fusion ou de classe est recommandée pour fusionner toutes ces détections en une seule détection centrée de la cible. Cette notion de classe décide si une nouvelle détection est adjacente à une des classes déjà ouvertes, alors cette nouvelle détection est ajoutée à ces classes, si non elle ouvre une nouvelle classe. Deux détections sont adjacentes, si deux de leurs paramètres (distance, azimut et élévation) sont les même et le paramètre restant n'est différent que d'une porte de résolution: cellule distance AR, ouverture du lobe $\Delta\theta$ ou la largeur bande de l'élévation.

#### 3.3.3.1. Règles de formation de classes:

Les règles de formation de classe sont les suivantes (voir figure 3.15):

- Un plot qui n'est corrélé avec aucune des classes, il ouvre (crée) une nouvelle classe.

Pour corréler un plot à une classe il faut qu'il soit voisin à au moins un des plots de la classe en distance et en azimut.

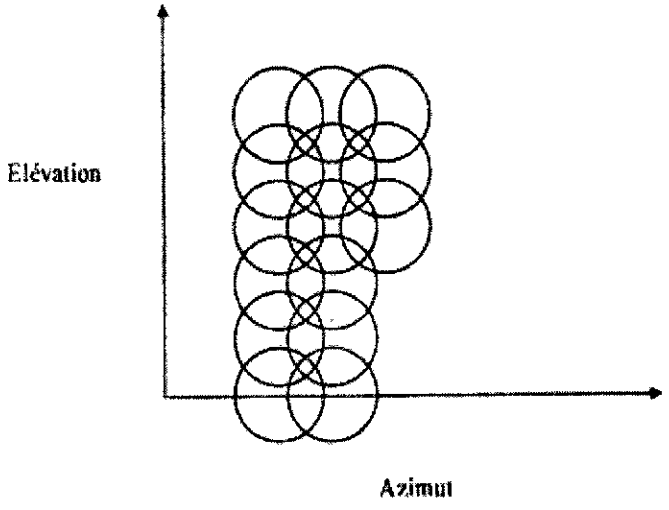



Figure 3.14 : Lobe d'une antenne tridimensionnelle

- Un plot ne peut-être corrélé à une classe si celui-ci provoque le surdimensionnement de la classe, il induit une classe plus large que la limite d'extension permise (30R, 4A0 dans le cas du MTD).
- Une classe est fermée lorsqu'elle n'a pas reçu de plots durant un CPI. Si un plot est corrélé avec une

classe, puis corrélé avec une autre classe, ce plot doit être inclus dans les deux classes.

➢ A la fin de chaque porte azimut, si une classe on ne lui a pas associé au moins un plot, cette classe est à fermer, et passe à l'unité de centrage.

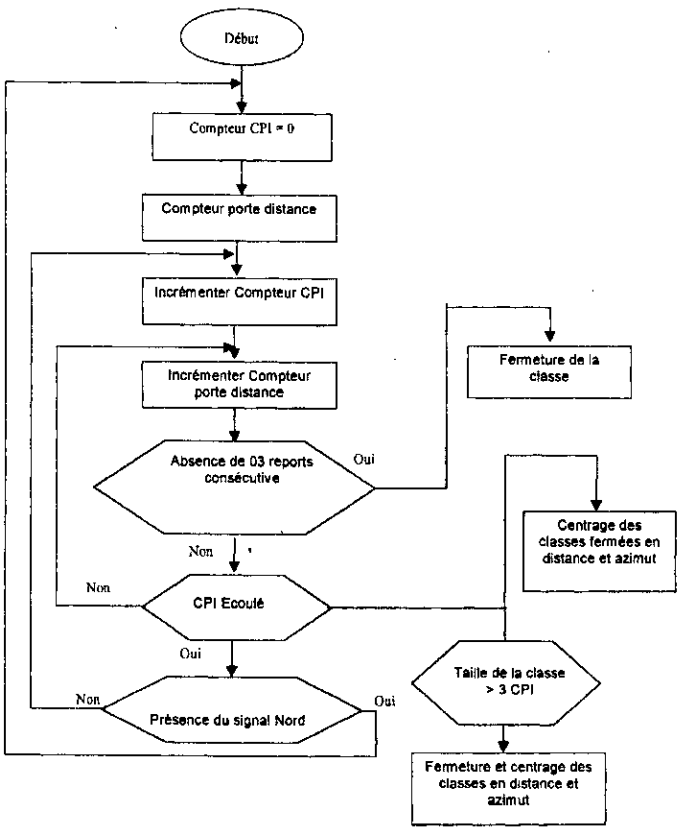



Figure 3.15: Organigramme de la règle de fusion

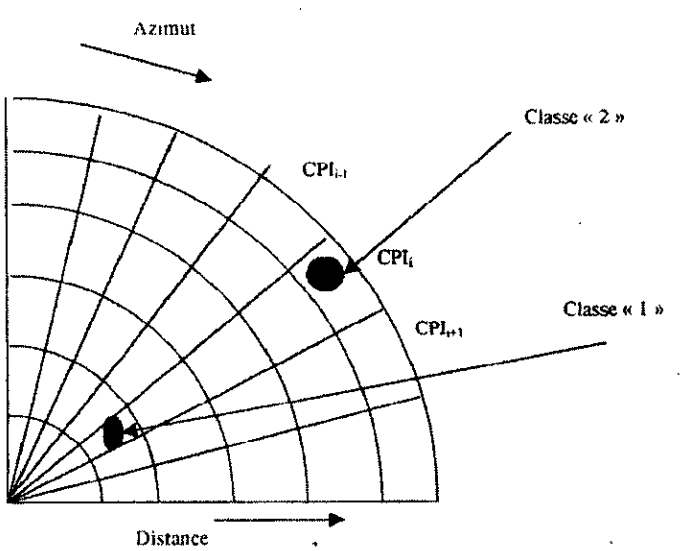



Figure 3.16: Absence de reports sur trois plots distance consécutive

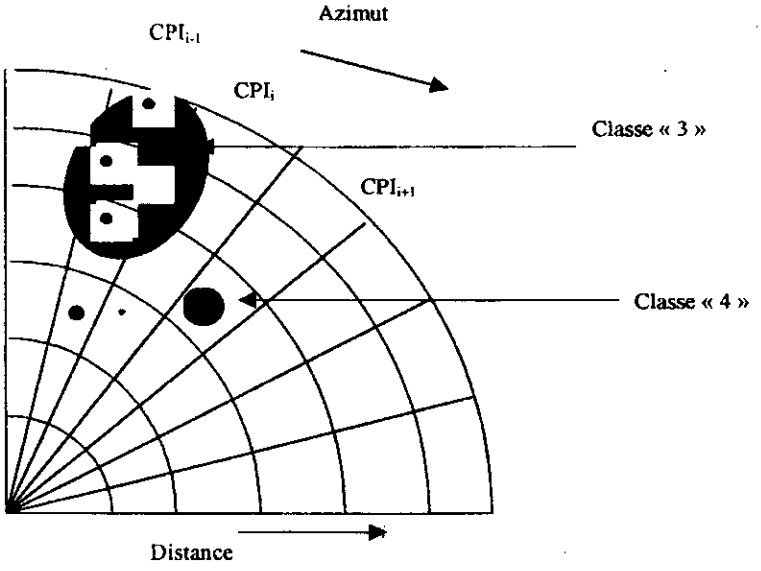



Figure 3.17: Absence de reports dans un CPI

### 3.3.3.2. Principe de calcul du centroïde :

Une classe est définie par des reportes adjacents sur trois portes distance et quatre ouvertures du lobe principal à -3dB (0-3dB) au maximum. Une porte distance est égale à cT/2 où c est la vitesse de la lumière et T est la durée de l'impulsion émise [37].

Cependant, la taille d'une classe peut être réduite s'il n'y a pas de report durant trois portes distance consécutives comme illustré dans la figure 3.15. Ainsi, nous fermons la classe « 1 » et tout nouveau report sera considéré dans la nouvelle classe « 2 ». Aussi, s'il n'y a pas de report azimut pendant un CPI, comme dans la figure 3.17, la classe existante (classe « 3 ») se ferme automatiquement et tout nouveau report sera considéré dans la nouvelle classe « 4 ». Une fois la classe (cluster) est fermée, ses reports subissent une opération de centrage (centoïde ou barycentre) pour donner un seul plot qui est représenté par sa distance, R et son azimut 8 à partir des expressions (3.1) et (3.2) respectivement.

$$R = \frac{\sum_{i=1}^{N} A_i D_i}{\sum_{i=1}^{N} A_i} \qquad (3.1)$$

et

$$\theta = \frac{\sum_{i=1}^{N} A_i \theta_i}{\sum_{i=1}^{N} A_i} \qquad (3.2)$$

Où A est l'amplitude du report, D; la distance, 8; l'azimut, et N le nombre de reports formant la classe.

Cet algorithme de la fusion peut être résumé comme indiqué sur l'organigramme de la figure.3.15.

### 3.3.3.3. Simulation :

Considérons un secteur saturé de 1000 cellules de distance et 256 cellules d'azimut. Le nombre de report générés, est de 256 000. Chaque report est compris dans une cellule de résolution de dimension d'une porte distance de 1/16 nmi (nautical miles : 1.8Km) et d'une ouverture de 1.40625°. Les 256 000 reports sont automatiquement répartis sous forme de classes (clusters). L'application de la règle de l'extraction est de l'ordre de 28444 plots.

Le second test a été réalisé pour un secteur alternativement saturé de 1000 cellules de distance et 256 cellules d'azimut 5 $CPI_j$ , saturé ; $CPI_{j+1}$ ne contient aucun plot, et ainsi de suite). Le nombre de reports générés, est de 128 000. A la sortie de l'extracteur le nombre de plots a été réduit à 42666 plots, ce qui est équivalent à un taux de réduction de nombre de plots d'environ 66%.

Nous observons clairement que l'extraction réduit sérieusement le nombre de reports qui risquent de saturer le processeur de poursuite. Le taux de réduction est lié à la disposition des reports dans l'espace de la couverture radar.

### 3.3.3.4. Conclusion :

L'extracteur ne répond pas à une étude exhaustive dans un premier temps, car chaque approche puise son efficacité à partir des données de l'environnement réel du site radar, qui permette de dresser un modèle approprié à cet environnement.

Cet algorithme spécifique, basé sur les deux paramètres de mesure qui sont la distance et l'azimut, permet d'apprécier l'apport de l'extracteur. L'application réelle de l'extracteur nous permet d'arrêter la règle de fusion la plus commode moyennant un compromis entre la densité du flux de données à l'entrée de l'unité de poursuite et la résolution radar.

Il est opportun au préalable d'élaborer un algorithme de fusion évolutif qui permet de vérifier les règles de fusion et d'apprécier l'apport à partir des données de synthèse. Ces règles servent de repère, car il est recommandé, avant de les adopter définitivement, de.disposer d'un fichier de données réelles, et d'essayer plusieurs règles selon des critères plus ou moins sévères pour adopter une seule.

# Chapitre 4

# Implantation du système d'extraction

## 4.1. Introduction:

Les techniques modernes d'extraction de plots utilisées dans la plupart des radars actuels ont une importance fondamentale dans les applications DSP.

Ces applications exigent de plus en plus des temps de calculs élevés pour que ces applications fonctionnent en temps réel. L'implémentation d'un algorithme d'extraction dépend des paramètres suivants :

- Les performances du DSP utilisé.
- Les caractéristiques du radar.

Le problème posé est donc, quel est le DSP capable d'effectuer l'opération d'extraction pour un radar avec des caractéristiques données.

Les performances d'un algorithme dépendent de la taille du programme d'une part, et du cycle du processeur utilisé d'autre part. Le temps d'exécution du programme donne une idée sur l'efficacité de l'algorithme et la possibilité d'implémenter en temps réel. Les contraintes de traitement en temps réel sont gérées de façon à satisfaire les exigences du radar ASRx du MTD, qui est un radar secondaire à longue portée (300Km). La largeur d'impulsion de 5ps, une ouverture du faisceau de 4 degrés, une vitesse de rotation de 3 et 6tours/min et une fréquence de répétition de 250 à 400Hz [5][6].

Comme la durée d'un balayage du ASRx du MTD est de 20s, et une précision de calcul en azimut 1.5° nous avons 256 cellules azimutales, donc le temps d'une cellule azimutale sera de 78.125ms qui correspond à 1.953.125 cycles sur un TMS320C40. Cet intervalle correspond au temps maximal tolérable pour effectuer le traitement en temps réel.

L'environnement du traitement s'effectue autour d'un processeur de traitement du signal en l'occurrence le TMS320C40 de Texas Instruments. Ce processeur est dédié à exécuter les opérations de synchronisation, d'acquisition, d'intégration, de validation, d'extraction et de communication conformément à un protocole bien défini.
La figure.4.1 montre ces différentes étapes de traitement pour le cas du programme de l'application IFF d'extraction de plots à l'intérieur du DSP. Pour la visualisation et l'exploitation de ces données un processeur Pentium II 200 lui est dédié.

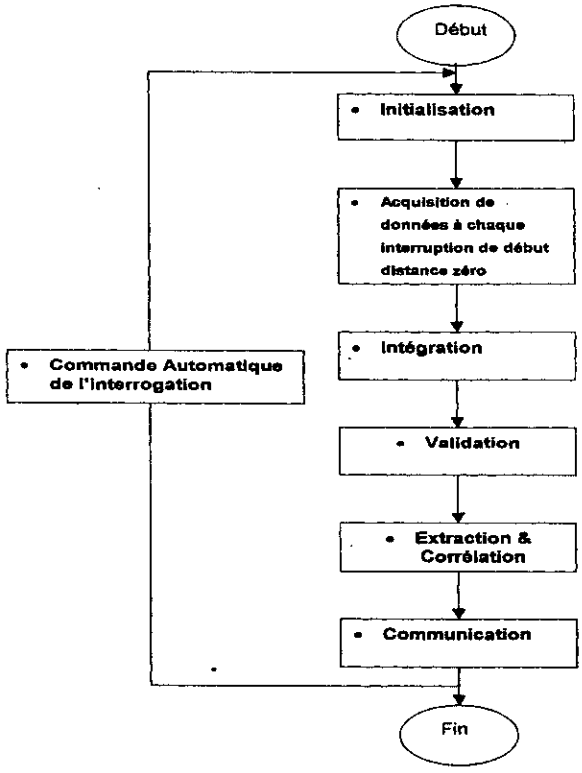



Figure 4.1: Organisation des étapes de traitement

## 4.2. Application IFF :

### 4.2.1. Synchronisation :

Le bloc de synchronisation de la station radar génère le signal de synchronisation PRFs et la mise en forme des signaux ACPs et nord pour le fonctionnement de l'IFF. Pour implanter l'algorithme d'extraction sur DSP, plusieurs paramètres doivent être pris en considération, à savoir :

- La période de répétition T, ou le signal début d'acquisition PRF (Pulse Repetition Frequency) ;
- La largeur d'impulsion $\zeta$ ;
- Les signaux générés par l'antenne correspondant à des cellules azimut particulière, le signal début / fin d'une résolution angulaire, CPI ( Coherent Pulse Interval) et le signal nord ( La référence azimut zéro ) ;
- La durée du balayage d'antenne ( durée d'un scan d'antenne).

***a) Le signal début d'acquisition « la période de répétition radar » :*** ce signal attaque la pin ***IIOFO*** du processeur. A chaque présence de l'impulsion PFR, l'interruption correspondante est validée, et permet d'effectuer les opération suivante :

- ❖ Incrémenter le compteur des PRFs par CPI;
- ❖ Relancer le timer 0 pour synchroniser l'échantillonnage avec la PRF.
- ❖ Charger l'adresse du buffer des réponse !FF.

- Autorisation de l'interruption 110F 3.

***b) Le signal « Début /Fin » d'un CPI :***

L'impulsion CPI apparaît toutes les 16 ACPs station radar solidaire à l'antenne, et correspond à la pin IIOF1 du processeur DSP.

A chaque présence du signal CPI, l'interruption c_int56 sera servie et consiste à :

- Remettre à zéro le compteur de PRF,
- Mettre le flag « new_cpi » à 1 pour autoriser le traitement dans le programme principal les données du nouveau CPI.

***c) Le signal nord :***

Indique le passage du nord par la référence azimut zéro et attaque la pin **IIOF2** du processeur. A chaque présence du Nord, on effectue les opérations suivantes : Remise à zéro le compteur CPI.

***d) Le signal d'identification :***

Le signal d'identification représente les réponses impulsionnelles de l'aéronef ami interrogé. Ce signal attaque la pin **IIOF3,** à chaque impulsion les opérations suivantes sont effectuées :

- Lecture du DMA4counter utilisé ici comme le compteur de porte distance ou d'échantillonnage.
- Test de la zone morte de la station radar qui est de 3 km ; le paramètre zm est constant et il est égale à 10 ( zm set...10) ce qui donne :

  $10*2*10^{-6}*3*10^{\$} / 2 = 3000$ m.
- Stocker le numéro de la porte distance dans le buffer des réponses d'identification.

Ce qui suit est le code source de l'interruption IIOF3:

```
;-------------------------------------------------------------------------------------------------
;     c_int06  - Interrupt service routine of iiof3

_c_int06
        push   st
        push   r0
        push   r1
        pushf  r0
        pushf  r1
        push   ar0

        ldi    @dma4adr,ar0
        ldi    *+ar0(3),r0              ;lire le numéro de la porte
        ldi    buffrows,r1
        subi   r0,r1                    ;distance de la réponse

        cmpi       zm,r1                ;test de la zone morte
        ble        no_intr              ;

        ldi    @resp_ptr,ar0
        sti    r1,*ar0++(1)
        sti    ar0,@resp_ptr        ;Stocker le numéro la porte
;distance
no_intr
        pop    ar0
        popf   r1
```

```
popf   r0
pop    r1
pop    r0
pop    st
       retiu
;-------------------------------------------------------------------------------
```

### 4.2.2. Installation des interruptions :

La figure 4.2 illustre l'installation des interruptions externes IIOFO, IIOFI, IIOF2 et IIOF3 dans la carte DSP TMS320C40 [10].

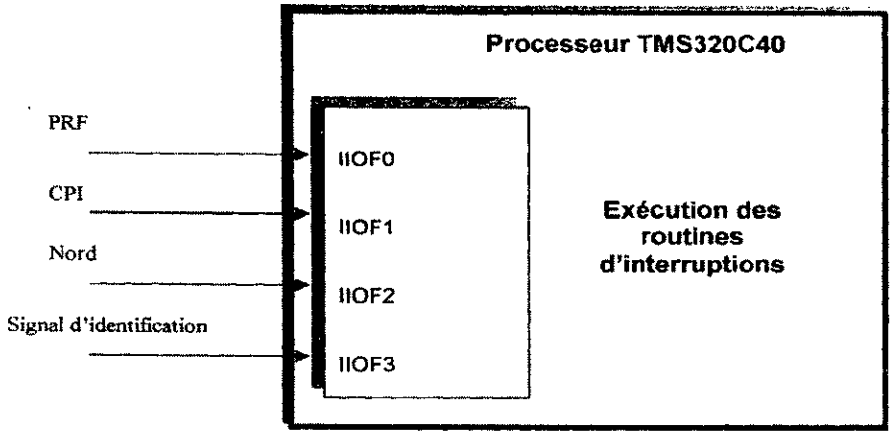



Figure 4.2: Correspondance entre les signaux externes et les pins d'interruption du DSP TMS320C40

La section du code source suivante écrit avec le compilateur C de Texas Instruments montre les instructions et l'installation des interruptions correspondantes aux pins II0FO...3 configurées en détection de front ( Edge Trigger Interrupt ), et celle de DMA1 et DMA4.

```
/*-------------------------------------------------------------*/
/* Define the external assembler interrupt.                    */
void c_int03(void);          /* Interruption PRF.              */
void c_int04(void);          /* Interruption NORD.             */
void c_int05(void);          /* Interruption CPI.              */
void c_int06(void);          /* Interruption Signal IFF.       */
void c_int38(void);          /* Interruption DMA1.             */
void c_int41(void);          /* Interruption DMA4.             */

/*-------------------------------------------------------------*/
/* initilize processor Interrupt registers                     */

void initInts(void)
{
      set_ivtp(DEFAULT);                    /*set IVTP         */

      install_int_vector(&c_int03,3);     /*install iiof0 int  */
      set_iiof(0,0x9);
```

```
      install_int_vector(&c_int04,5);      /*install iiof1 int  */
       set_iiof(2,0x9);

      install_int_vector(&c_int05,4);      /*install iiof2 int  */
       set_iiof(1,0x9);                    /*enable the nord int*/

      install_int_vector(&c_int06,6);    /*install iiof3 int  */
      set_iiof(3,0x9);

       install_int_vector(&c_int38,38); /*install dma int1     */
       install_int_vector(&c_int41,41); /*install dma int4     */

       INT_ENABLE();                    /*Enable GIE bit       */
}
/*-------------------------------------------------------------*/
```

L'instruction ***install int vector(&c int03,3);*** signifie qu' à chaque présence d'interruption au niveau de la pin IIOFO du DSP la routine ***c_int03*** sera exécutée ; la même chose s'applique pour les pins IIOF1,IIOF2 et IIOF3 qui correspondent aux routines c int04, c_intO5 et c_intO6 respectivement.

L'instruction ***install int vector(&c_int38,38);*** permet d'effectuer l' opération d'integration à chaque présence d'interruption DMA4.

L'instruction ***INT ENABLE();*** permet l'autorisation des interruptions installées ( Global Interrupt Enable ).

### 4.2.3. Acquisition :

L'acquisition des réponses s'effectue selon l'organigramme de la figure .4.3, et consiste à lire le compteur du co-processeur DMA4 qui est synchronisé avec le TimerO ; le compteur du co-processeur DMA4 se décremente à chaque interruption interne du TimerO, car le TimerO du DSP TMS320C40 est programmé pour fonctionner en arrière plan comme échantillonneur. Le mot de contrôle nécessaire pour le fonctionnement du co-processeur est donné dans le code source suivant :

```
; préparation du dma canal #4. Echantillonnage en distance.; dma4 set up

dma4adr          .word 01000eOh              ;DMA4 Register address
ctlbufl          .word dma4 func
src4             .word compO                 ;inds4
                .word     Oh
count4           .word buffrows              ;number of pulses
dst4             .word buffer0
indd4            .word    1h
lnk4             .word ctlbuf2
ctlbuf2          .word dma4 func
                .word compi
                . word Oh
                . word buffrows ;number of pulses
                . word bufferl ;address of ad2detl
                . word Ih
                . word ctlbufl
-------------------------------------------------------------
```

Le compteur du TIMERO est initialisé à 12 pour générer une interruption chaque 2 ps qui correspond à la période d'échantillonnage, cette interruption décrémente le compteur du co-processeur DMA4 comme le montre le code source suivant :

```
die_mask4          . word 0600000h ;
die mask to enable syncread to timer0
        ldi        @die mask4, r0
        or         r0, die
; enable timer0 sync
```

Ce mot du ***DMA Coprocessor Enable Register*** (DIE) permet de synchroniser le DMA4 **en *Unified Mode*** avec l'interruption TIMERO.

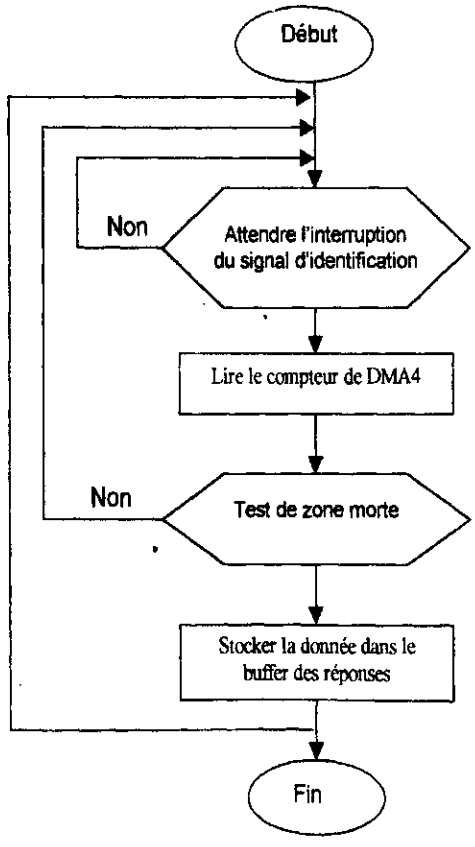



Figure 4.3:Organigramme d'acquisition

Ainsi à chaque interruption IIOF3 le contenu du compteur est stocké dans le buffer des réponses est quand le compteur de DMA4 devient nul l'autre interruption c int41 est générée et elle correspond à la portée maximale du radar et du traitement au niveau de l'application IFF ; le code source suivant donne une idée de cette installation :

```
buffrows        .set 1045;
```

La résolution en distance 2 ps = pour 1045 portes on aura 2090 lis.

2090 2090*10-6*3*108/2 = 313500 mou 313Km.

La portée du radar est de 300KM ce qui est suffisant, pour les 13500m en plus pour des raison de sécurité dans les conditions limites.

```
#define timer0 R          12 // programme header VAR.H
timOprd=timerO R;                  //le programme principal ALL. C
```

Cette variable est déclarée avec le compilateur C et le programme assembleur fait appelle à cette variable en la déclarant globale « *.global _timOprd* ».

### 4.2.4. Intégration :

L'intégration se fait selon l'organigramme de la figure.4.4. Les réponses dans le buffer ifresp sont lues puis placées dans la cellule distance dont le numéro est la valeur de la case mémoire du buffer ifresp lui même. Les réponses dans les buffers bufferO ou bufferl sont compressées en distance de 5 cellules.

A leurs tours les portes distance de chaque CPI sont intégrées en azimut comme le montre le code source suivant :

```
;Emplacement des réponse
                ldi     @resp adr, art
near            ldi     *art ir0
                cmpi O,irO
```

| | |
|---|---|
| bz | far |
| sti | 1, *+ar0(ir0) |
| sti | 0, *art ++(1) |
| b | near |

```
;Compression et intégration des réponses compre     ldi     Oh,r1
                addi 01h,irO
                cmpi cprows,ir0
                beq libre
                ldi     04,rc
                rptbd b_integ
                nop nop nop
                ldi     *ar0++(1),r0
```

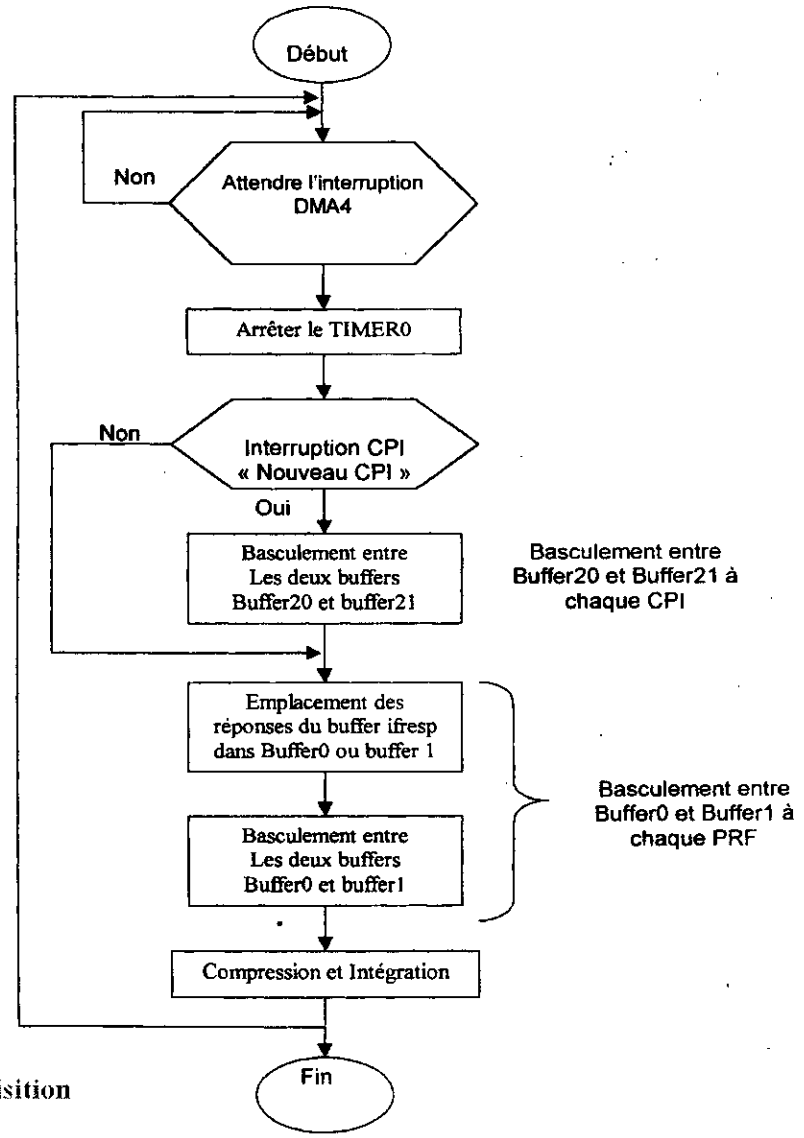



**Figure 4.4 : Organigramme d'acquisition**

Afin d'éviter l'écrasement des échantillons de deux récurrences consécutives, nous avons opté pour la technique dite « Double Buffering Technique» pour les buffers bufferO , buffer1 et buffer20, buffer2l qui consiste à utiliser deux buffers avec un basculement d'une récurrence à la suivante de manière automatique comme le montre la figure 4.5.

```
;Double Buffering Technique entre Buffer2O et Buffer2l pour chaque CPI.
        Idi @ cf)
        flag, rO cmpi Oh,rO
        Idiz @buf20adr, ar2              ; chargement de l'adresse de buffer20
        Idip @buf21 adr, ar2             ; chargement de l'adresse de buffer2l
;Double Buffering Technique entre BufferO et Bufferl pour chaque PRF.
        Idi         @buf flag,rO
        cmpi        O,rO
        Idiz        @bufO adr, arO       ; chargement de l'adresse de bufferO
        Idip        @buf1_adr,arO        ; chargement de l'adresse de buffer1
        Idi         @buf flag,rO
        xor         Ol h, rO             ;flag de basculement
        sti         rO, @buf flag
```

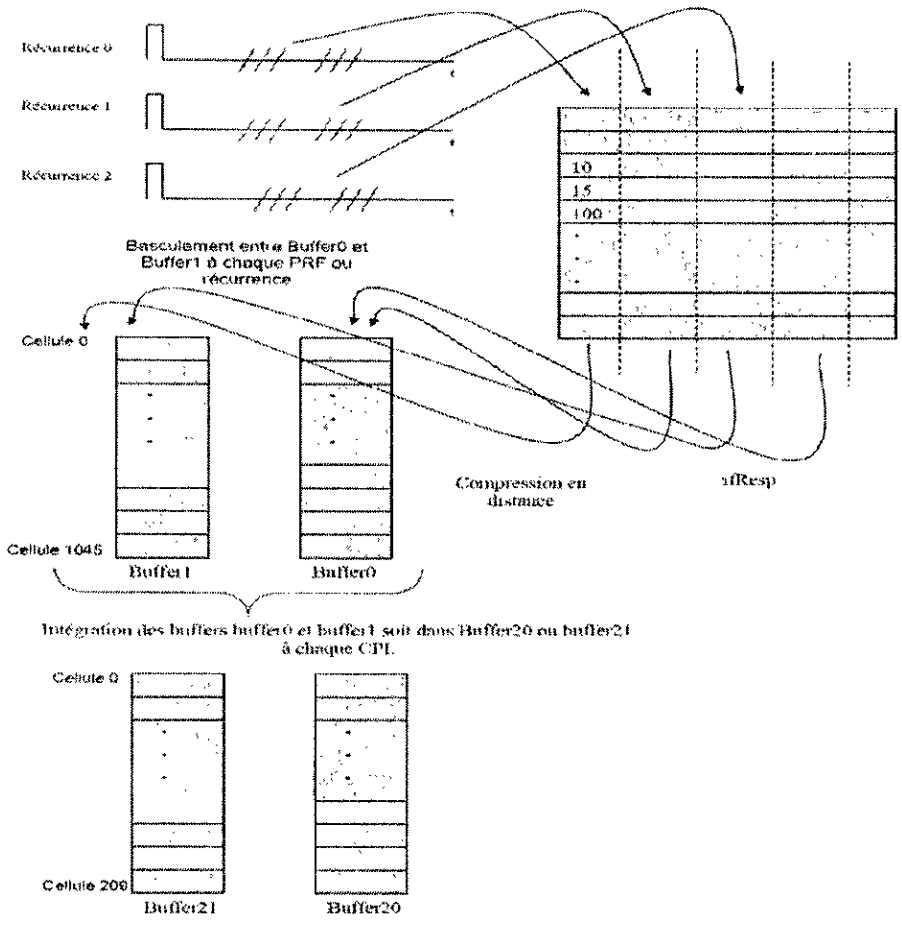



**Figure 4.5: Organisation des buffers de l'intégration**

### 4.2.5. Validation:

La fonction de validation se fait selon l'organigramme de la figure.4.6. Après intégration des échantillons du CPI courant , le CPU commence le traitement par la

validation des réponses de chaque cellule distance. Les cellules distance du CPI courant sont parcourues puis comparées à un seuil calculé par rapport au nombre de PRF par le CPI courant.

Le test de validation se fait pour chaque cellule, et s'il y a validation ce qui veut dire présence de cible amie, on fait automatiquement appelle à la fonction de conversion des coordonnées polaires vers les coordonnées cartésiennes et le résultat est stocké dans le buffer de communication Le code suivant donne une idée de la façon dont le programme écrit avec le compilateur C de Texas Instruments pour appeler une fonction écrite en assembleur :

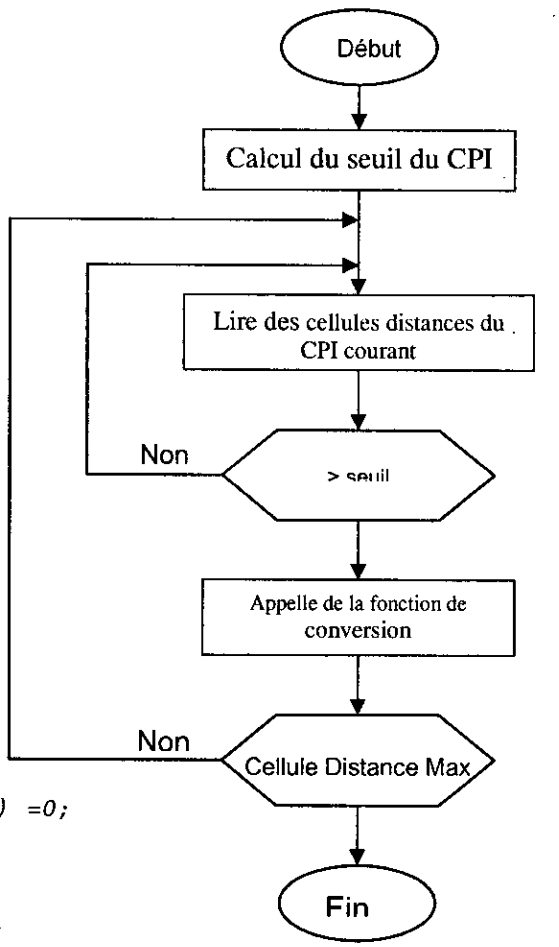



Figure 4.6: Organigramme de validation

```
for (i=2; i<sizebuf; i++)
      {
      if (* (buf ptr+i) <seuil) * (buf ptr+i) =0;
      else {
                  * (buf ptr+i)=1;
                  fanc= (float) anc;
                  fi= ( (float) i) +depliff;
                  if (fi < 0. 0) fi=0. 0;
                  Radius= fi*del taD;
                  Angle= fanc ;
                  asm ("    ldf     @Radius, r0     ") ;
                  asm ("    ldf     @Angle, r1          ")
                  ; roth2xy 0 ;
                  asm ("    stf     r0, @_x ") ;
                  asm ("    stf     ri, @ y ") ;
                  nbc++;
                  mvt [nbc] [0]= (int) x;
                  mvt [nbc] [1]= (int) y;
                  }
      }
```

Les coordonnées de ces cibles sont envoyées vers le host pour être directement visualiser comme une vidéo brute de l'IFF sans pour autant faire l'extraction.

*La fonction de conversion :*

La conversion des coordonnées polaires vers les coordonnées cartésiennes se fait dans la carte DSP TMS320C40 vu le temps de calcul important qui est de 50 cycles comme le montre le code source de la fonction :

---

```
 ; Conversion from polar coordinates to cartesian Coordinates
   file name : roth2xy.asm

 ; Input:            R0 -> Radius
R1 -> Angle  ( 0 <= Angle <= 360 degrees)
; Output:        R0 <- Radius*cos (angle)
;                 R1 <- Radius*sin (angle)
;
; Registers altered and not restored : st, ir0, r9, r10
;
; Calling sequence :     laju  roth2xy
;                        ldf   r0,@angle
;                        ldf   r1,@radius
;                        nop
;-------------------------------------------------------------------------------
-----------------------------------------
            .global _roth2xy
      .text
_roth2xy
      push    st
      push    r0
      push    r1
      push    r9
      push    r10
      push    r11
      pushf   r0
      pushf   r1
      pushf   r9
      pushf   r10
      pushf   r11
      push    ir0
      push    ar0                ;
      ldf     1,r9               ; Sign of sin(x)
      ldi     @sinad, ar0        ;
      cmpf    180.0,r1           ;
      ble     index              ;
      ldf     -1,r9              ; change sign
      subf    180.0,r1           ;
index   mpyf    2.0,r1             ; Index
      ldf     r1,r10             ;
      addf    0.5,r10            ;
      cmpf    r10,r1             ;
      ble     low                ;
      addf    1.0,r1             ;
low   fix     r1,ir0             ;
      mpyf    *+ar0(ir0),r9,r1   ;
      mpyf    r0,r1,r1           ; r1=radius*sin(angle)
      ldi     @cosad, ar0        ;
      bud     r11                ; delayed return
      mpyf    *+ar0(ir0),r9,r9   ; r9=cos(angle)
      mpyf    r0,r9,r0           ; r0=radius*cos(angle)
      pop     ar0                ;
      pop     ir0
      popf    r11
      popf    r10
      popf    r9
      popf    r1
      popf    r0
      pop     r11
      pop     r10
```

```
        pop     r9
        pop     r1
        pop     r0
        pop     st


    .data
sinad    .word sintab
cosad    .word costab
sintab   ; sin(x);x¯0..180, deltax=0.5
        .float 0.0000, 0.0087, 0.0175,  0.0262, 0.0349,0.0436
        .float 0.0523, 0.0610, 0.0698,  0.0785, 0.0872,0.0958
        .float 0.1045, 0.1132, 0.1219,  0.1305, 0.1392,0.1478
costab ; cos(x);x¯0..180, deltax=0.5


        .float  1.0000, 1.0000, 0.9998, 0.9997, 0.9994, 0.9990
        .float  0.9986, 0.9981, 0.9976, 0.9969, 0.9962, 0.9954
        .float  0.9945, 0.9936, 0.9925, 0.9914, 0.9903, 0.9890
        .float  0.9877, 0.9863, 0.9848, 0.9833, 0.9816, 0.9799
        .float  0.9781, 0.9763, 0.9744, 0.9724, 0.9703, 0.9681
```

Les tables sinus et cosinus sont calculées entre 0° et 180° avec un pas de 0.5°.

## 4.2.6. Extraction et Corrélation :

### 4.2.6.1. Organisation mémoire :

Après validation des cibles, les données sont sauvegardées dans un buffer de sortie selon le format présenté dans la figure 4.7.

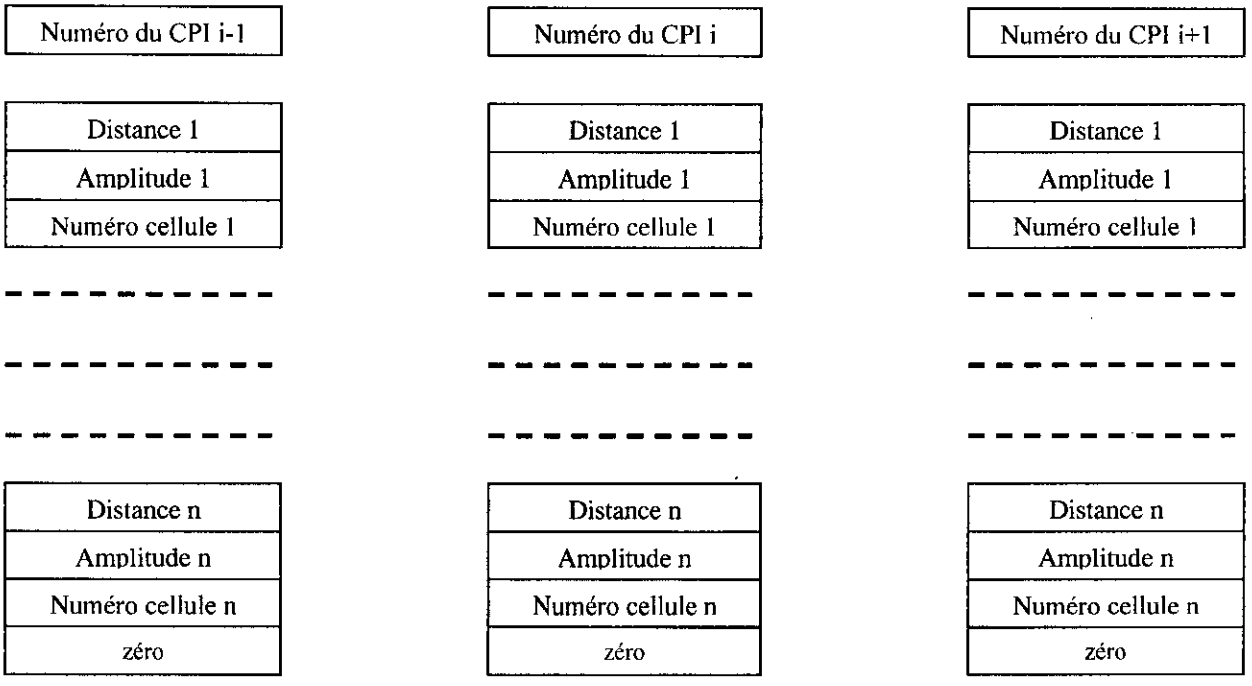



**Figure 4.7: Organisation des buffers de sortie du la validation**

Le résultat de validation de chaque CPI est sauvegardé selon le format présenté dans la figure 4.8. il comprend, le numéro de CPI, la distance, l'amplitude qui présente le nombre de réponse dans la cellule, et le numéro de la cellule distance et un zéro dans la dernier case mémoire (après le dernier plot validé) pour indiquer la fin du CPI.

### 4.2.6.2. Implantation :

L'implantation se fait selon l'organigramme de la figure 4.8. Après fusion des classes, on effectue le centrage qui consiste à calculer le centre de gravité du plot selon les équations (3.1) et (3.2) du chapitre 3.

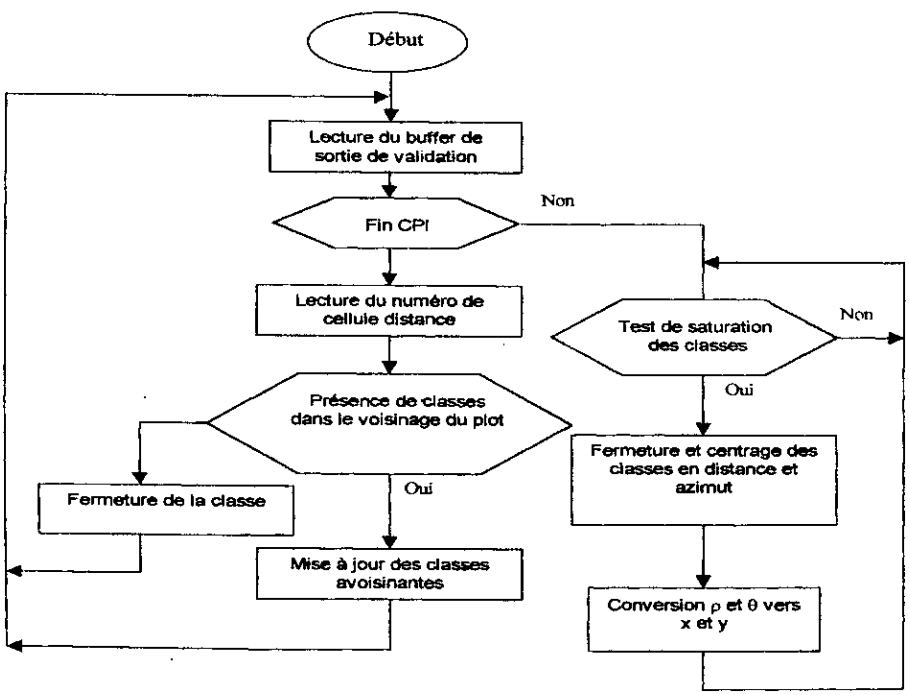



Figure 4.8: Organigramme de la règle de fusion

La section du code suivant implante la règle de fusion pour l'association des plots aux classes.

```
/* lecture des plots du signal iff.                          */
     if (*(buf_ptr+i) != 0)
                {
/* Test d'existance d'une classe à la porte distance i.      */
          if (pp[i] < 0)
/* Ouverture d'une nouvelle classe.                          */
                {
                pp[i]=point[ind_mp];
                mp[pp[i]][0]=anc;mp[pp[i]][1]=1;
                point[ind_mp]=-1;ind_mp++;}
          else
                { mp[pp[i]][1]++;
/* Test de saturation de la classe.                          */
                if(mp[pp[i]][1]==ouvt_int)
                      {theta=mp[pp[i]][0]+(mp[pp[i]][1]-1)/2;
```

```
                        if(theta>255)theta=theta-256;
                        mp[pp[i]][0]=mp[pp[i]][1]=0;
                        ind_mp--;point[ind_mp]=pp[i];pp[i]=-1;
/* Calcul du centroïde de la classe.                                    */
                        ftheta=(float) theta;
                        fi=((float) i)+depliff;if(fi < 0.0)fi=0.0;
                                y=fi*deltaD*sin(deltaA*ftheta);
                                x=fi*deltaD*cos(deltaA*ftheta);
                                corp++;
                                mtt[corp][0]=(int) x;
                                mtt[corp][1]=(int) y;}
                    }
                }
/* Test dans le cas où la classe ne reçoit pas de plot dans un cpi.    */
        else    {
                if(pp[i] >= 0 )
                {theta=mp[pp[i]][0]+(mp[pp[i]][1]-1)/2;
                if(theta>255)theta=theta-256;
                mp[pp[i]][0]=mp[pp[i]][1]=0;
                ind_mp--;point[ind_mp]=pp[i];pp[i]=-1;
/* Calcul du centroïde de la classe.                                    */
                ftheta=(float) theta;
                fi=((float) i)+depliff;if(fi < 0.0)fi=0.0;
                    y=fi*deltaD*sin(deltaA*ftheta);
                    x=fi*deltaD*cos(deltaA*ftheta);
                    corp++;
                    mtt[corp][0]=(int) x;
                    mtt[corp][1]=(int) y;}
                }
        }
}
```

### 4.2.7. Communication :

La communication se fait à travers le port de communication n°1 et beaucoup de données doivent circuler entre la carte DSP TMS320C40 et le processeur Pentium II via le VMEbus.

Ainsi la programmation du DMA1 avec le port de communication n°1 doit se faire en split mode pour la communication dans les deux sens en même temps à la place de unified mode qui permet la circulation dans un seul sens en même moment. Le code de la routine de communication est en assembleur.

```
;-------------------------------------------------------------------------------------------------
DMA1ADR         .word  001000b0h    ; DMA ch 1 register address to be synchronised
DMAFUNC1        .word  dma_func1    ; ONLY to port1 interrupt lines
dma1_pr_st  .word  0c00000h
dma1_aux_st     .word  03000000h

ctl_buf11   .word  dma_func1
SRC1        .word  _mvt
SRCINDX1        .word  1h
COUNT1      .word  0
LINK1           .word  ctl_buf11
```

```
ctl_aux11       .word  dma_func1
DEST1           .word  _m
DESTIDX1                .word  1h
AUXCNT          .word  nc
AUXLNK1         .word  ctl_aux11
;-------------------------------------------------------------------------------------------------
```

La routine d'initialisation du DMA1 est se fait avec dma1_set() dans le programme principal ALL.C:

```
/*-------------------------------------------------------------------------------------------------*/
/* Initialisation du dma1 pour la communication en split mode                                      */
/* Lecture et l'envoi  des données avec l'unité de visualisation.                                  */

dma1_set();
/*-------------------------------------------------------------------------------------------------*/
```

Les données circulant entre le Host et la carte DSP TMS320C44 sont de types et sont constituées de deux messages ; le message des résultats d'extraction dont le protocole est donné dans le tableau 7.1, ce message contient en premier lieu le code du message qui est celui de la carte DSP IFF « 0xbbbb » qui sert à faire la différence entre les données qui peuvent provenir soit de la carte de détection « 0xaaaa » ou la carte réponse active « 0xdddd » ou des carte de communication pour le système au complet, puis le code de type de données à lire ; données pour visualisation seulement ou données destinées pour l'unité de poursuite, le message de demande d'interrogation ; ce message provient de l'unité de poursuite vers la carte DSP IFF dont le protocole est montré dans le tableau 7.2, le code du message est dans la première case est « 0x7fff » , la deuxième case mémoire est le mode de fonctionnement du radar, la troisième case est le mode d'interrogation avec deux autre case mémoire pour le début et la fin d'interrogation, deux cases réservées pour un éventuel rajout de fonctions et enfin les numéros de CPI des cibles à interrogées.

La communication des données **DSP→ VME** se fait à chaque CPI par le code source suivant qui se trouve dans le fichier COMM.H:

```
dma1_pr_go();   /* Ordre d'envoi des données vers le Host. */
```

Alors que la communication **VME→ DSP** se fait par le bouton de l'interface graphique de l'application de visualisation SendIff après sélection de ces cibles, comme le montre le code source suivant :

```
for (int i=0;i<SizeMessage;i++) Controle->Send_Iff_Message((DWORD)Message[i];
```

| Tableau 4.1. Protocole de communication des données de la carte DSP TMS320C40 vers l'unité de visualisation | | | |
|---|---|---|---|
| code message iff | mvt[O][0] | Oxbbbb | |
| Mode | mvt[0][1] | Ox7ccc IFF<br>Ox7eee Message unité de poursuite | |
| N° CPI | mvt[1j[O] - | Numéro du CPI courant | |
| Nombre de Plots | mvt[1][1] | Nombre de plots dans le CPI courant | Plots destinés pour la visualisation |
| XO | mvt[2][0] | | |
| YO | mvt[2][1] | | |
| **X1** | mvt[3][0] | | |
| Y1 | mvt[3j[1 ] | | |
| X2 | mvt[4][0] | | |
| Y2 | mvt[4][ 1 ] | | |
| | | | |
| | | | |
| | | | |
| N° CPI début intrg | mvt[nbc+1][0] | | Plot destinés pour l'unité de poursuit (ces plots seront communiqués via un support téléphonique vers l'unité de poursuite |
| Nombre de Plots | mvt[nbc+1][1j | | |
| **XO** | mvt[1+nbc+1][0] | | |
| YO | mvt[1 +nbc+1 ][1 ] | | |

| **Tableau 4.2. Protocole de communication des données de l'unité visualisation vers la carte DSP TMS320C40** | | | |
|---|---|---|---|
| code message iff | m[0] | Ox7fff | |
| Mode | m[1] | Ox2fff<br>Ox3fff<br>0x0000 | mode RARE<br>mode fréquent<br>pas de changement |
| Interrog continue | m[2] | Ox2aaa<br><br>0x1 aaa | Demende interrogation continue.<br>Demande arret interrogation continue. |
| CPI début interrog | | | |
| CPI fin interrog | | | |
| Réserve | | | |
| Réserve | m[6] | | |
| Nombre de N° de CPI | m[7] | | |
| CPIO | m[8] | N° de CPI des cibles ou direction à interroger | |
| CPII | m[9] | | |
| CPI2 | m[10] | | |
| | | | |
| | | | |
| | | | |

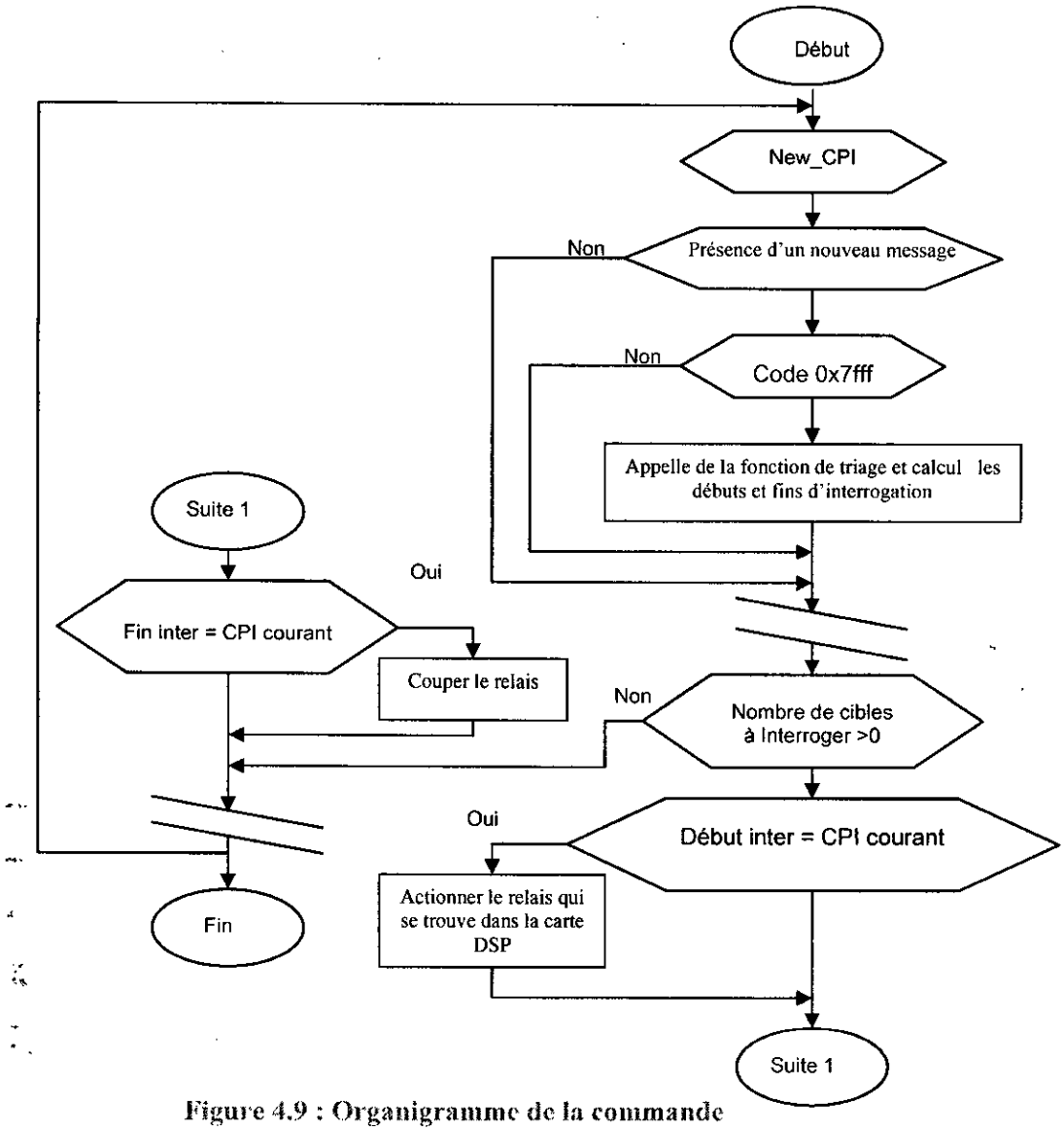



**Figure 4.9 : Organigramme de la commande automatique de l'interrogation**

Le code source suivant qui se trouve dans le programme ALL.0 écrit avec le compilateur C montre la commande du relais de la carte DSP TMS320C40 :

Fermeture du relais :

```
if(ancc == b){dac=1;if flag next=1;n dac=1; /* storage the converter value
asm(" push rO          ');
asm(" push arO          ');
asm(" ldi @_dac adr, ar0 ');
asm(" ldi @_dac,r0          ');
asm(" sti rO, *ar0
asm(" pop arO ');
asm(" pop r0 ');)
```

Ouverture du relais:

```
if(dac==1)(if(f==ancc){rf(al dac==0){dac=0;if flag next=0;) 1++;n dac=0;
/* storage the converter value
    asm(" push r0          ');
    asm(" push ar0          ");
```

```
asm(" ldi @_dacadr, ar0 ');
asm(" ldi a_dac,r0              ');
asm(" sti r0, *ar0 ');
asm(" pop arO '9;
asm(" pop r0 ")
```

### 4.2.8. Temps-Réel:

Dans la plupart des applications DSP, le traitement en temps réel est très critique. Des choix adéquats doivent être effectués pour la sélection d'un DSP capable d'exécuter la tâche en temps réel. Par exemple, dans une application de traitement de la parole, le taux d'échantillonnage est de 8 KHz, qui correspond à un intervalle de 125ps entre deux échantillons consécutifs. Cet intervalle est le temps tolérable maximal pour une opération temps réel, correspondant à 3125 cycles sur un TMS320C40. Dans le but d'exécuter les tâches de traitement du signal dans l'intervalle indiqué, il est nécessaire de réduire le temps d'exécution. Pour ce faire, il est recommandé d'exploiter toutes les facilités offertes par l'architecture et les instructions du processeur utilisé.

## 4.3. La visualisation:

### 4.3.1. Introduction :

L'application de visualisation appelée **AllouVisu** a pour mission la visualisation des résultats de l'extraction de plots du radar secondaire. Dans chapitre on retrouvera les raisons du choix du langage de programmation qui est le Borland C++Builder ainsi que les différentes tâches de l'application, entre autre; chargement du programme DSP dans la carte DSP TMS320C40, l'interface graphique, l'affichage des données ainsi que les paramètres voulus pour exploitation ou les tests, la lecture des données envoyées par la carte DSP, l'interrogation automatique et l'écriture dans la carte de DSP etc.

### 4.3.2. Langage de programmation :

Dans cette section nous allons donner un aperçu sur quelques langages qui utilisent l'approche de programmation classique événementielle, ainsi qu'une représentation détaillée du langage de programmation utilisé.

#### 4.3.2.1. La programmation classique :

Ce type de programmation exige une structuration bien définie du code, c'est à dire que le programme a une certaine structure spécifiée par le langage de programmation que le programmeur doit respecter [38] [39] .

Les langages de programmation qui adoptent cette approche sont connus sous le nom de langages classiques ou procéduraux. Ces langages ne contiennent pas d'outils d'aide à la conception d'applications. De plus, l'exécution des programmes se fait de haut vers le bas d'une manière ordonnée. Le cheminement de l'algorithme peut varier suivant les données transmises au programme ou suivant certaines contraintes spécifiées par l'environnement,

mais essentiellement il reste prévisible. On peut citer comme exemple de ce type de langages : « le C, le Pascal, et le Basic ».

#### 4.3.2.1. La programmation événementielle :

Ce mode de programmation est piloté par les événements ; les applications répandent à ces événements en traitant les messages émis par le système d'exploitation. En effet, c'est l'ensemble des actions de l'utilisateur sur un objet qui déclenchent l'exécution d'un ensemble d'actions de procédures. L'enchaînement des procédures est donc imprévisible, contrairement au mode de programmation procédurale classique qui spécifie l'ordre de leurs exécutions.

Les langages de programmation qui adoptent cette approche sont actuellement les plus utilisés, car ils fournissent un grand nombre d'outils d'aide à la conception ; ils se distinguent par leurs variétés de performance dans un domaine à un autre ; certains sont très performants dans des applications systèmes alors qu'ils le sont moins dans les applications de gestion comme le Visual C++. D'autre sont orientés plus vers la gestion, comme le Delphi et le Windev, alors que Java est orienté essentiellement vers les applications réseaux.

En fin le langage Borland C++ Builder, avec lequel nous allons implanter notre application, est adapté presque à toutes sortes d'applications, grâce à sa simplicité, sa souplesse ainsi que sa force d'expression.

### 4.3.3. Présentation de Borland C++ Builder :

La programmation avec le Borland C++ Builder se fait en trois étapes indissociables et complémentaires. Une étape de conception visuelle et une étape d'écriture du code [40].

L'étape de conception signifie que la conception de l'interface de l'application à développer se fait graphiquement, en faisant glisser à l'aide de la souris ou du clavier les contrôles nécessaires à l'application ( Bolton, Edit, EditRich, Label, ListBox, ...) ; C++ Builder permet cette performance sans pourtant sacrifier la vitesse d'exécution, puisque on dispose toujours de la puissance du langage C++.

L'étape d'écriture du code consiste à choisir un événement applicable à un contrôle spécifique et de lui associer le code approprié ; par exemple on associe un code à l'événement « OnClick » du contrôle « Button ».

L'environnement de Développement Intégré C++ Builder, se compose en trois parties :

- La fenêtre supérieure : qui est considérée comme la fenêtre principal. Elle contient la Barre d'outils et la palette des composants.
- L'inspecteur d'obief : avec lequel on modifie les propriétés et les événements d'un composant.

L'espace de travail : qui affiche à l'origine le Concepteur de fiche (concevoir une fenêtre ou une boite de dialogue par exemple). Derrière le Concepteur de fiche on trouve l'Editeur de code où va être tapé le texte lors de l'écriture des programmes.

L'inspecteur d'objet, le Concepteur de fiche, l'Editeur de code et la palette des composants fonctionnent sont interactifs lorsqu'on construit les applications. La figure suivante donne un aperçu de l'EDI de C++ Builder :

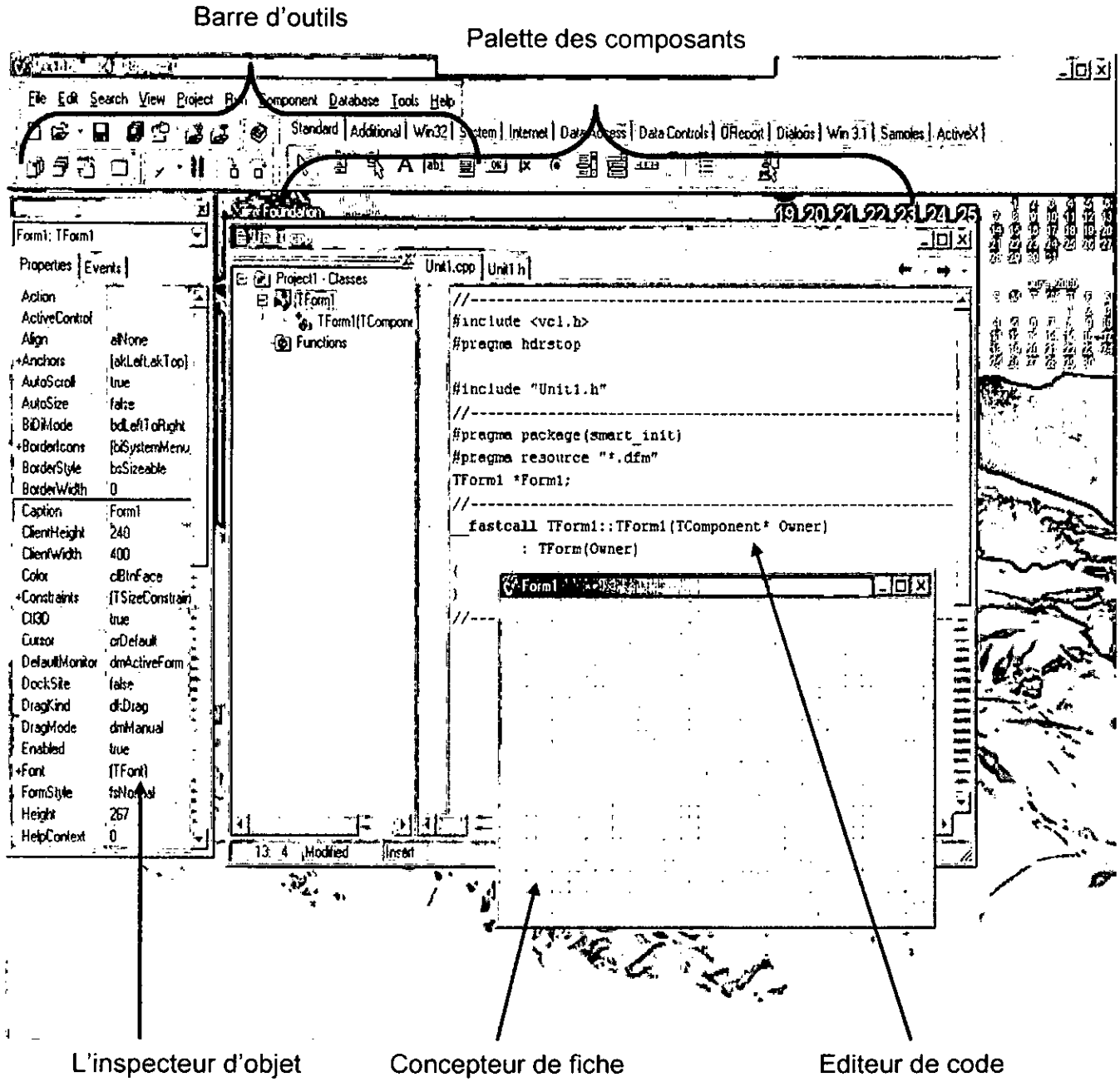



**Figure 4.10 : Environnement de Développement Intégré C++ Builder**

4.3.4. Application de visualisation:

L'application visualisation écrite avec le Borland C++Builder 4 a pour tâche principales la lecture des données envoyées par la carte DSP via le VMEbus pour éviter toute perte d'information et qui dit information dit coordonnées de cibles aériennes soient civile, militaire amie ou ennemie et visualiser ces dernières pour l'exploitation.

Une deuxième tâche de l'application consiste à l'interrogation automatique des cibles par la sélection de ces dernières avec la souris et envoyer les coordonnées de ces cibles via le VMEbus vers la carte DSP. La figure 4.11 montre l'organigramme de l'application.

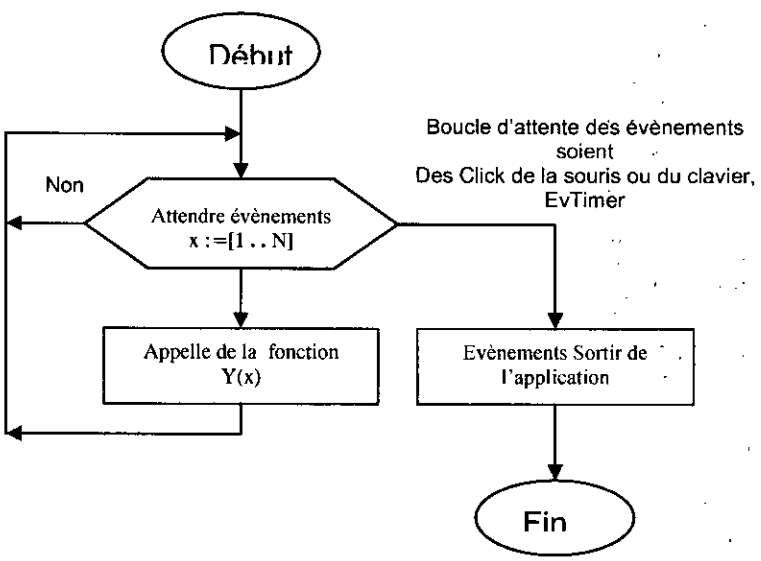



Figure 4.11: Organigramme de l'application

### 4.3.4.1. Temps réel :

Le temps réel recherché dans cette application est la lecture des données à chaque CPI car la carte DSP écrit dans le bus VME à chaque CPI, et le temps d'un CPI est de 78.125ms pour une vitesse de rotation de l'antenne T= 20s et 39,0625ms pour T= 10s.

Le flux des données au minimum est estimé à 20 plots par CPI. On a 2*20+2 +4=46 cycles ce qui donne en temps de calcul de 46*40ns = 1840 ns.

Pour l'écrasement des données au niveau de la FIFO de 2048 mot, peut arriver dans le cas où le processeur Pentium ne récupère pas les données pendant un temps inférieur au remplissage de la FIFO.

La FIFO peut se remplire pendant 2048/50 plots = 40.96 CPIs c'est à dire soit 40.96*78.125=3.2s et 40.96*39.0625=1.6s pour T=20s et 10s respectivement. Le temps réel principal signifie l'affichage de tous les plots à chaque CPI.

La récupération se fait en synchronisation avec la fonction EvTime qui à chaque

interruption essaie de récupérer le maximum de données de la DSPFifo. Le code de EvTimer suivant la visualisation lit toutes les données de DSP->FIFO, cette interruption est générée toutes les 150 ms.

```
void fastcall Tf3rincipale: TtmerlTimer(TObject *Sender)
{
  Controle->FifoEmpty();
  Controle->Read_Dsp_Fifo();
}
```

#### 4.3.4.2. Sélection de la carte DSP :

Cette fonction est très importante vu son utilisation dans la plupart des applications. Elle consiste à initialisé le registre Tundra à l'adresse de la carte DSP selon le code source suivant :

```
void SelectlfJCardO
      {
        long Iff Card Win = 0x40830000L;
        TundraBasePtr[Ox048] = Iff Card Win - Ox000E0000L;
      }
//*****************************************************************************
```

| | |
|---|---|
| 0x000E0000 | présente l'adresse de base des cartes DSP. |
| 0x40830000 | présente l'adresse de la carte DSP. |
| 0x000D0000 | présente l'adresse de base du Tundra. |
| 0x000D048 | présente l'adresse du registre Tundra. |

Cette fonction permet de sélectionner la carte DSP IFF sur le système XVME653 dont l'adresse est 0x040830000 via le circuit Tundra qui fait l'interface entre Bus PCI et VME.

#### 4.3.4.3. Chargement de l'application DSP :

Cette fonction a pour tâche le chargement de l'application IFF présentée par le fichier ALL.BTL dans la carte DSP TMS320C40 via le VMEbus vers DSPFIFO en faisant appelle au **BootLoad.**

Le programme **bootload** ou chargement du programme de démarrage a pour fonction de s'assurer que l'application DSP est chargée sur système et qu'elle fonctionne correctement quand le système est allumé ou après un reset hardware du système.

Pour la carte DSP TMS320C40 est liée par le port de communication n°1 ainsi le chargement se fera à travers le port de communication. Pour cette raison, l'état des lignes d'interruption dans la figure.4.13 après allumage du système ou après reset hardware est fixé comme suit :

- IIOFO : High
- IIOFI : High
- IIOF2 : High
- IIOF3 : High

L'opération se fait grâce au circuit U23 FPGA XC95108, à l'allumage l'état des pins IIOFO..3 sont mis à 1, et si on veut un bootload depuis l'EPROM il suffit de placer le Link qui mis l'entrée BootSel à 0 ainsi on aura à la sortie du FPGA « 0111 » et le programme implanté à l'intérieur du FPGA est écrit en VHDL comme le montre le code suivant :

```
  port (
    T:          in std logic vector (3 downto 0); — The T3.. TO discrete inputs
    BOOTSEL: in std_logic;                        — The BOOT Option Select input
    IACKN:      in std logic;                     — The IACKN signal from the DSP
    RESET:      in std_logic;                     — The RESW. GSR reset signal
    IIOF3 IN: • in std logic;                     — The 11OF3 input from the DSP
IIOF3 INT EN:in std_logic;                              — when '0, enables the I/OF3 signal
to be a DSP output
  IIOF3 OUT::out std_logic;                     — The 11OF3 output to the DSP
  I/OF2_OUT::out std_logic;                     — The I/OF2 output to the DSP
  IIOF1 OUT. out std_logic;                     — The IIOF1 output to the DSP
  IIOFO OUT::out std_logic;                     — The IIOFO output to the DSP
  11OF3 EN: out std_logic;                      — 'enables' IIOF3 to be a DSP input
  11OF3_INT: out std_logic;                     — the 1/OF VME Bus interrupt output
  BOOT OK: buffer std_logic                     — Asserted when DSP Bootload has finished
);
```

Le code ci-dessus montre la façon dont les entrées sorties sont déclarées, pour le code qui suit on voit les différents états des sortie IIOFO..3.

I IOF_out_control:

```
        process(BOOTSEL, BOOT_OK, T)
        begin
          if (BOOT OK = '0') then if
            (BOOTSEL ='1') then
              IIOF3 OUT <= '1';— Bootload from a DSP COM port
              IIOF2_OUT <='1
              IIOF1_OUT <='1'; II
              OFO OUT <= '1';
            else
              IIOF3_OUT <='0';— Bootload from address 80000000h
              IIOF2_OUT      <='1';
              IIOF1_OUT   <=   '1';
              IIOFO_OUT <= '1';
            end if;
          else
              IIOF3 OUT <= T(3);            - Autorise le passage des interruptions externes
vers le DSP si IACK est activé.
              IIOF2_OUT <= T(2);
              IIOF1_OUT <= T(1);
              IIOFO_OUT <= T(0);
          end if;
```

```
end process;
```

Le signal **IACK** envoyé par le DSP TMS320C40 est actionné dans l'application IFF pour autoriser ces impulsions comme l'indique le code source suivant du programme **INIT.H** :

```
/* Validation des interruptions IIOFs par IACK                    */
    asm(" push ar0          ');
    asm(" ldi       @_dummy,ARO');
asm(" IACK *arO y; asm(" pop art)     ');
```

Le code en VHDL suivant montre l'action du signal IACK car si IACK = 0 alors BOOT OK =0 ainsi IIOF="1111" si BootSel =1, si BootSel=O IIOFO..3 = "0111" et si IACK = 1 alors BOOT_OK=1 ainsi les Signaux IIOFO..3 passent directement vers le DSP.

Pour générer Le programme boot, on utilise un fichier de commande, et ces commandes seront traduites en code exécutable par le programme utilitaire de conversion HEX30.EXE de Texas Instruments voir la table 4.1.

Le programme de conversion génère un fichier au format TEKTRONIX qui sera chargé dans le DSP TMS320C40 voir le tableau 4.2.

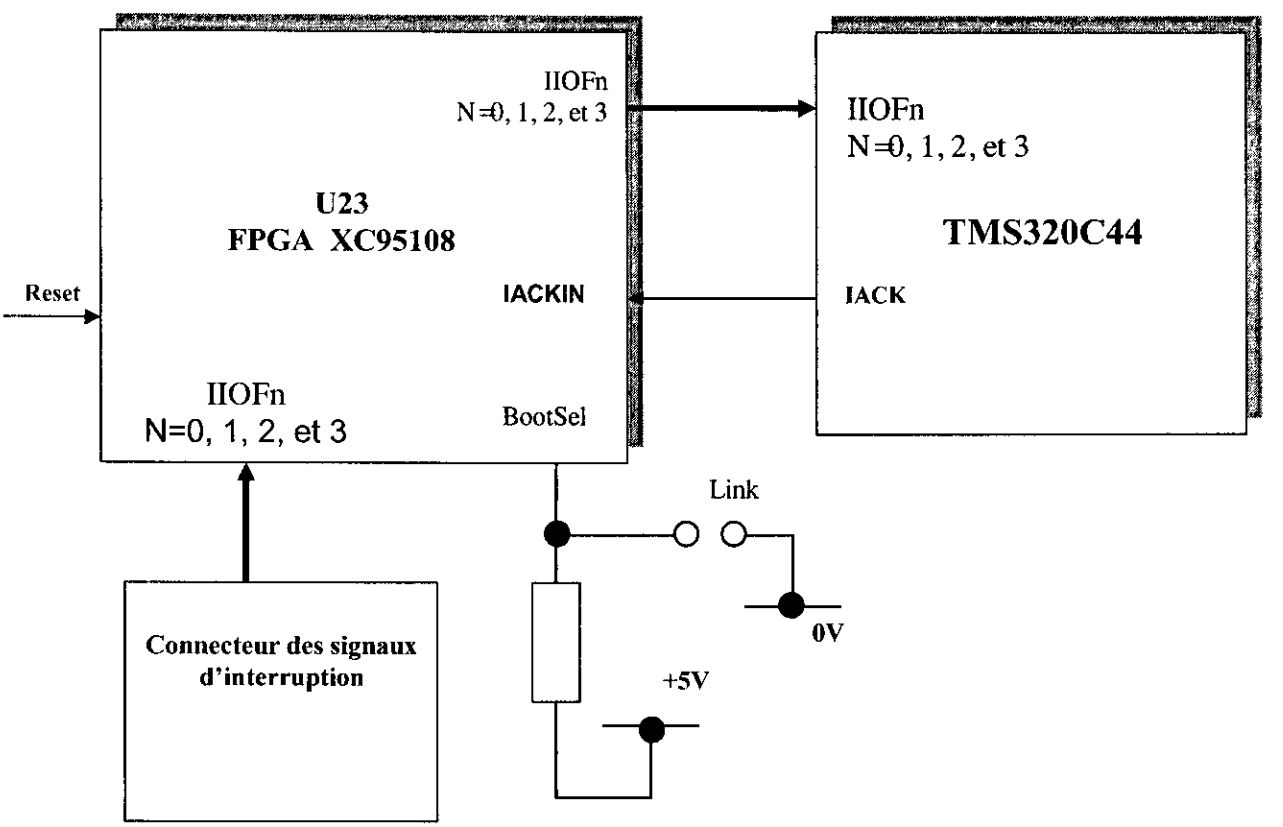



Figure 4.12: Génération du signal HIGH pour BootLoad

Table .4.1. fichier de commande HEX30.

```
all.out
-o all.btl
-map all.mxp
-e 003005b8h
-x                /* Select the Tektonix format                              */
-memwidth 32      /* applicable because the c32 8/16/32 structure            */
-romwidth 32      /* rom width 32 if comm boot loading a C40-C44             */
-boot             /*                                                         */
-bootorg COMM     /*                                                         */
-cg    3593c750h  /*Global control register
-cl    35ad4000h  /*Local control register                                 */
-ivtp 002ffc00h   /*                                                        */
-tvtp 002ffc00h   /*                                                        */
-iack 00300000h   /*                                                        */
/*************************************************************************/
/* May also add the rom location start address if booting from memory     */
/* Remember to change the bootorg = COMM for rom bootloading              */
/*************************************************************************/
/* ROMS {EPROM: org = 0300000h, len = 2000h}                              */
/*************************************************************************/
```

Table 4.2. Fichier Hex format TEKTRONIX.

```
%4E6CC8000000003593C75035AD400000000001003004DF00000000000006F5003000000F2B0000
%4E628800000008080B00140820079E04E000016A06013615000A6F0821089E15210A701500079E
%4E6388000000101500089204A108926A08000B1EA80F101EB10892026800081EA90F1008404000
%4E61B800000018154041000820089231010001152108920 4A10A706A07FFF5082007A0152007B6
%4E6F880000002004E000F86A0700021860010015200 7B63101000815210 7B61500089208200892
%4E64A80000002804A00A706A0A00170820089215200893 04A00A706A0A000E1EA80F101EB10892
%4E60D8000000301EB2089323E040806A0800040840400015200C1C08418000C201408008200893
%4E6138000000383101000115210893 04A10A706A07FFF208200892310100011521089204A10A70
%4E66D8000000406A07FFE9150008920820089204A00A706A0A000D1EA80F101EB10892082007B6
%4E
```

#### 4.3.4.4. Lecture de la carte DSP:

La lecture des données se fait à chaque CPI avec la fonction **Read_Dsp_Fifo()** , le processeur inspecte si la FIFO est vide et si le temps d'attente dépasse les 100 ms revient à l'application pour les autre tâches. Le code suivant :

*while (DspReadFifoEmpty() && (GetTickCountO-LocalTimeStart)< MaxTimeout);*

**DspReadFifo Empty()** lit le registre d'état de la carte DSP pour tester le premier bit s'il est égal à 1 c'est à dire la fifo est vide.

**GetTickCount()** est une application Windows qui donne le temps écoulé depuis l'ouverture de la fenêtre.

La lecture des données se fait selon le protocole de communication de l'application IFF comme le montre l'organigramme 4.13.

*La lecture du Flag:* la lecture des données le premier mot qui sera lu est le Flag qui doit être égal à OxObbbb ainsi le Pentium va boucler tant que la données lue est différente du code OxObbbb et s'il y a des données le Pentium va procéder au vidage de cette fifo. *La lecture du N° CPI :* le n°CPI doit être compris entre 0 et 255 si non ; si n°CPI <0 n°CPI=O, et si n°CPI > 255 n°CPI = n°CPI précédent.

*La lecture du nombre de plots :* Le nombre de plots doit être positif et inférieur à 100 si non il est remis à 0.

*La lecture des données de l'unité de poursuite :* La lecture de ces données se fait si le flag ModeInterg = Ox07eee.

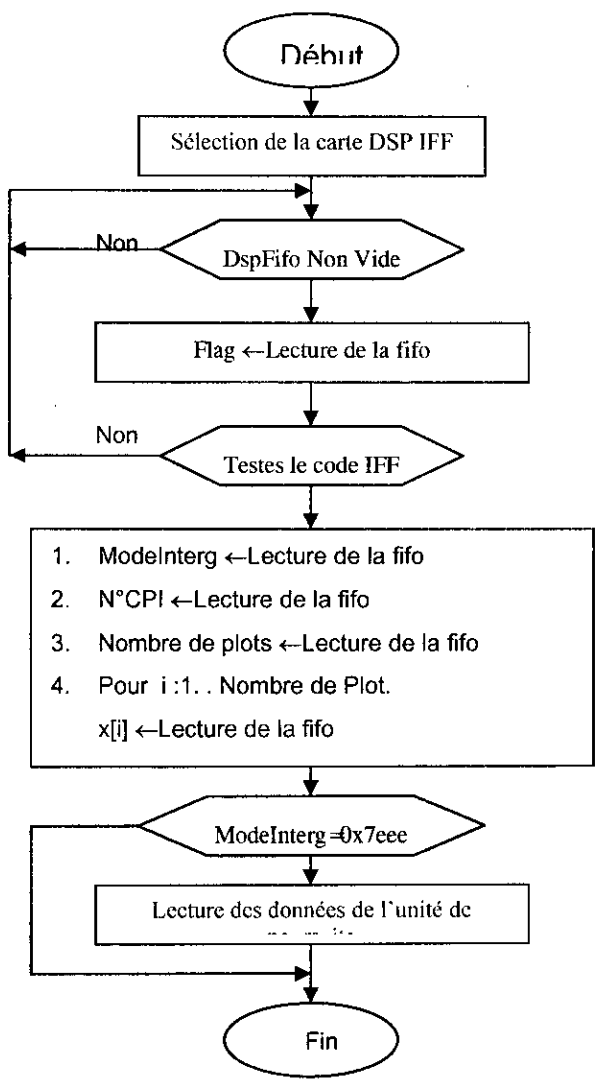



Figure 4.13: Organigramme de Lecture

### 4.3.4.5. Ecriture dans la carte DSP:

L'écriture des données dans la carte DSP IFF se fait avec la fonction ***Send Iff Message() ;*** cette fonction permet d'envoyer le message à destination de la carte DSP et qui est constitué du tableau Message[24]. L'ordre de l'envoi est donné avec un click sur le bouton SendIff. Le message est constitué de l'entête qui est le code du message IFF (Message[I]), le mode de fonctionnement et le nombre de demandes IFF dans Message[7] comme l'indique le code source suivant :

```
void _fastcall TPrincipale:: SendIffClick(TObject *Sender)
{ Message[7]=IffRequest;
for (int i=0;i<SizeMessage;i++) Controle->Send Iff Message((DWORD)Message[i] );
for(int i=0; i<SizeMessage; i++)Message(i]=0;)
```

### 4.3.4.6. Visualisation:

Pour dire que l'application de visualisation est fonctionnelle sans erreurs c'est quand le balayage avec la couleur jaune du radar (voir la figure 4.14). ne s'arrête pas de tourner et que l'affichage se fait correctement sur toute la portée du radar. Dans le cas où il y a perte d'information en particulier le n°CPI ; la perte de numéro CPI ce vecteur ne sera effacé qu'après le tour d'antenne suivant (voir la figure.4.15).

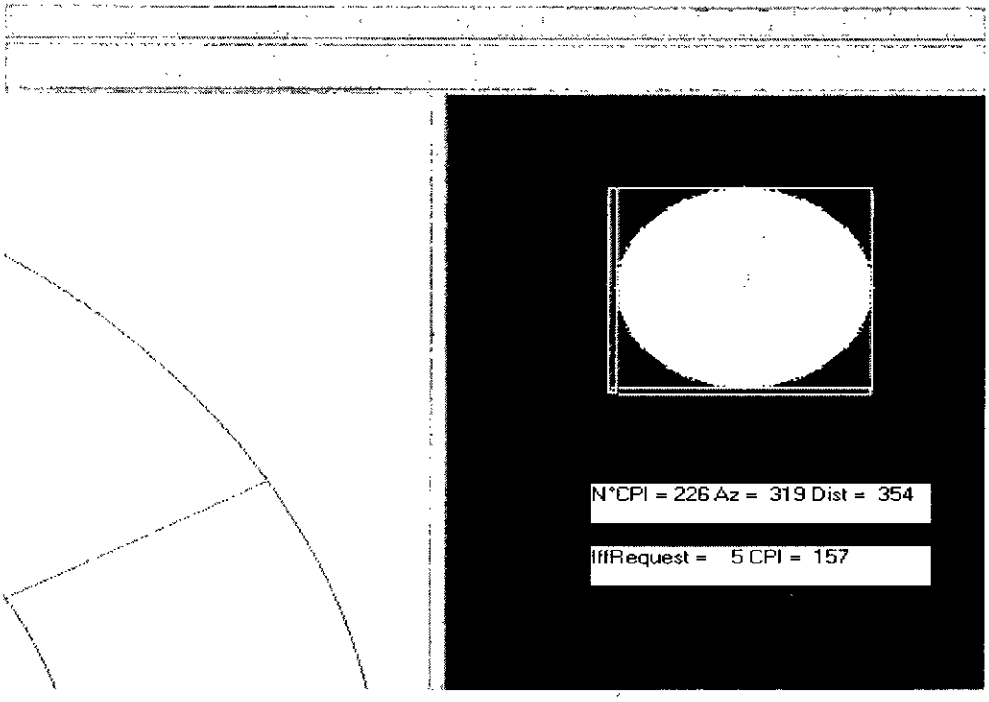



**Figure 4.14: Balayage du radar dans l'application de visualisation**

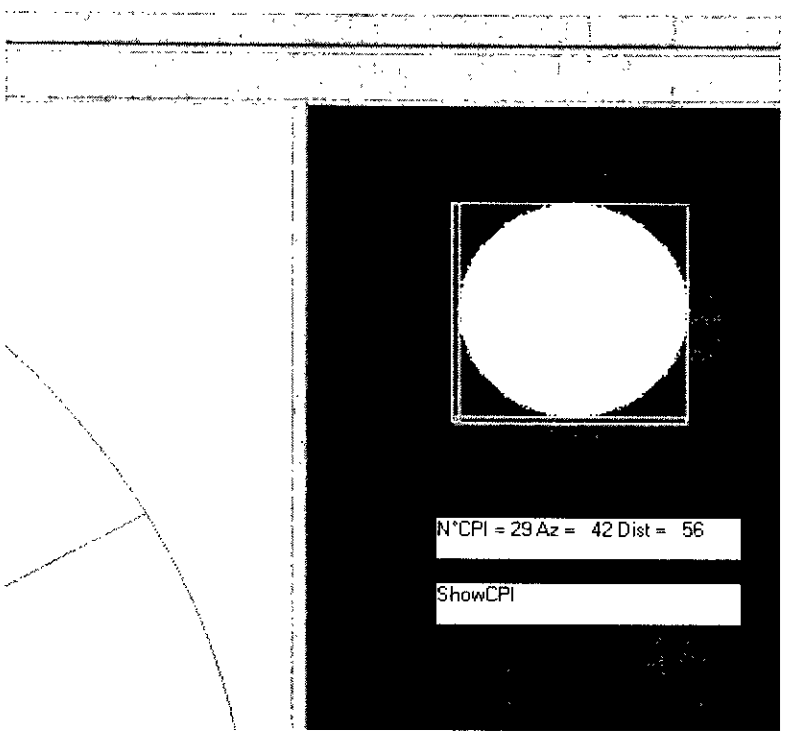



Figure 4.15: Balayage du radar avec une erreur de CPI

Le dessin du vecteur de rotation est lié directement aux n°CPI car le vecteur est traçé avec ces N°CPI. Le tracé du vecteur se fait en temps réel juste après la lecture de la carte DSP TMS320C44 du N°CPI, un vecteur est tracé en noir sur le vecteur du CPI précédent ainsi l'effacer, et après un autre vecteur en jaune est tracé avec la couleur jaune et ainsi de suite on voit le vecteur tournait. L'erreur du balayage apparaît quand le N°CPI n'est pas bien lu alors le vecteur du CPI ne sra pas effacé. Le code source suivant montre la façon du tracé du du vecteur appelé ScopeVecteur:

```
int c_cpi=(int)DspReadFifo(TRUE);
if( c_cpi < 0) c_cpi=0;
if( c_cpi > 255) c_cpi=Next;
Next=(c_cpi+1)%256;
DrawScopeVector(c_cpi,Next,Choise);

void __fastcall TControle::DrawScopeVector(int Current,int Next,int Choice)
{  Canvas->Pen->Color = clBlack;
   Canvas->MoveTo(ScopeCentX,ScopeCentY);
   Canvas->LineTo(XScopePos[Current],YScopePos[Current]);
   if (Choice == 0) Canvas->Pen->Color = clYellow;
   Canvas->MoveTo(ScopeCentX,ScopeCentY);
   Canvas->LineTo(XScopePos[Next],YScopePos[Next]);}
```

### 4.3.4.7. Interface graphique :

L'interface graphique de l'application de visualisation comme le montre la figure 4.16 permet de visualiser les résultats de l'extraction de plots radar secondaires d'une situation radar et de rendre plus facile à l'opérateur radar l'exploitation de ces résultats.

L'interface graphique se compose de trois parties principales ; la fenêtre principale, la fenêtre de contrôle et la fenêtre de visualisation.

*La fenêtre principale :*

La fenêtre principale de l'application est l'interface visuelle qui est affichée à l'écran du VME au lancement de l'application. Cette interface occupe toute la surface de l'écran sous forme d'application sous windows en haute résolution 1280*1024 pixels et avec 256 couleur pour l'affichage. On retrouve dans cette interface toute les fonctionnalités d'une application Windows telles que les menus déroulant, la gestion de tous les évènements par des clicks de la souris etc. Dans cette interface graphique on aura :

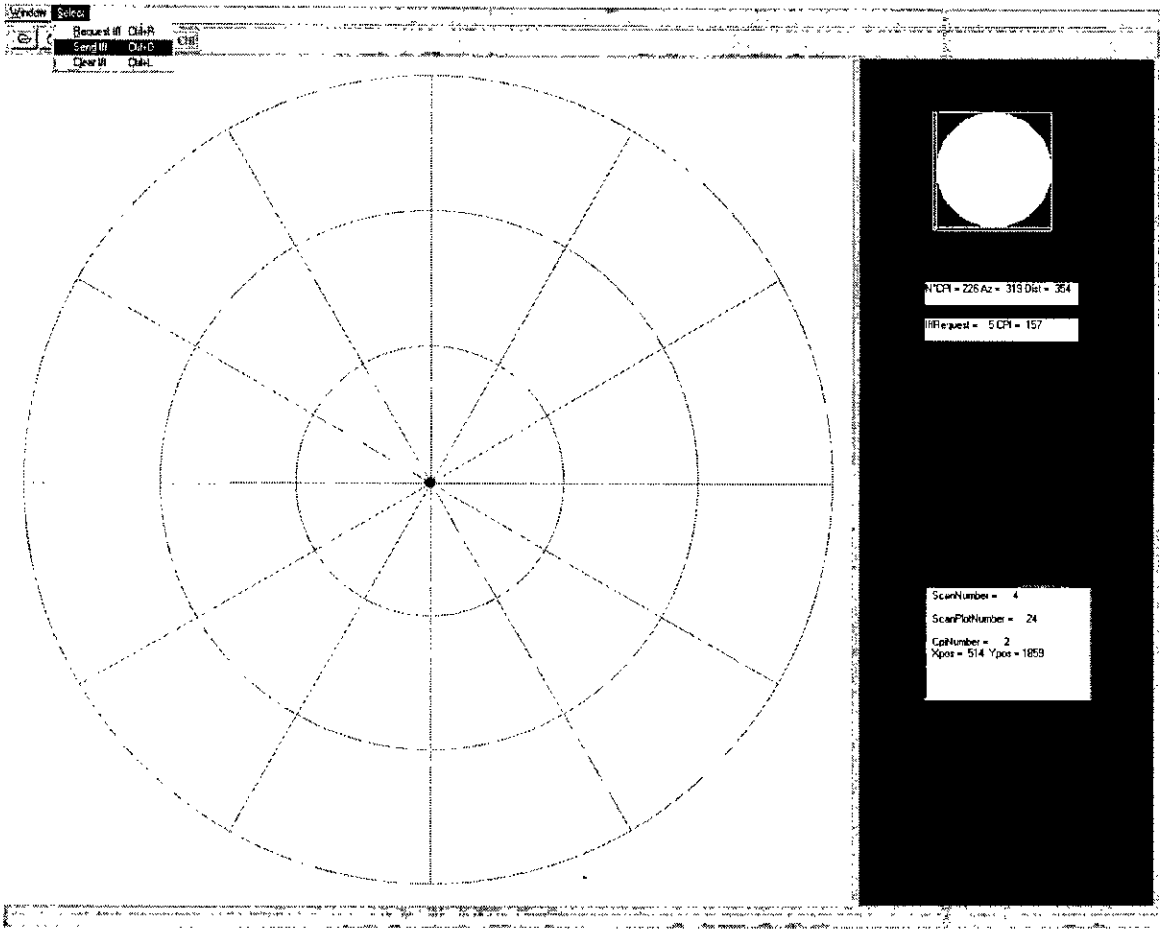



Figure 4.16: Interface Graphique de l'Application Visualisation

- La barre de menu :

  La barre de menu contient un ensemble de menu déroulant, l'ensemble des menus est composé des menus : Windows et Select, et on peut rajouter plus menus tel que le menu zoom et le menu Move.

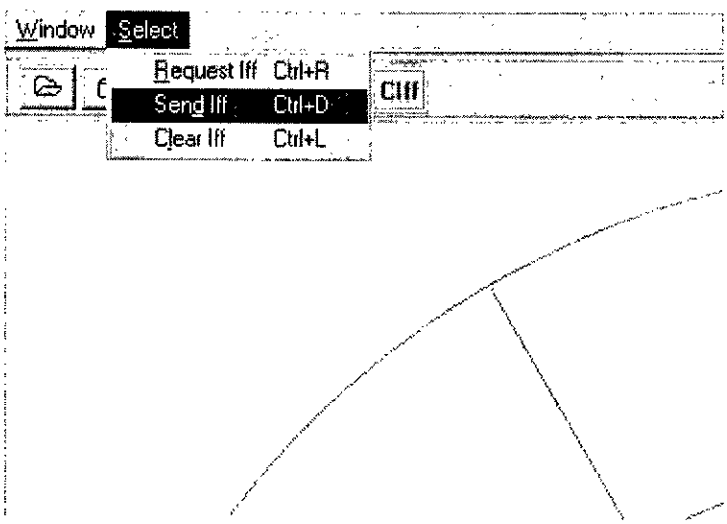



Figure 4.17: Le menu déroulant de l'interface graphique

Le menu ***Windows*** avec un raccourcis clavier de « Ctrl+W » permet d'ouvrir une nouvelle situation radar, fermer une situation déjà ouverte et enfin quitter complètement l'application.

Le menu ***Select*** avec un raccourci clavier de « Ctrl+S »permet de faire trois actions : sélectionner des cibles radar pour les interroger, donner l'ordre d'interrogation et enfin annuler cet ordre d'interrogation.

- Panneau des raccourcis :

Le panneau des raccourcis comprend 6 boutons, ces boutons représentent des raccourcis des commandes de la barre de menu.

*La fenêtre de contrôle :*

La fenêtre de contrôle occupe la partie droite de l'écran voir la figure .4.18. La fenêtre est menue d'un ensemble d'indicateur pour montrer un ensemble d'information utiles. Parmi ces informations, on trouve :

- Un vecteur tournant qui indique à tout instant la position angulaire actuelle du radar.
- Une zone texte qui affiche la date en jour mois et année.
- Une zone texte pour afficher l'heure courante en heure, minute et seconde. Une zone texte pour afficher le numéro du CPI courant , l'angle en degré et la distance en kilomètre entre le site de la station radar et le pointeur de la souris lors du déplacement du curseur de la souris sur la fenêtre de visualisation.
- Une zone texte pour afficher les informations suivantes :
  - Numéro du tour d'antenne depuis le lancement de l'application « ScanNumber ».
  - N°CPI courant « CpiNumber ».
  - Nombre de plot du tour d'antenne "ScanPlotNumber".
  - Nombre de plots par CPI « CpiNumber ».
  - Les coordonnées cartésiennes X,Y des plots lus.

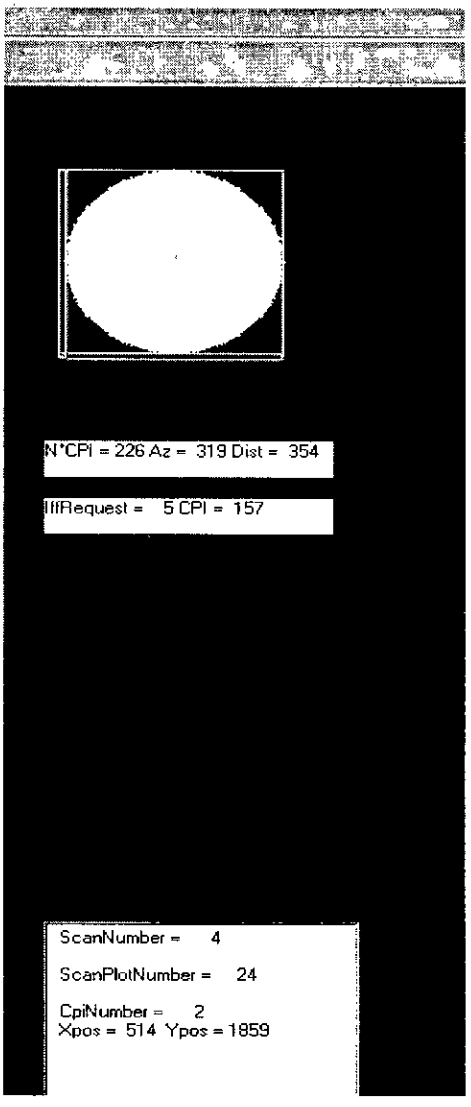



**Figure 4.19: La fenêtre de contrôle de l'interface graphique**

La fenêtre de visualisation :

Les plots de l'extracteur IFF sont visualisés dans la fenêtre de visualisation. Cette dernière est une fenêtre carrée. La taille de la fenêtre de visualisation est maximale est autour de 940 pixels pour une résolution graphique de 1280*1024. Le fond de cette fenêtre est en couleur noire et elle peut contenir :

- Un gros point rouge qui représente le lieu ou la position du site de la station radar.
- Des marques distances circulaires et des marques azimuts sous forme de lignes centrées sur le site radar. Les marques distances indiquent les distances 50, 150, 200,250 et 300 Km. Pour les marques azimut divisent la vue du radar en 12 secteurs de 30°.
- Les plots de la réponse IFF de visualisation .
- Les plots IFF de poursuite si on veut.

La fenêtre de visualisation est menue aussi d'une interactivité avec le pointeur de la souris :

- Le déplacement du pointeur de la souris sur la fenêtre de visualisation permet d'afficher automatiquement la position du curseur sur la fenêtre de visualisation (distance en Km et l'angle en degré) dans la fenêtre de contrôle. Cette fonction est réalisé avec l'événement de ***OnMouseMove*** qui retourne lors du déplacement du curseur de la souris la position momentanée de ce dernier comme le montre le code source suivant :

```
void __fastcall TVisualisation::FormMouseMove(TObject *Sender,
      TShiftState Shift, int mouseX, int mouseY)
{
int PosX,PosY,Dist,Angle,Azimut;
char Buffer0[20];
CeofFactor = 1/1.5;
PosX = (mouseX-(Width/2));
PosY = ((Width/2)-mouseY);
Dist = (sqrt(pow(PosX,2)+pow(PosY,2))*CeofFactor);
if (PosY != 0)
     { Angle=57.29577951*atan((double)abs(PosX)/(double)abs(PosY));
     if(PosY > 0)
        {if(PosX > 0) Angle =Angle;
        else Angle =360-Angle;}
     else
        {if(PosX > 0)
         Angle = 180 - Angle;
        else Angle = 180+Angle;
        }}
else Angle=0;
Azimut = Angle / 1.40625;
sprintf(Buffer0, "N°CPI = %d Az = %4d Dist = %4d\n",Azimut,Angle,Dist);
Controle->ShowPosition->SetTextBuf(Buffer0);}
```

MouseX et MouseY représentent le position du curseur de la souris en pixels par rapport à (0,0) de l'écran qui se trouve en haut à tout à fait à gauche de l'écran.

***PosX*** = *(mouseX-(Width/2));*

***PosY*** = *((Width/2)-mouseY);*

PosX et PosY représentent le centre de la fenêtre de visualisation ainsi tous les calcul vont se faire par rapport à ce nouveau repère qui est la position du site radar.

Automatiquement « *Dist = (sqrt(pow(PosX,2)+pow(PosY,2))*CeofFactor);* » va nous donner la distance entre la position du site radar et la position du curseur.

Le click avec le pointeur de la souris lorsque la commande des demande IFF est activée sur la fenêtre de visualisation permet de sélection un ensemble de demande IFF, les N°CPI des cibles est stocké dans le tableau Message[ ] pour les envoyer ensuite via le bus VME vers la carte DSP TMS320C44. La procédure de sélection se fait grâce à l'événement ***OnMouseDown*** qui retourne en cliquant sur la souris la position du curseur de la souris comme le montre le code source suivant :

```
void __fastcall TVisualisation::FormMouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int mouseX0, int mouseY0)
{int PosX0,PosY0,Angle0,Azimut0;
char Buffer1[20];
CeofFactor = 1/1.5;

if(Principale->RequestIFF == true )
{PosX0 = ((mouseX0-(Width/2))*CeofFactor);
      PosY0 = (((Width/2)-mouseY0)*CeofFactor);
      if (PosX0 != 0)
Angle0=57.29577951*atan((double)abs(PosY0)/(double)abs(PosX0));
           if (PosY0 != 0)
    {Angle0=57.29577951*atan((double)abs(PosX0)/(double)abs(PosY0));

                if(PosY0 > 0)
                        {if(PosX0 > 0)
                                Angle0 =Angle0;
                        else Angle0 =360-Angle0;}
                else
                        {if(PosX0 > 0)
                                Angle0 = 180 - Angle0;
                        else Angle0 = 180+Angle0;}}
                else Angle0=0;
        if(Principale->IffRequest < 16)
        {Principale->IffRequest++;
        Azimut0 = Angle0 / 1.40625;
        Principale->Message[Principale->IffRequest+7]=Azimut0;
        sprintf(Buffer1, "IffRequest = %4d CPI = %4d\n",Principale-
>IffRequest,Azimut0);
        Controle->ShowCPI->SetTextBuf(Buffer1);}}}
//---------------------------------------------------------------------
```

## 4.4. Les bancs de tests :

Pendant la réalisation du nouveau système en particulier l'application IFF les résultats des tests étaient satisfaisant car l'importance était le bon déroulement de l'application IFF, et pour tester le bon fonctionnement beaucoup d'outils ont été utilisés, je citerai l'application **IntCount**, **IffRead, IffRW et IffVisu.**

### IntCount :

Cette application est une application DSP chargée dans la carte DSP TMS320C44 permet de faire le comptage des différents signaux d'interruption qui arrivent aux pin IIOF0...IIOF3 en cas où il y a absence des signaux ou quand il y a des signaux parasites aussi :

*Continuously reading DSP FIFO words using ReadDspFifo()*

*1* *; Premier tour d'antenne T=12s.*

*3840* *;Nombre de PRFs par tour d'antenne (3840 ⇒ PRF=320Hz).*

*277* *;Nombre de CPI pour un angle de résolution de 1.3°.*

*0* *;Absence de signaux sur la pin IIOF3.*

*2*

*6423*
*256*
*0*
*3*
*6999*
*256*
*0*

**IffRead :**

L'application IffRead est écrite avec le Borland C++ version 4.52 sous DOS, et a pour fonction la lecture continue sans aucun protocole des données **DSPFIFO → VME** et enregistrer ces données dans des fichiers pour vérification et à son exécution on aura sur l'écran les données suivantes :

| | |
|---|---|
| *-17477* | *(code message iff).* |
| *32221* | *(ModeInterrogation).* |
| *1* | *(N° CPI).* |
| *1* | *(Nombre de plots).* |
| *1108* | *(x en centaine de mètres).* |
| *27* | *(y en centaine de mètres).* |

*-17477*
*32221*
*2*
*1*
*1107*
*54*
*-17477*
*32221*
*3*
*1*
*1105*
*81*

**IffRW :**

L'application IffRW est la même que IffRead sauf que cette application prend en compte le protocole de communication et a une fonction en plus pour le test de l'écriture dans la carte DSP TMS320C44.

L'utilisation de cette application permet aussi un enregistrement dans des fichiers des données récupérées via le VMEbus et on aura sur l'écran les données suivantes :

*Flag = -17477 flag = 31948 cpi = 80 number = 1*
*1037 -429*
*Flag = -17477 flag = 31948 cpi = 81 number = 1*
*1026 -455*
*Flag = -17477 flag = 31948 cpi = 82 number = 1*
*1015 -480*
*Flag = -17477 flag = 31948 cpi = 83 number = 1*
*1003 -505*
*Flag = -17477 flag = 32494 cpi = 84 number = 1*
*990 -529*
*iff_cpi = 69 corp = 1*
*1074 -326*
*Flag = -17477 flag = 32221 cpi = 85 number = 1*
*977 -553*
*Flag = -17477 flag = 32221 cpi = 86 number = 1*
*963 -577*
*Flag = -17477 flag = 32221 cpi = 87 number = 1*
*948 -600*
*Flag = -17477 flag = 32221 cpi = 88 number = 1*
*933 -624*

**IffVisu :**

L'application IffVisu sert à vérifier le bon fonctionnement de l'application IFF dont la communication, le protocole ainsi que le traitement à l'intérieur du DSP. Cette application qui a pour fonction la visualisation des données lues de la carte DSP pour les afficher en temps réel, cette application a permet de détecter plusieurs bug au niveau de l'application IFF surtout au niveau des condition limites comme le passage de l'antenne par le nord, le comportement de l'application IFF à la portée maximale du radar.

# Chapitre 5

# Simulation et validation du système d'extraction

## 5.1. Introduction :

La simulation revient à tester le plus exhaustivement possible les fonctions soient ; association, saturation, passage du nord ou dernier CPI et le calcul du centre de gravité des reports par différentes méthodes. Les moyens de simulation utilisés dans ce travail sont divisés en deux parties qui sont les suivantes :

- Simulation des différentes situations par des fichiers de données bien déterminées par le **simulateur SIM4X** de TMS320C4x.
- Emulateur XDS 510 du TMS320C4x avec JTAG.
- Simulation avec la carte DSP TMS320C40 avec le boot de l'application du disque dur à travers le port de communication n°1.
- Simulation de l'application par des signaux radar générés par une carte à base d'un FPGA.
- Dans ce chapitre sera présenté aussi des image et des enregistrements effectués dans le site opérationnel pendant que la station radar effectue sa vacation de surveillance.

## 5.2. Sim4x :

Sim4x est le simulateur des instructions du DSP TMS320C40, ce simulateur a été utilisé pour la simulation des différents modules de l'application de l'IFF avec leur implantation dans la carte DSP TMS320C40. Ce simulateur a pour avantage la simulation des ports de communication en entrée sortie à des fichiers qu'il peut lire ou écrire en utilisant les instructions suivantes

```
ma Ox100062,1,oport          ; rajouter une mémoire pour un port en sortie.
mc Ox100062,dataout.txt,write ; connecter le fichier dataout.txt pour les données en sortie.
ma 0x100061,1, iport         ; rajouter une mémoire pour un port en sortie.
mc Ox100061,dataout.txt,write ; connecter le fichier datain.txt pour les données en sortie.
```

Le sim4x a été utilisée pour le test de simulation des DMA avec les ports de communication en mode unifié et en mode split.

## 5.3. Emulateur XDS510 :

L'émulateur XDS510 est utilisé pour débuguer l'application IFF en temps réel à l'intérieur de la carte DSP TMS320C40. La possibilité offerte par cet émulateur est le chargement de l'application IFF qui se trouve dans un PC à part à travers le câble JTAG vers la carte DSP TMS320C44 qui se trouve dans le rack VME voir le figure 5.1.

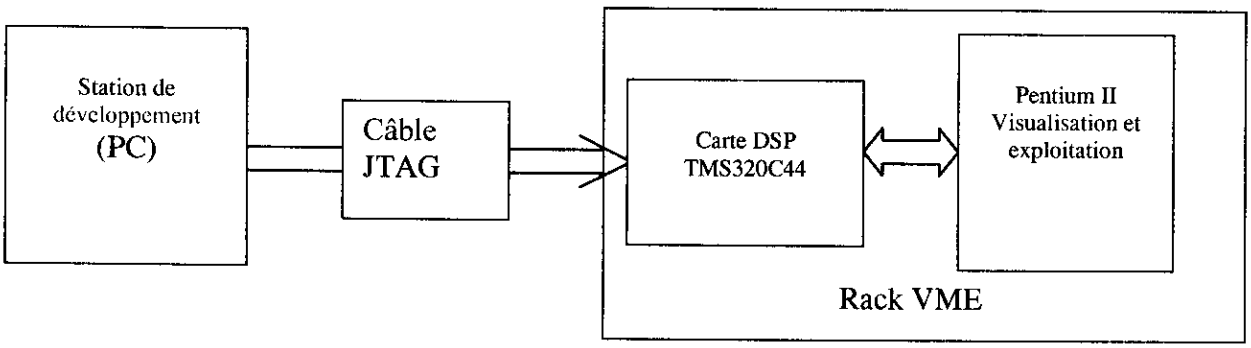



Figure 5.1: La connexion du JTAG

## 5.4. Carte CPLD:

La carte synthétique de génération des signaux de synchronisation et d'IFF est à base de CPLD se qui donne plus de liberté à programmer si on le veut les différentes options de fonctionnement de la station radar, comme il permet aussi à tout chercheur d'utiliser cette carte pour tester des algorithmes de traitement de signal radar en temps réel vu la présence des trois signaux principaux de tous radar de surveillance bidimensionnelle.

L'intérêt de la carte revient à donner la possibilité à un chercheur d'utiliser les 03 signaux principaux de synchronisation dans toute station radar PRFs, CPI et Nord, pour les échantillons du signal écho est stocké dans une mémoire en forme de tableau telle que la ligne *i* représente les échantillons de la PRF *i* ainsi en peut construire un tableau 1000x256x20x5 = 24Moctets dont 1000 nombre de cellules en distance, 256 nombre de cellules en azimut, 20 nombre de PRFs par CPI et le 5 le nombre de scans de la simulation. Mais le problème ici c'est qu'il faut disposer d'un système d'interface pour ces interruptions dans notre cas les entrées IIOF0..3 du DSP TMS320C44.

La figure 5.2 montre le schéma électrique de la carte de génération des signaux. La carte est composée de 4 compteurs 12-bit pour avoir les signaux de synchronisation PRF, CPI et Nord dont les fréquences varient entre 300Hz pour la PRF et 0.05Hz pour le signal nord.

- Le signal PRF : la possibilité de générer deux fréquences 380Hz si ChangePrf = 0 et le double 760Hz si ChangePrf = 1.
- Le signal CPI est modulé par le signal nord à chaque 256 impulsions ce qui donne une résolution angulaire de 360°/256 = 1.40625°, la fréquence du signal cpi peut être doublée avec la mise à 1 de l'entrée ChangeCPI.
- Le signal nord est généré avec la fréquence est de 0.05 Hz pour une vitesse de rotation de 3tr/mn dans le cas où ChangeNord = 0, et de 0.1 Hz pour une vitesse de rotation de 6 tr/mn dans le cas 1.

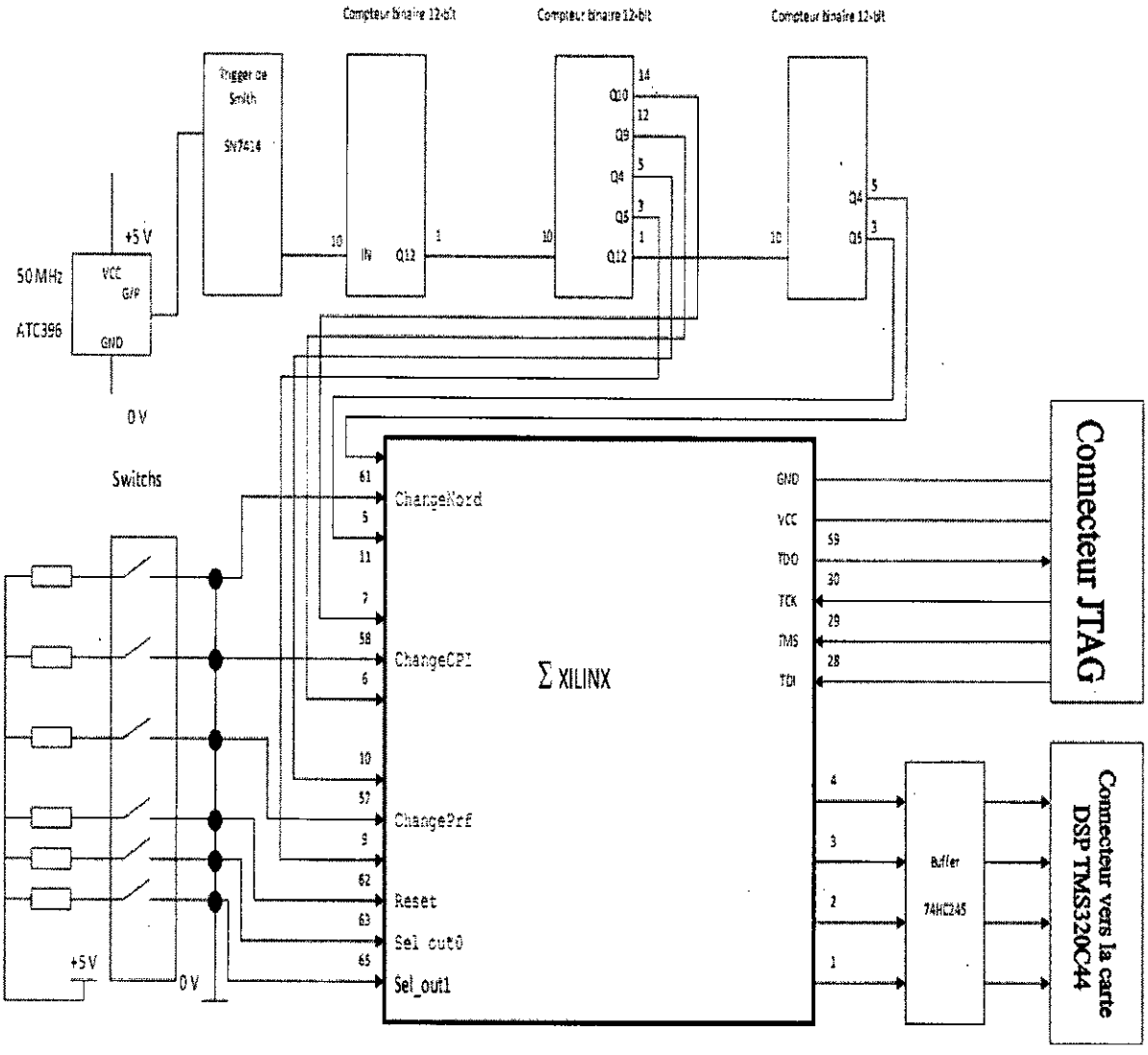



**Figure 5.2: Schéma synoptique de la carte génératrice de signaux radar**

Les signaux à l'entrée du CPLD sont des signaux carrés alors une détection de front montant est effectuée par le CPLD avant d'attaquer le DSP TMS320C44 avec une horloge de 1MHz et le code est écrit en VHDL comme le montre le code suivant :

```
process (clk1Mhz,reset,in_prf,in_cpi,in_nord)
        begin
                if (reset = '1') then
                        out_prf <= '0';
                        out_cpi <= '0';
                        out_nord <= '0';
                        int_prf_flag <= '0';
                        int_cpi_flag <= '0';
                        int_nord_flag <= '0';
                elsif (clk1Mhz 'event and clk1Mhz = '1') then
                        if(in_nord = '1' and int_nord_flag ='0') then
                                int_nord_flag <= '1';
                                out_nord <= '1';
                        elsif (in_nord = '0') then
                                int_nord_flag <= '0';
                        else out_nord <= '0';
```

```
                    end if;

                    if(in_cpi = '1' and int_cpi_flag ='0') then
                            int_cpi_flag <= '1';
                            out_cpi <= '1';
                    elsif (in_cpi = '0') then
                            int_cpi_flag <= '0';
                    else out_cpi <= '0';
                    end if;

                    if(in_prf = '1' and int_prf_flag ='0') then
                            int_prf_flag <= '1';
                            out_prf <= '1';
                    elsif (in_prf = '0') then
                            int_prf_flag <= '0';
                    else out_prf <= '0';
                    end if;
            end if;
end process;
```

Génération de signaux à des distances fixes qui donne au niveau de la visualisation des cercles, voir la figure 5.3. La réponse est située à 190Km comme le montre la boite de distance de la figure 5.3 ainsi que le nombre de plots par CPI qui est de 1 plot, pour exemple pour le CPI 42 CpiNumber = 1 dans la boite Informations. L'affichage des plots secondaires dans cette figure est effectué avec la couleur rouge à la place de la couleur jaune qui n'ai pas visible sur un font blanc. Un enregistrement de la situation est effectué dans le fichier **af00.dat**.

La génération de la cible à cette distance constante est écrit en VHDL comme le montre le code suivant où on peut varier cette distance et générer aussi plusieurs cibles en même temps.

```
process (clk1Mhz,reset,out_prf)
begin
        if (reset = '1' or out_prf = '1') then
                out_iff <= '0';
                compteur <= 0;
        elsif (clk1Mhz 'event and clk1Mhz = '1') then
                compteur <= compteur + 1;
                if(compteur = 1000 or compteur = 1500) then
                        out_iff <= '1';
                else out_iff <= '0';
                end if;
        end if;
end process;
```

On remarque que le compteur des cellules en distance est remis à zéro soit par le signal OUT_PRF qui est la PRF de la station radar.

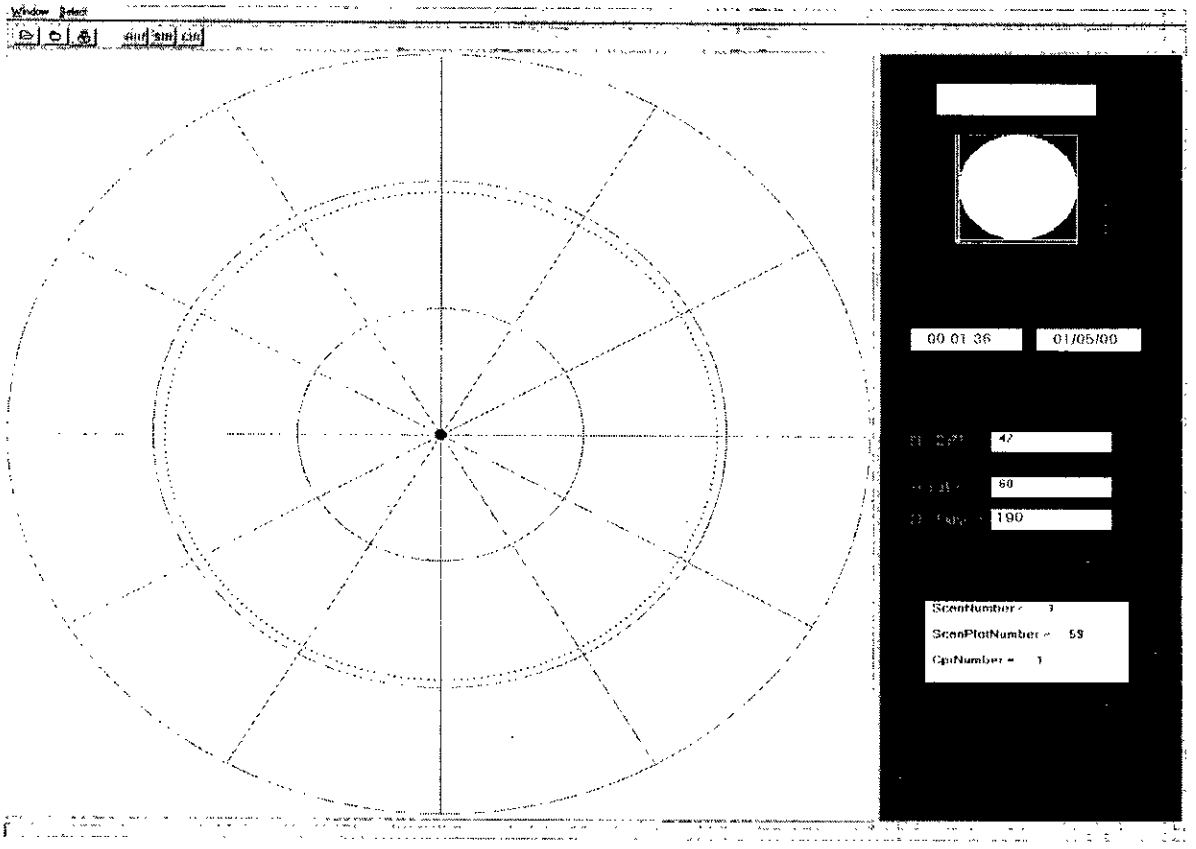



**Figure 5.3 : Image de simulation de cibles en cercle**

La génération de réponses dans un secteur limité comme le montre la figure 5.4, le code VHDL est le suivant :

```
process (clk1Mhz,reset,cpi_counter)
begin
        if (reset = '1' ) then
                cpi_iff_flag <= '0';
        elsif (clk1Mhz 'event and clk1Mhz = '1') then
                if(cpi_counter < 100 and cpi_counter > 64) then
                        cpi_iff_flag <= '1';
                else cpi_iff_flag <= '0';
                end if;
        end if;
end process;
```

Les cibles sont générées dans les CPIs allant de 64 à 100 et à la distance de 190Km.

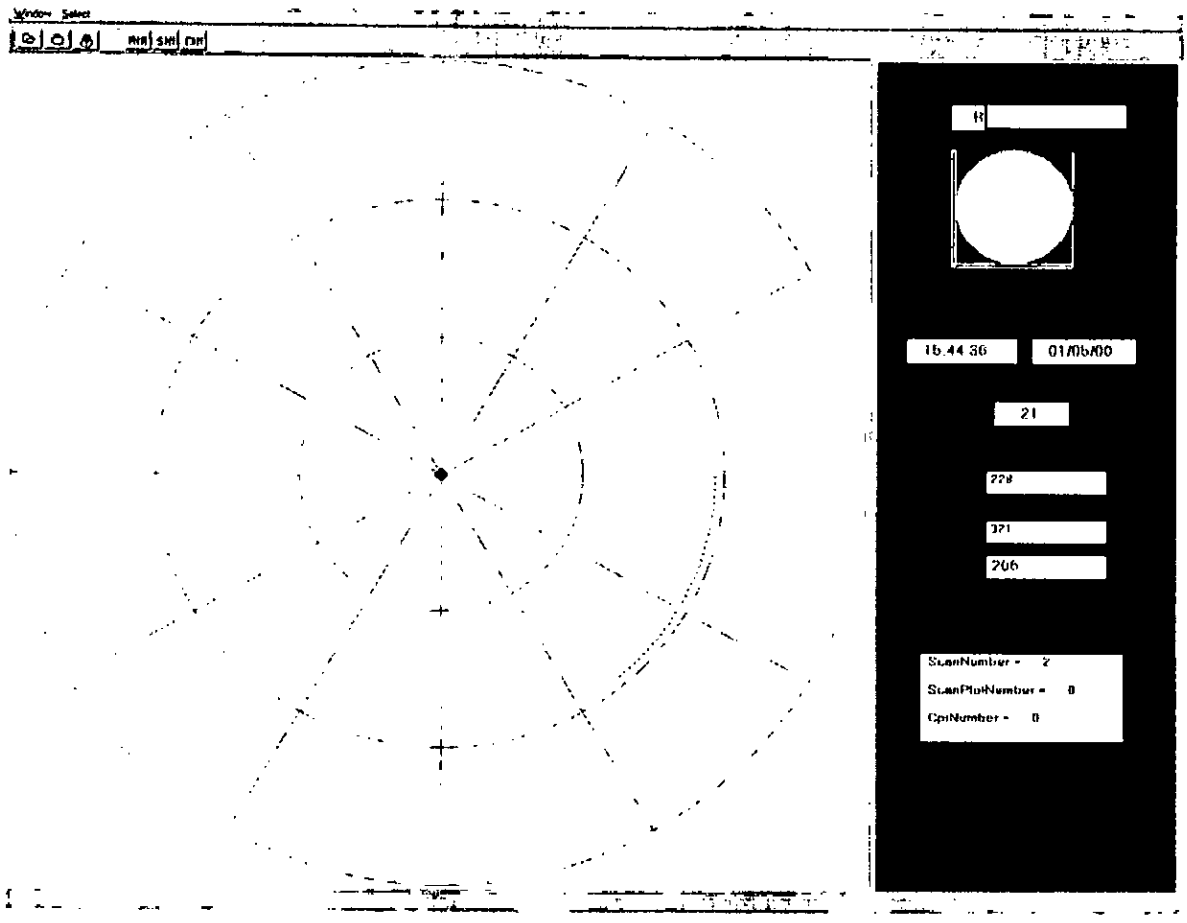



Figure 5.4: Image de simulation de cibles entre le CPI 64 et le CPI 100

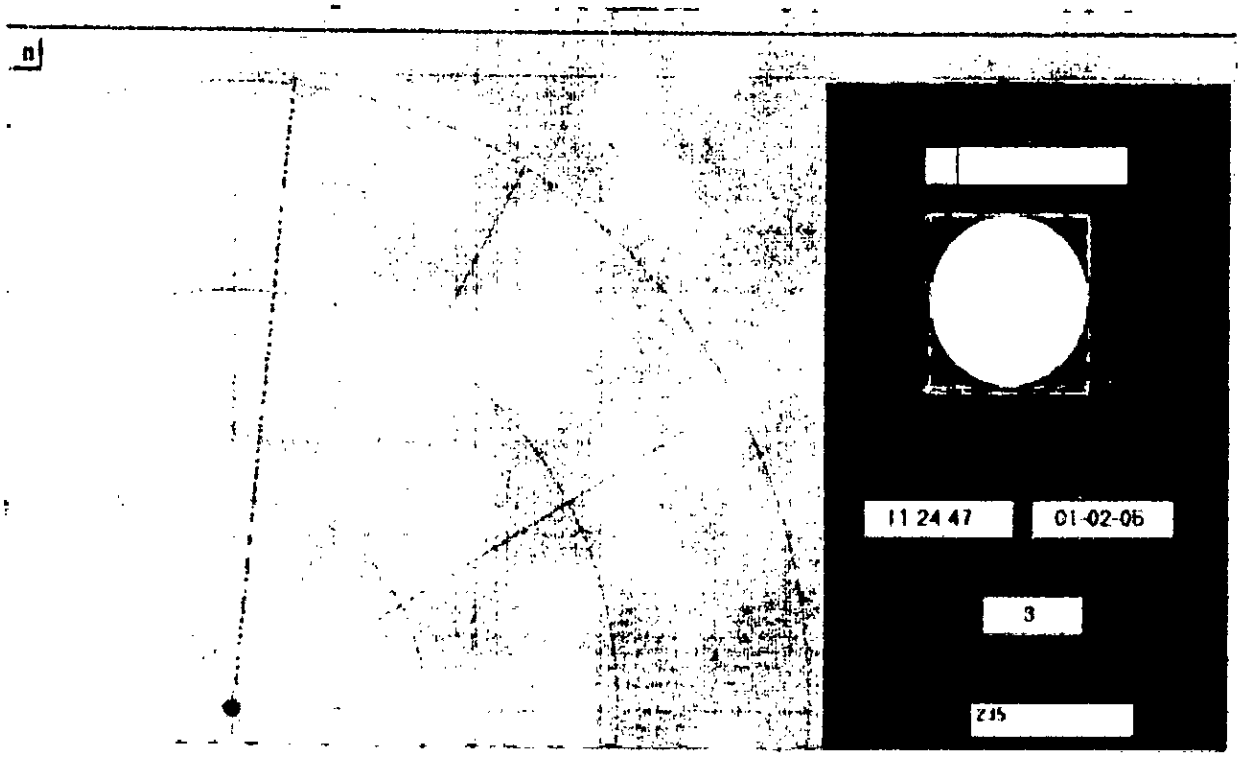



Figure 5.5: Image de simulation de cibles avec signal aléatoire

## 5.5. Tests avec le simulateur:

Le bloc de simulation de la station est un système de test utilisé pour le test du bon fonctionnement du radar secondaire.

Le bloc de simulation est composé d'un codeur, ce qui explique l'existante de l'emplacement du filtre code, d'une petite antenne. Le simulateur envoie des impulsions codées en actionnant la commande manuelle de l'interrogation, ce qui met en marche l'émetteur et le récepteur du radar secondaire et le simulateur et les synchronise avec l'impulsion de déclenchement ou PRF.

Le simulateur génère des impulsions codées pour une cible située à 57 Km et cette distance peut être changée avec un potentiomètre qui agit sur la ligne à retard du simulateur. Sur l'indicateur panoramique une banane apparaît à la distance indiquée ci-dessus tout au long de l'interrogation. L'image de la figure 5.6 montre la situation réelle du radar secondaire sans traitement. De la réponse visualisée sur le nouveau système d'extraction.

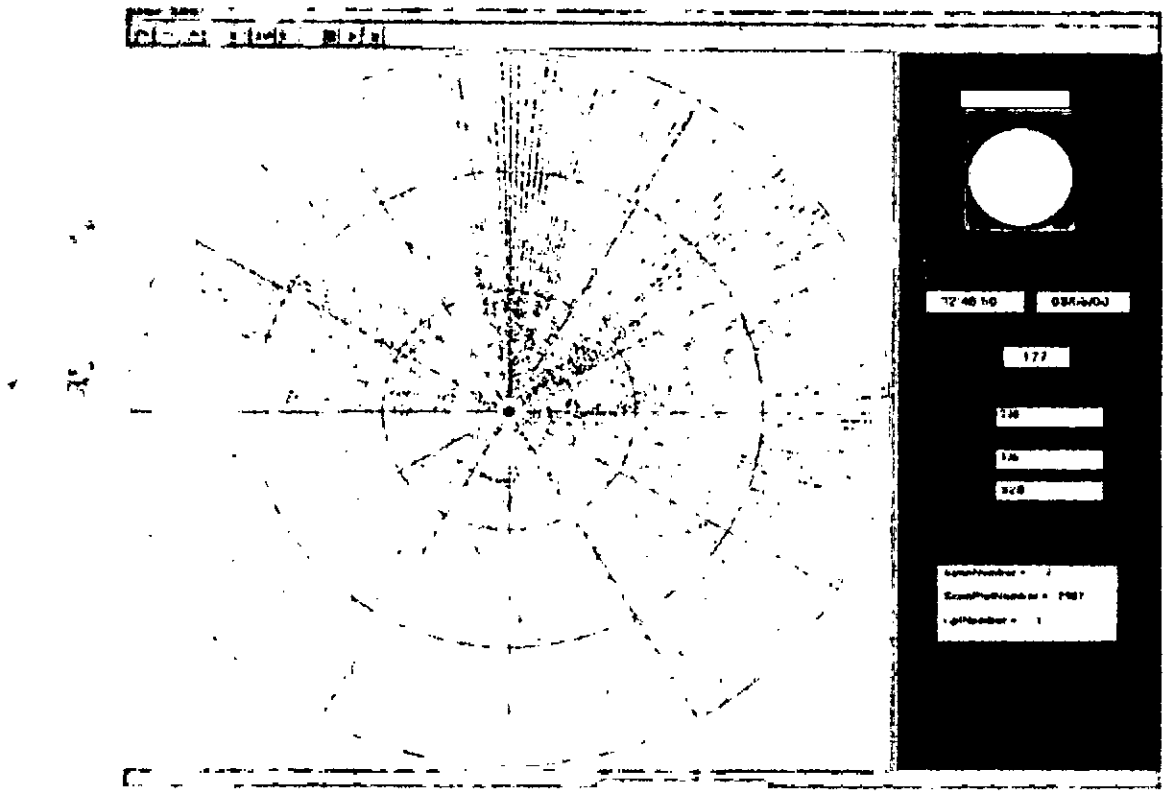


**Figure 5.6: Image du simulateur sans traitement**

## Génération d'un signal pseudo-aléatoire:

La génération du signal est réalisée sur la base de la théorie des séquences binaires dites è longueur maximum [41]. La séquence est produite avec un registre à décalage comportant 15 bascules en série avec un circuit de contre-réaction réinjectant dans la première bascule la somme modulo 2 (circuit OU-exclusif) de la sortie 15 avec la sortie 3. La fréquence de l'horloge est de 1.2Mhz.

Ainsi d'après cette valeur la période du signal est de 0.83 X $2^{15}$ = 27197.44ps. L'image de la figure 5.5 montre la saturation de toute la situation radar avec un signal pseudo aléatoire avec un seuil égale à 0.1 le nombre de plots par CPI est supérieur à 100 ainsi on a plus de 24000 plots par tour d'antenne. Un enregistrement de la situation est effectué dans le fichier **af02.dat.** On remarque dans cette figure le vecteur en pointillé noir qui représente la procédure d'effacement des plots précédents en dessinant les mêmes plots du tour d'antenne précédent de chaque CPI avec la couleur noire.

Sur l'image 5.6 on voit le bruit se répondre sur toute la situation radar, la présence du bruit du côté nord est très importante (nombre de plots par CPI supérieur à 200) vue la présence d'obstacles de ce côté. Entre le N°CPI 120 et le N°CPI 165 on remarque la présence de réponses IFF du simulateur, la réponse dans ce secteur est très apparente à cause de la position de l'antenne simulatrice par rapport à l'antenne du radar à 209° (N° CPI 148).

L'image de la figure 5.7, présente la réponse du simulateur radar avec l'application IFF alors il n'y a que les cibles dépassant le seuil de validation qui sont envoyées vers la visualisation. Le fichier **af04.dat** présente l'enregistrement de la réponse du simulateur.

Sur l'image, la réponse du simulateur apparaît avec la couleur rouge qui choisi pour dire que l'interrogation n'est pas commandé par le système d'extraction mais elle est faite à partir de l'interrupteur manuel. D'une part en remarque l'interface graphique de l'application dans nos sites sur les figures 5.8, 5.9. Comme on le remarque, la présence de la situation du radar primaire avec la couleur verte ainsi 03 aéronefs (66°,134Km), (128°,177Km) et (250°,234KM).

Avec le simulateur, des enregistrements ont été fait avec les applications du banc d'essai **iffrw.exe** cette application n'a pas seulement pour fonction le test de l'écriture et la lecture dans la carte DSP TMS320C40 de L'IFF à l'adresse 0x4083 mais aussi l'ouverture d'un fichier donnée **AFR.DAT** pour sauvegarder les données lues de la carte DSP IFF. Ainsi le premier enregistrement **AFR0.DAT** présente la réponse du simulateur radar avec le bruit, et on remarque que le bruit est très dense. L'enregistrement du bruit est effectué en réduisant le niveau de validation des réponses à 0.2 de 0.5. Le deuxième enregistrement **AFR1.DAT** présente la réponse du simulateur avec un seuil de validation de 0.5 déterminé après des essais sur site avec le simulateur. Le seuil est choisi une fois que toutes les fausses réponses de la situation radar ont disparues sauf la réponse du simulateur.

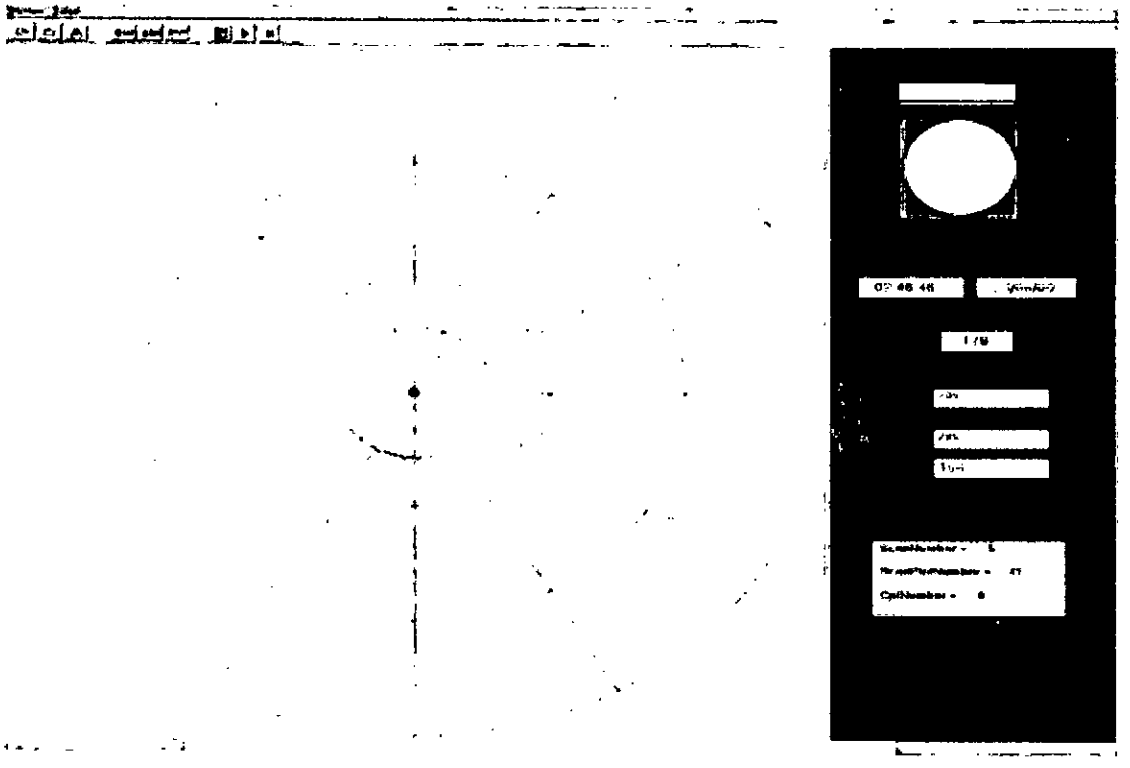


**Figure 5.8: Image du simulateur avec traitement**

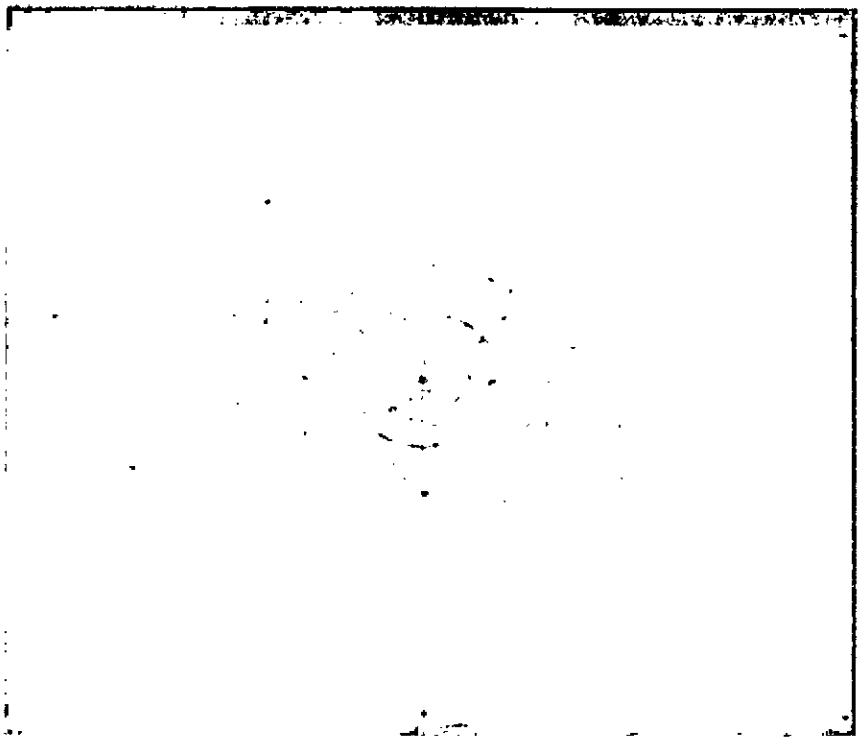


**Figure 5.7: Image du simulateur interrogation manuelle après traitement**

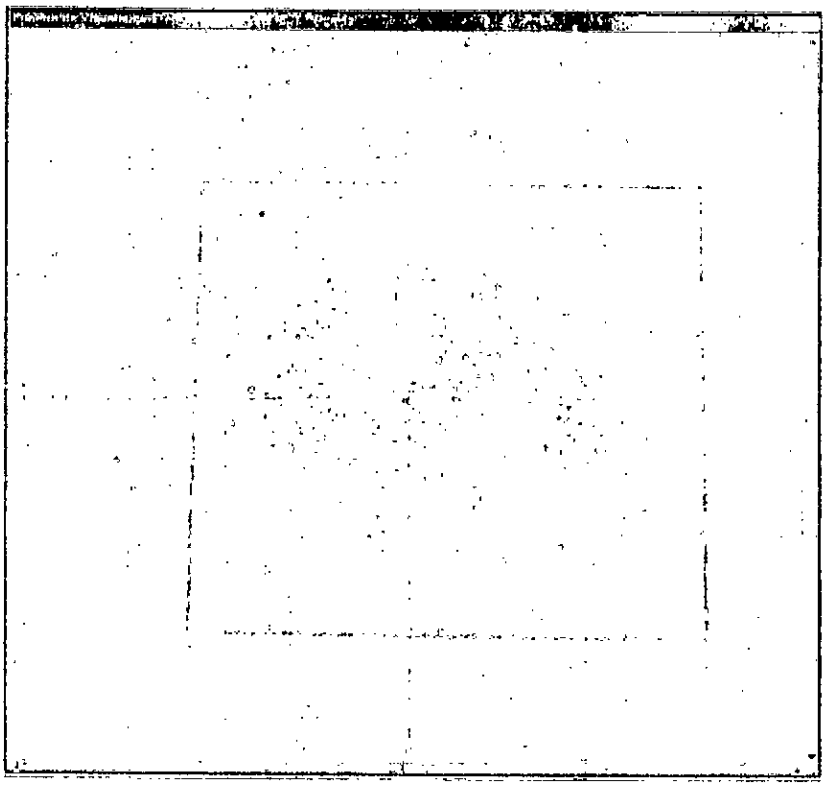


**Figure 5.9: Image du simulateur interrogation automatique avec seuil avant traitement**

## 5.6. Tests réels :

Tous les tests réels se font avec des avions qui sont équipés de répondeur adéquats et dont le code d'identification est le même que celui de la station radar.

L'identification ami ou ennemi est souvent utilisée pour le suivi pour une non détection ou mauvais temps au cours des opérations comme le montre la figure 5.10 qui montre la réponse d'un avion à 157° et 160Km avec un point en jaune ainsi que l'absence totale des faux plots sur toute la situation radar dans les indicateurs panoramiques anciens. Généralement pour un bon réglage de la station radar surtout dans notre cas le radar secondaire à chaque interrogation demandée il y a réponse de l'avion.

La figure 5.11 nous montre les réponses d'une formation de deux avions à (178° et 155Km) et (184°,160Km) avec leurs échos primaires. La figure 5.12 présente les réponses de 03 avions devant leurs échos primaires. Les coordonnées de ces avions sont (198° et 85Km) et (184°,88Km) et (203°,120Km) sauf la réponse du quatrième avion (185°,116Km) où il n'y a pas eu d'écho primaire correspondant. Ce qui signifie qu'il y a absence de détection. On remarque aussi la réponse d'un avion dont les coordonnées sont 198° et 85Km dans 03 CPIs. Cela est du au premier à la visualisation des réponses à chaque CPI, le deuxième point c'est l'ouverture du lobe du radar secondaire qui est supérieur à 4°, ainsi la cible est rayonnée durant 4/1.40625 = 2.84 CPIs.

Figure 5.11: Image des réponses d'un avion à 157° et 160Km)

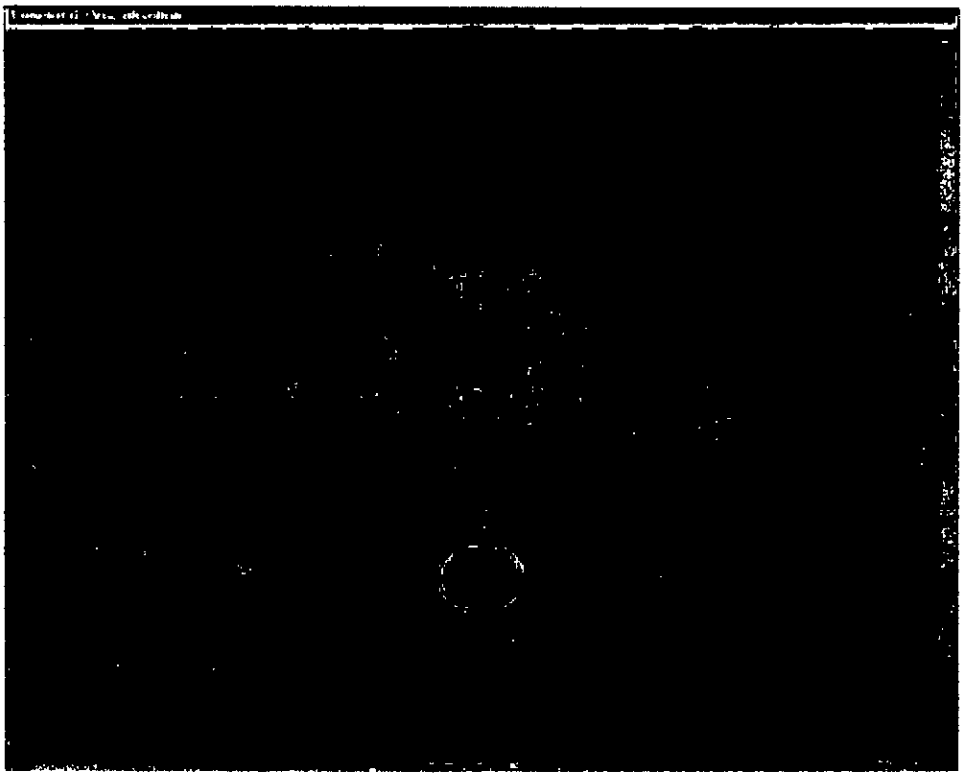


Figure 5.10 : Image des réponses de deux avions (178° et 155Km) et (184°,160Km)

**Figure 5.12: Image de des réponses de 04 avions (18 5 ° et 11 6 K m), (1 84°,88Km),(203°,120)**

# Conclusion

En conclusion nous pouvons dire que le but escompté de ce travail de thèse est de moderniser la station radar analogique en réalisant un extracteur de plots radar numérique.
Les tâches accomplies sont comme suit :
L'étude du radar existant et la compréhension du fonctionnement du système de synchronisation qui est le cœur de tout radar malgré les difficultés trouvées au niveau de la lecture de la documentation et des schémas électriques.

- ✓ Nous avons intégré deux technologies différentes analogique et numérique.
- ✓ Concevoir et implanter un algorithme d'extraction de plots radar secondaire dédier à ce radar et qui fonctionne en temps réel qui peut traiter jusqu'à 200 plots par CPI ce qui est loin pour le cas des situations réelles.
- ✓ Nous avons effectué les tests du bon fonctionnement des cartes DSP **TMS320C40,** l'évaluation et leur validation avec nos programmes écrits en l'assembleur ; côté interruptions, communication des données ( DMAs et PORTs) et enfin le noyau du DSP.
- ✓ Écriture de la procédure de communication du DSP TMS320C40 avec le Pentium en split mode (communication bi directionnelle en même temps) avec le compilateur C de Texas Instruments et le langage assembleur.
- ✓ La commande automatique par les clicks de la souris de l'interrogateur qui se faisait manuellement avec un interrupteur qu'il faut actionner au passage de l'antenne.
- ✓ Détermination automatique des coordonnées des cibles (angle et la distance) avec le déplacement de la souris alors qu'avant l'opérateur doit lire les coordonnées sur les indicateurs panoramiques et dans l'obscurité et avec beaucoup moins de précision.
- ✓ Nous avons réalisé des bancs d'essai pour le test soit de l'application d'extraction ; soit la carte génératrice à base FPGA, qui peut être utilisée pour le test en temps réel de tout algorithme de traitement de signal radar sans avoir besoin d'un radar.
- ✓ Écriture avec le Borland C++ Builder de l'application de visualisation des plots radar secondaire, en plus de l'interface graphique qui peut être amélioré beaucoup plus, mais la chose la plus importante dans l'application de visualisation est la priorité donnée à la lecture des données de la carte DSP TMS320C40 sans perte pour éviter le blocage du système malgré les précaution prise au niveau des tests des données lues.
- ✓ Conception du protocole de communication entre le Pentium et le DSP TMS320C40 au niveau de l'écriture et de la lecture.

Mais il reste toujours des travaux à faire comme la vidéo brute du signal d'identification qui peut laisser le choix à l'opérateur de prendre lui-même le décision de présence ou d'absence de la cible. Un autre point c'est d'intervenir au niveau de la RF du

récepteur du radar secondaire pour avoir une meilleure détection et diminuer les faux plots.

L'intérêt de l'application est de voir la réponse de la cible et aussi qu'il n'y ait pas d'erreur dans le calcul des coordonnées des cibles, avec une situation radar bien nettoyée et où l'aéronef répond à toute interrogation sauf évidemment le cas d'un aéronef ennemi ou un aéronef dont le répondeur est hors service ou à l'arrêt.

# Annexe

# Annexe A

# Les Processeurs de Traitement de Signal D.S.P

## A.I. Introduction:

Dans cette annexe, nous allons cité les principales générations de processeurs de traitement de signal, DSPs *« Digital Signal Processors »*. En suite, une présentation détaillée du TMS320C40 du Ti, le DSP que nous allons exploiter pour l'implémentation des algorithmes d'extraction, sera fournie.

Nous fournissons également les principales caractéristiques du TMS320C40 et les outils de développement existants. Et nous terminerons avec une conclusion et la raison pour la quelle nous avons choisi ce processeur pour notre application.

## A.2. Généralités :

### A.2.1. Historique des D.S.P.s

Sans revenir aux systèmes réalisés en logique câblée ou avec des microprocesseurs en tranches (Bit Slice Processor), on peut considérer que les premiers dispositifs « légers » de traitement furent constitués de cœurs de microprocesseurs classiques dotés de larges possibilités d'entrées / sorties. L'implantation d'algorithmes complexes, avec des contraintes de temps qui deviennent donc plus sévères si on veut travailler dans le traitement de signal ou DSP *(Digital Signal Processing)*. A titre d'exemple, les traitements effectués dans un modem " 4800 bauds " exigent un DSP de puissance de calcul de l'ordre de 1.5 million d'opérations par seconde (Mops) [3]. La satisfaction de ces deux objectifs du calcul numérique et temps réel a abouti à des architectures particulières orientées vers la performance et non l'obtention de fonctionnalités étendues ou le confort des programmeurs.

Les premiers DSP sont apparus en début des années 1980 avec le DSP de Bell Labs et le 7720 de NEC. Le premier n'a jamais été distribué en dehors de AT&T. Le S2811 d'AMI avait été annoncé plus tôt, mais sa livraison a été retardée à cause de problèmes liés à la maîtrise de la technologie VMOS. Tous ces DSPs disposaient de mémoire interne permettant la réalisation des systèmes autonomes.

Un processeur de signal existait déjà avant 1979. Il s'agissait du 2920 d'Intel qui incluait des possibilités de conversions A/D (Analog/Digital) et D/A (Digital/Analog) internes au circuit, mais ne disposait pas de multiplicateur câblé. D'autre microcontrôleurs ont été écartés de ce panorama pour la même raison.

En 1982 **Texas Instruments** introduit le **TMS32010,** premier membre de ce qui allait devenir la famille de DSPs la plus répandue, en grande partie grâce aux efforts déployés par ce constructeur pour des outils de développement aussi bien matériels que logiciels. A peu près à la même époque, la compagnie **Hitachi** introduit le premier DSP de technologie CMOS ; le HD61810. Ce fût également le premier DSP à utiliser un format virgule flottante (12 bits pour la mantisse et **4** bits pour l'exposant). Un an plus tard, **Fujitsu** fait faire un bond à la vitesse de traitement des DSPs programmables avec le MB8764 qui effectue une multiplication / accumulation en 120 ns.

En 1984, le DSP32 de Bel Labs apparaît sur le marché. Il fut le premier dans deux domaines importants : premier DSP commercialisé par AT&T, et premier DSP 32 bits virgule flottante. Peu de temps

après, NEC commercialisait le upD77230, un processeur 32 bits virgule flottante avec une multiplication accumulation en 150 ns. NEC arrive à ce niveau de performances en partie en isolant la normalisation des résultats flottants dans une instruction séparée.

Le premier processeur utilisant le format virgule flottante " IEEE-754" est le MB86232 de Fujitsu apparu en 1987. Motorola, relativement récente sur le marché des DSP, introduit son premier DSP 24 bits virgule fixe, le DSP56001, en 1987 la fin des années 1980 vit l'apparition de nombreux autres DSPs ; ils comprennent à la fois des circuits plus rapides en virgule fixe comme DSPi d'Hitachi et le DSP16A d'AT&T et d'autres processeurs virgule flottante, en particulier le TMS320C30 de TI et les DSP96001 et DSP96002 de Motorola. Les DSPs de Motorola, comme le MB82232 de Fujitsu, ou Zoran avec le 35325, utilisent de façon interne le standard flottant IEEE.

### A.2.2. Processeur dans le traitement numérique du signal :

L'utilisation de techniques numériques a nettement pris le pas sur les techniques analogiques dans de nombreux domaines, parmi les quels on peut citer :

- ✓ Le traitement du signal classique (filtrage, transformées rapides, générateur de - signaux... )
- ✓ Les télécommunications (codage – décodage, modulation – démodulation, égalisation adaptative, annulation d'écho, cryptage... ) ;
- ✓ Le traitement de la parole (codage - compression, analyse, reconnaissance, synthèse... ) et des images (codage - compression, reconnaissance de formes... ) ;
- ✓ Le radar ( poursuite multi mode, traitement anti réverbération, identification des cibles..) ;
- ✓ Les applications médicales (traitement de signaux EEG, EMG, EOG... imagerie biomédicale dans les RMN... ) ;
- ✓ La commande (industrielle, avionique...) ; etc.
- ✓ L'apparition des microprocesseurs standards, suivis des microcontrôleurs, puis des DSP *(Digital Signal Processor),* microprocesseurs dédiés au traitement du signal, a rapidement assuré l'avantage des solutions programmées sur les solutions câblées. Ces techniques présentent de nombreuses caractéristiques intéressantes parmi lesquelles
  - ➢ Une reproductibilité des traitements facilitant les tests ; La réalisation de fonctions n'ayant pas, ou du moins très difficilement d'équivalent analogique ;
  - ➢ Une facilité de modification des algorithmes et/ou des paramètres que celui-ci nécessite ;
  - ➢ L'existence d'outils de simulation.

D'un point de vue économique, on peut être assuré d'une croissance importante du marché des DSPs eu égard aux équipements dans lesquels ils sont présents :

- ✓ Les téléphones mobiles sous forme de circuits spécialisés à coeur de DSP ;
- ✓ Les modems sous forme similaire ;

- ✓ Les terminaux DSL **(Digital** Subsciber Line), HDSL, ADSL...dans lesquels ils assurent l'égalisation, le brouillagé, annulation d'écho, la suppression de la télédiaphonie, etc.
- ✓ La télévision haute définition (TVHD) (codage – décodage du son et de l'image) ;
- ✓ La radiodiffusion numérique (DABDirect Broadcast Audio);
- ✓ Les DVD (Digital Video Disks) pour le décodage son Dolby AC-3 et le décodage vidéo MPEG-2 ;
- ✓ La commande des moteurs à courant alternatif ;
- ✓ Le contrôle des disques durs : utilisation de techniques à maximum de vraisemblance (PRML Partial Reponse, Maximum Likehood) pour augmenter la densité d'enregistrement, etc.

### A.2.3. Architecture des DSPs

Tous les DSPs sont constitués des modules fondamentaux suivants:

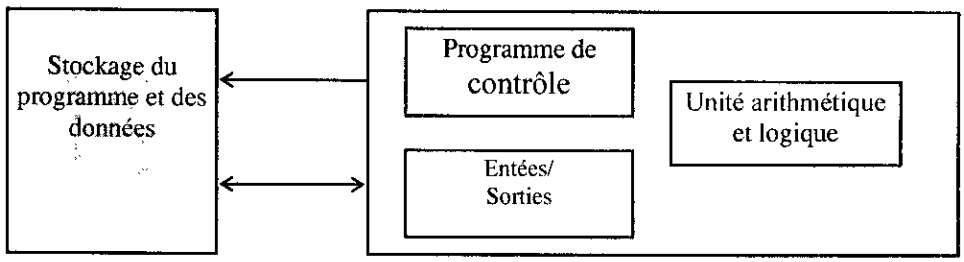
1. Le noyau du DSP qui exécute les opérations arithmétiques.
2. La mémoire de stockage des programmes (Données programme: instructions) et des données (fichiers de données qui vont subir des changements).
3. Les composants (pour quelques DSPs ) mixte pour la conversion entre les domaines analogique et numérique.

Comme le programme de la machine est stocké dans une mémoire, le processeur doit savoir ce qu'il va faire à chaque cycle d'horloge. Ainsi, le processeur cherche l'instruction et quelques données si nécessaire dans la mémoire, fait les opérations puis retourne les données manipulées pour les stocker. Cette façon de faire n'est pas la même pour tous les processeurs. Deux architectures peuvent être distinguées; l'architecture de *Von Newmann* et l'architecture de *Harvard* comme les montre la Figure A.1.

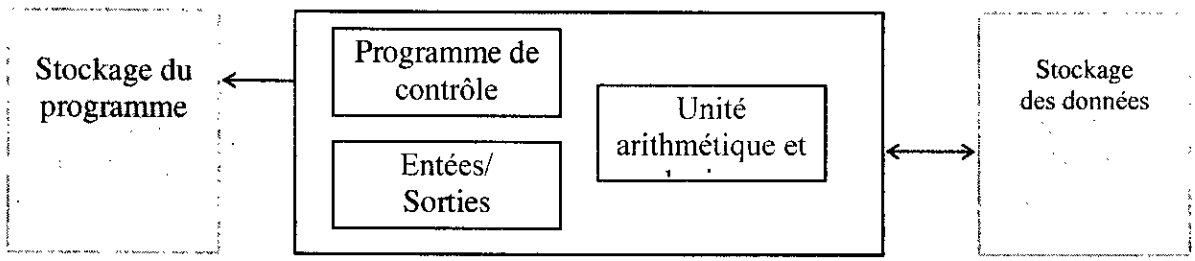
En plus de la mémoire et de la configuration des périphériques, l'application du DSP habituellement dicte le type d'architecture à utiliser.

L'**architecture de *Von Newmann*** est utilisée comme norme standard dans le développement des calculateurs depuis 40 ans. L'essentielle dans cette architecture, c'est quelle est très simple. Le programme et les données peuvent résider ensemble dans le même espace mémoire, c'est l'architecture de base qu'utilisent les processeurs comme Intel dans la gamme x86.

L'inconvénient dans cette architecture c'est qu'il y a un seul bus qui est partagé pour le transport des adresses donnée et mémoire. Pour cette raison, seulement l'espace donnée où l'espace programme peut être accédé en un seul cycle d'horloge. Lorsque la manipulation rapide des données est vitale, l'accès à la fois à la mémoire programme et à la mémoire données en un cycle est un grand avantage.




*Architecture de Von Newmann*




*Architecture de Harvard*

Figure A.1 : Architectures des Processeurs Digitaux

**L'architecture de *Harvard*** sépare les espaces mémoires programme et données. Il y a deux bus différents pour servir les deux espaces mémoires ainsi assurer l'accès au programme et aux données en parallèle, ce qui fait augmenter la vitesse de traitement. Mais pour une telle architecture il y a un coût à payer. Deux espaces mémoires, dit plus d'adresses, ce qui veut dire aussi plus de pins. Une solution à l'amiable a été trouvée et qui est un compromis entre le prix et les performances. Ainsi l'architecture de *Harvard a* été modifiée et un seul bus externe est retenu (réduction du nombre de pins), ainsi que les bus programme et données internes.

## A.3. Le DSP TMS320C40

Le DSP 'C40 est un processeur 32 bits, fabriqué en : *0.8µm, double level metal* en technologie CMOS. Il est disponible dans un package de 325-pin PGA (Pin Grid Array). La désignation des pins et les noms des signaux correspondants sont donnés dans laréférence [10].

### A.3.1. Architecture

*Architecture de Harvard:*

Le schéma block du TMS320C40 est présenté dans la Figure A.2.

***1- Unité de traitement centrale (CPU)***

Architecture à base de registres.

Le CPU comprend les composants suivants:

- *Un multiplieur (virgule* flottante/fixe) qui effectue la multiplication sur des entrées entières 32

bits / sorties 32 bits, et des entrées en virgule flottante 40 bits / sorties 40 bits en un seul cycle d'horloge.

- *Une unité Arithmétique et Logique (UAL )*

Elle effectue des opérations sur des entiers 32 bits, logiques 32 bits et flottants de 40 bits ( 32 bits pour la mantisse et **8** bits pour l'exposant ). Les résultats sont maintenus toujours sur 32 bits (entiers) ou 40 bits (flottants). Les bus internes, CPUI/CPU2 et REGI/REG2, supportent deux opérandes de la mémoire et deux autres des registres, ce qui permis des multiplications et additions/soustractions sur quatre opérandes entières ou flottantes en un seul cycle. Un registre à décalage à gauche ou à droite de 32-bits. Deux unités arithmétiques pour les registres auxiliaires (ARAUO, ARAU1) qui peuvent générer deux adresses en un seul cycle. Les ARAUs opèrent en parallèle avec le multiplieur et l'UAL.

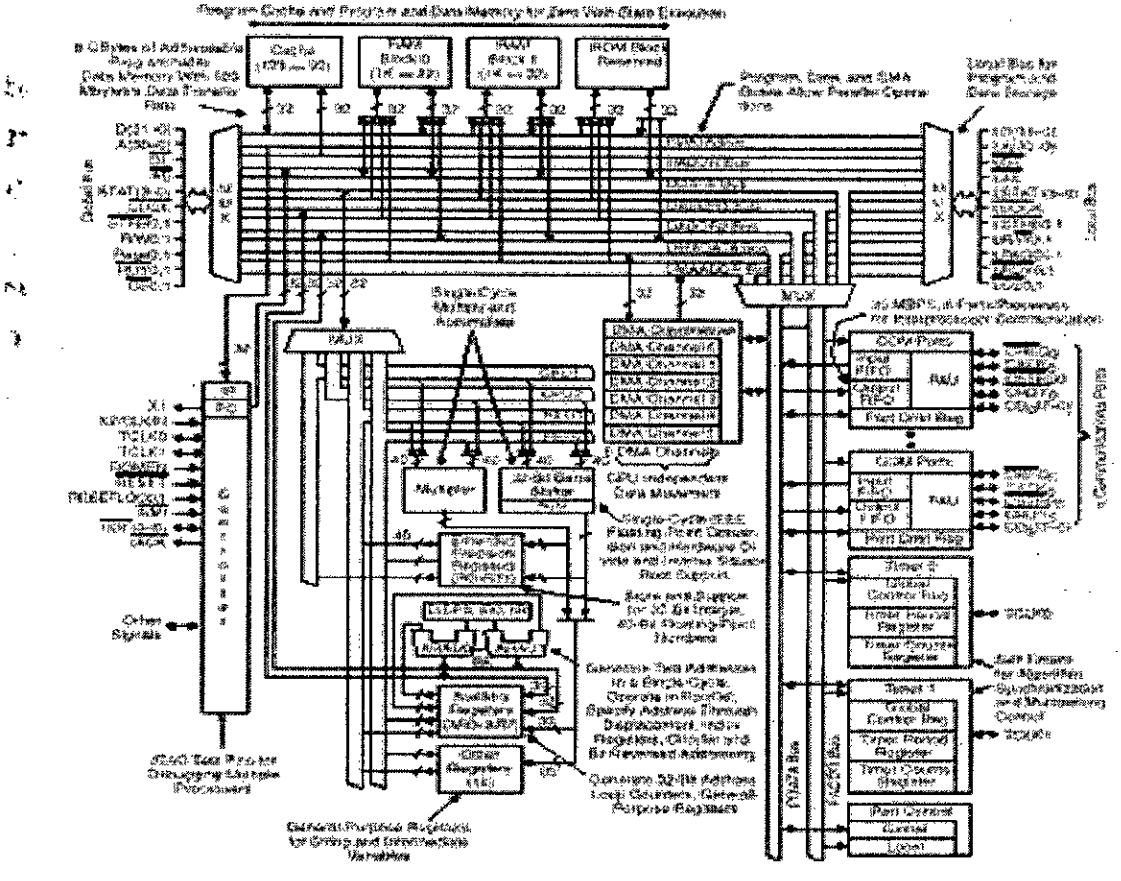


**Figure A.2 : Architecture du TMS320C40**

- *Les Registres du CPU*

  - ✓ 12 registres étendus de précision 40 bits « *RO - R11* ».
  - ✓ 08 registres auxiliaires 32 bits (adressage indirect) « *ARO - AR7* ».
  - ✓ 02 registres d'index *IRO* et *IRI.*
  - ✓ Data Page Pointer ( DP)
  - ✓ Pointeur de pile (SP).
  - ✓ Registre d'état (ST).
  - ✓ Registre de validation des interruptions des DMAs, *(DIE).*
  - ✓ Registre de validation des interruptions internes du CPU, *(I1E).*
  - ✓ Registre flag des 110F, *(11F).*
  - ✓ Registre **dimension bloc (BK).**
  - ✓ Compteur **de** répétition (RC).
  - ✓ Compteur Programme, (PC).
  - ✓ Registre adresse début répétition (RS).
  - ✓ Registre adresse fin répétition (RE).
  - ✓ Registres *IVTP* et *TVTP:* pointeurs des tables des vecteurs d'interruption et trap respectivement.

### A.3.2. Organisation mémoire

*Organisation :*

La mémoire totale du 'C40 est de **4G** mots de 32 bits (16 Gbytes). La mémoire programme, timers, ports de communication et DMAs appartiennent à cette espace mémoire. Ce qui permet aux données, code du programme, les tables et les coefficients d'être stockés dans la RAM ou la ROM.

En manipulant une pine externe (ROMEN), on peut configurer le premier Mega de la zone mémoire ( 0000 0000h à 000F FFFFh) d'être une partie du bus d'adresses local ou d'adresser la ROM lorsqu'on utilise le *boot loader.*

La Figure A.4 montre l'organisation mémoire du 'C40. Les blocks 0 et 1 de la RAM sont de (1 K de 32 bits) chacun. Le block ROM est réservé et contient le *boot loader.* Chaque block RAM ou ROM peut supporter deux accès à chaque cycle d'horloge.

Le block ROM supporte un *bootloader* qui effectue le chargement des programmes et données en reset. Le chargement peut être fait à partir des mémoires de 8, 16 ou 32 bits ou un port de communication quelconque.

*La mémoire cache* (128x32bits): pour mémoriser les sections des blocks à répéter souvent. Deux cartes mémoire sont présentées dans la Figure A.3 qui sont dépendantes du niveau du

signal à la pine externe ROMEN. Chaque carte représente les 4Gmots du 'C4x, cependant, elles diffèrent dans le premier Gigaword.

- ROMEN =1: ROM est validée à l'adresse 0000h.
- ROMEN=0 : 0000 0000h - 000F FFFFh est accessible par le bus local.

Le reste de la mémoire est le même quelque soit ROMEN.

- 2ème Megaword : réservé pour les périphériques.
- 3ème Megaword : réservé au RAM bloc0/1.
- Le reste des premiers 2 giga est réservé au bus local.
- Les 2 Gigas qui restent sont réservés au bus global.

*Mode d'Adressage mémoire :*

*1. Adressage immédiat:*

*LDI 44h, RO.* L'instruction *LDI* charge le registre *RO* avec la valeur hexadécimale *44hex. (* max. une valeur de 16 bits ).

*2. Adressage Registre :*

*LDI R1,RO .* L'instruction *LDI* charge le registre *RO* avec la valeur chargée dans *RI.*

*3. Adressage direct :*

C'est une manière d'obtenir les données a partir des locations mémoire spécifiques. .bss xyz,1

*LDP xyz*

*LDI @xyz,RO.* L'instruction *LDI* charge le registre *RO* avec la valeur stockée dans la position mémoire xyz.

*4. Adressage indirect :*

Ce type d'adressage utilise les registres auxiliaires *AR0-AR7* comme supports d'adresse.

*LDI *AR0,R1,* charger *R1* avec la valeur indiquée par *AR0.*

*LDI *AR0++,R1,* charger *R1* avec la valeur indiquée par *AR0* et incrémente *AR0.*

*LDI *AR0--,R1,* charger *R1* avec la valeur indiquée par *AR0* et décrémente *AR0.*

*LDI *AR0++( IR0),R1,* charger *R1* avec la valeur indiquée par *AR0* et incrémente *AR0* de la valeur de *IR0.*

*Instructions de base du 'C40 :*

Les instructions du 'C40 sont exceptionnellement adaptées au traitement du signal. La plupart des instructions s'exécutent en un seul cycle. Le système d'instructions contient 145 instructions organisées dans des groupes.

*Instructions parallèles :*

- ❖ Le groupe d'instructions parallèles fournit un grand degré du parallélisme possible. Ces instructions offrent les caractéristiques suivantes: Chargement parallèle de registres.
- ❖ Stockage parallèle.

- ❖ Opérations arithmétiques parallèles.
- ❖ Instructions arithmétiques/logiques utilisées en parallèle avec une instruction de stockage.

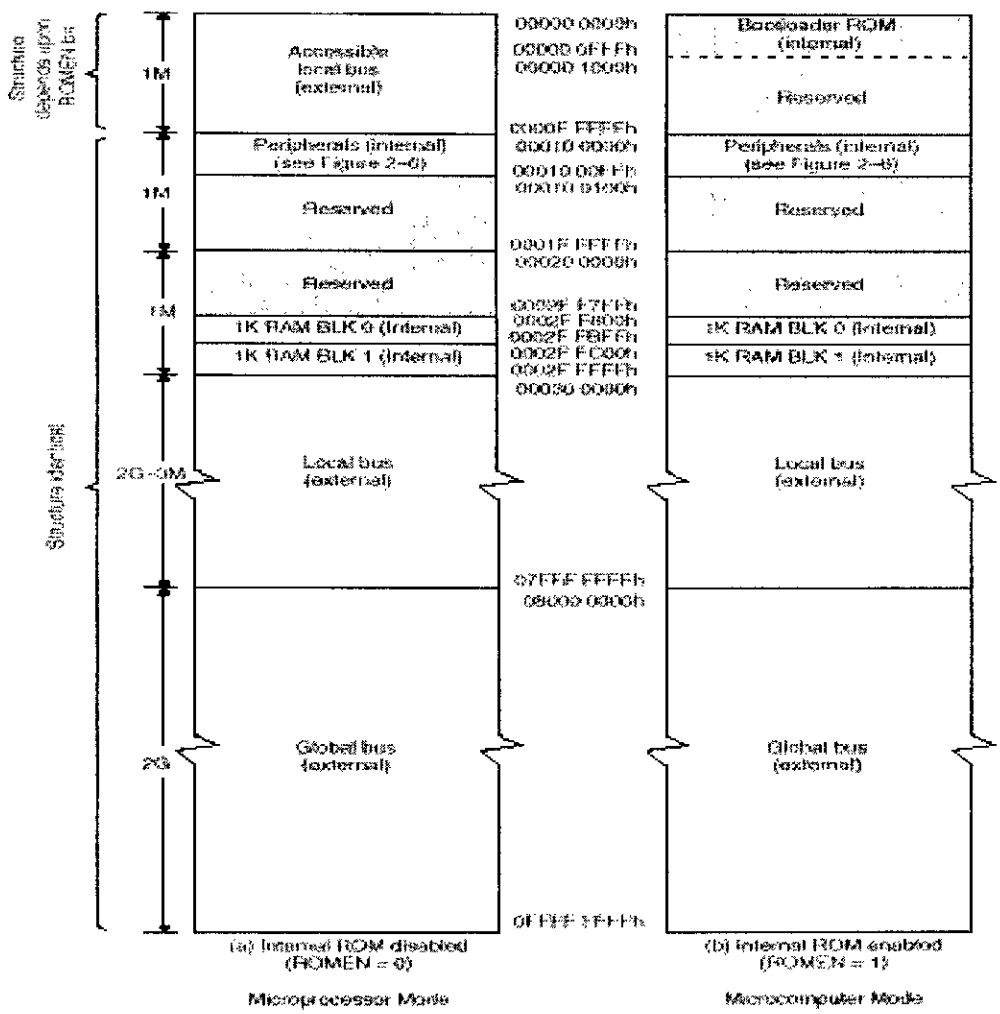



Figure A.3 : Carte mémoire du C40

### A.3.3. Interruptions :

Le 'C40 possède plusieurs interruptions internes et externes qui peuvent être utilisées. Les interruptions *internes* sont générées par les contrôleurs DMA, les timers et les ports de communication. Les cinq interruptions *externes* contiennent quatre (4) interruptions masquables (IIOFO-IIOF3) et une non-masquable qui est le NMI (Non masquable Interrupt). La Figure A.4 présente les différentes interruptions internes et externes du 'C40.

La priorité est définie en fonction de la position dans la table, l'interruption située à l'adresse la

plus basse est la plus prioritaire. On parle de priorité lorsque deux interruptions arrivent pendant le même cycle d'horloge ou lorsqu'elles sont en attente.

Quatre registres du CPU contiennent des bits utilisés pour contrôler les opérations d'interruptions.

- ST « Status register » (bit GIE: Global Interrupt Enable)
- IIE « Internai Interrupt Enable » (Timers, DMAs et ports de communication)
- IIOF( IIOFO-IIOF3)
- DIE « DMA Interrupt Enable » (DMA: ne dépend pas du bit GIE (ST)) .

### A.3.4. Les périphériques

Tous les périphériques du 'C40 sont contrôlés à travers des registres dédiés au bus des périphériques. Ce bus est composé d'un bus de données *32* bits et

d'un bus d'adresses 32 bits. Les périphériques du 'C40 contiennent deux Timers, six DMAs et six ports de communication.

***Timers :***

Les deux Timers sont des compteurs 32 bits avec deux modes de signalisation et une horloge interne ou externe. Le Timer peut être utilisé pour signaler au 'C40 ou le monde extérieur à des intervalles spécifiés, ou pour compter des événements extérieurs. Avec une horloge externe, pour signaler à un convertisseur A/D externe pour commencer la conversion, ou il peut interompre le DMA du 'C40 pour commencer le transfert de données. Avec une *horloge interne,* le Timer peut compter des événements extérieurs et interrompre le CPU après un certain nombre d'événements.

Ces Timers sont contrôlés par trois registres logés dans la mémoire réservée aux périphériques, et qui sont : *Global contrat register, Period register, et Counter register.*

***Ports de communication :***

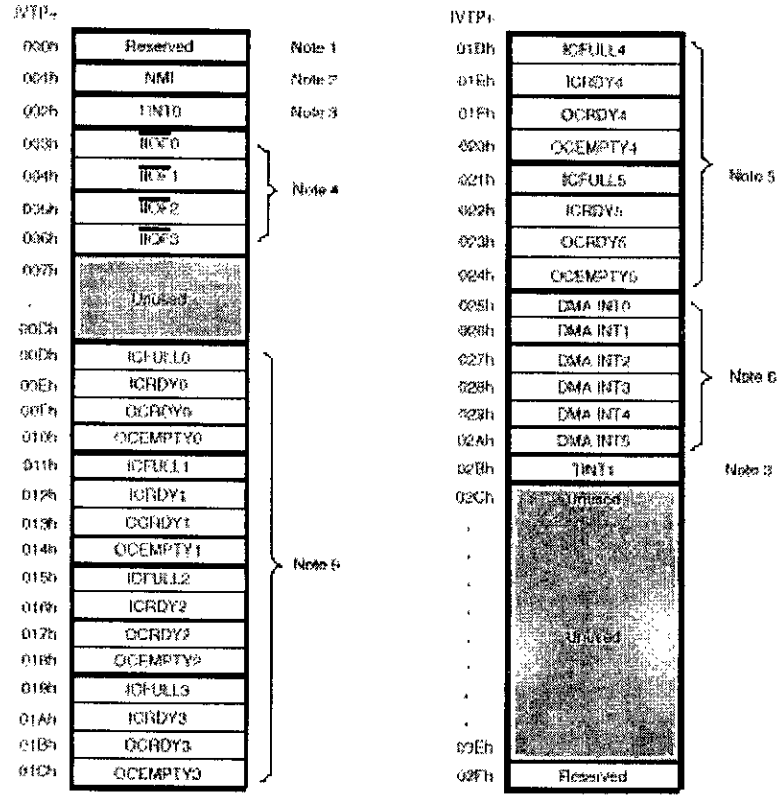
Le 'C40 possède six ports de communication bidirectionnels.

Chaque port de communication contient:

- Quatre pines de contrôle et huit pines de données. Ces douze pines fournissent une interface souple avec un autre 'C40.
- Un « fifo » du canal d'entrée ( input fifo)
- Un « fifo » du canal de sortie ( output fifo)
- Une unité d'arbitrage de port (assure la synchronisation de séquencement du transfert).
- Un registre de contrôle du port de communication (CPCR)

Pour la coordination des ports de communication avec la CPU et les DMAs, chaque port peut générer 4 interruptions; icfull, icrdy, ocrdy et oempty.

*Figure 7–2. Interrupt-Vector Table (IVT)*




Notes: 1) Reserved for the reset vector. See Table 7–4.
2) NMI (the nonmaskable interrupt) is discussed in subsection 7.4.5.
3) Timer interrupts TINT0 and TINT1 are enabled by the IIE register (subsection 3.1.9, page 3-11) and monitored at the IIF register (subsection 3.1.10, page 3-13).
4) External pins IIOF0–IIOF3 are programmed in the IIF register (subsection 3.1.10, page 3-13).
5) The communication port I/O buffers full/empty/ready interrupts are enabled by the IIE register and are also described in Figure 12–4, on page 12-8. (OUTPUT LEVEL and INPUT LEVEL bits).
6) Interrupts from the DMA are enabled at the IIE register and DMA channel contr ol register at bits TCC and AUX TCC (see Figure 11–2, on page 11-8, for bit descriptions).
7) In the 'C44, the interrupts for communication ports 0 and 3 are active. If you enable them with the IF bit, the ISR will be executed.

**Figure A.4 : Table du vecteur d'interruptions du 'C40**

Le diagramme bloc d'un port de communication du 'C40 est représenté sur la Figure A.5. Les caractéristiques des ports contiennent:

- 160 Mégabits/s bidirectionnel pour chaque port.
- Synchronisation entre la CPU ou les DMAs et les ports de communication à travers des interruptions internes et les signaux *ready*.
- Grande variété d'architectures multiprocesseurs, à savoir : Anneau, arbre, hypercube,... etc.
- Arbitrage automatique pour une connexion directe processeur à processeur ;
- Capacité de huit mots des buffers d'entrée et de sortie.

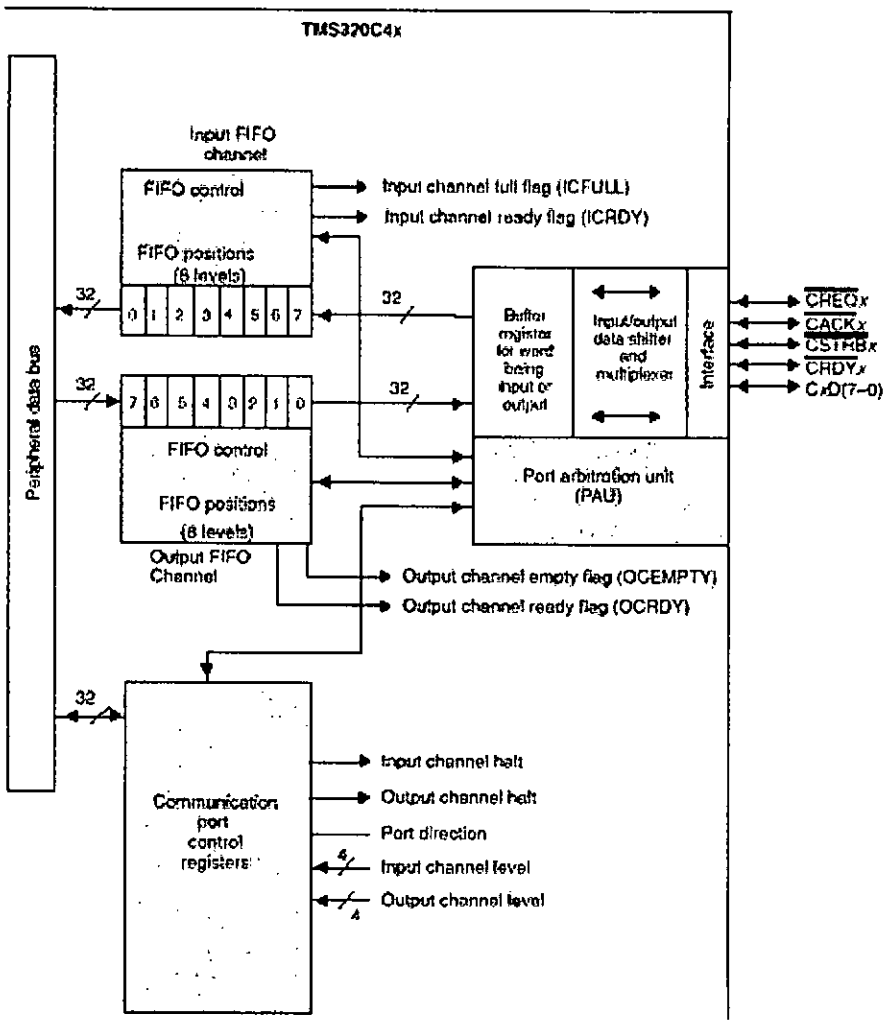



Figure A.5 : Diagramme bloc du port de communication

***Les co-processeurs DMAs :***

Le 'C40 possède 6 canaux DMA chargés du transfert de données selon les deux configurations présentées sur la Figure A.6, mode unifié « *unified mode* » ou mode split « *split mode* ».

Le DMA opère indépendamment du CPU , et il a ces propres bus d'adresses et de données pour éviter le conflits de bus .

Le DMA est programmé pour transférer les données du n'importe quelle location mémoire à une autre. Le DMA peut commencer une tâche à partir du CPU ou des interruptions externes et peut interrompre le CPU à chaque fin de tâche. Le DMA contient un registre LINK qui permet de programmer la tâche suivante sans intervention du CPU.

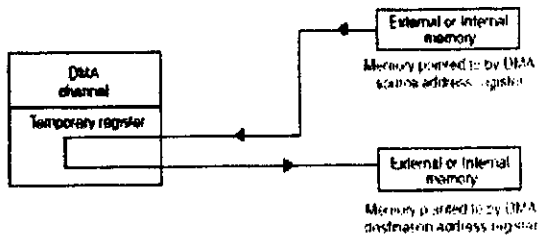
Comme chaque port de communication a la possibilité de transmettre et de recevoir, 12 canaux

DMA sont nécessaires si tous les six ports de communication sont utilisés en mode bidirectionnel.

Caractéristiques du co-processeur DMA du 'C40:

- Auto-initialisation.
- Trois opérations / cycle:
- Transfert d'une donnée 32 bit
- Réactualisation d'un registre d'adresse.
- Réactualisation du compteur du transfert.

120 MOPS (Million d'opérations par seconde).




A.3.5. Pipeline :

Le 'C40 atteint sa vitesse de traitement maximale par l'utilisation du pipeline des instructions.

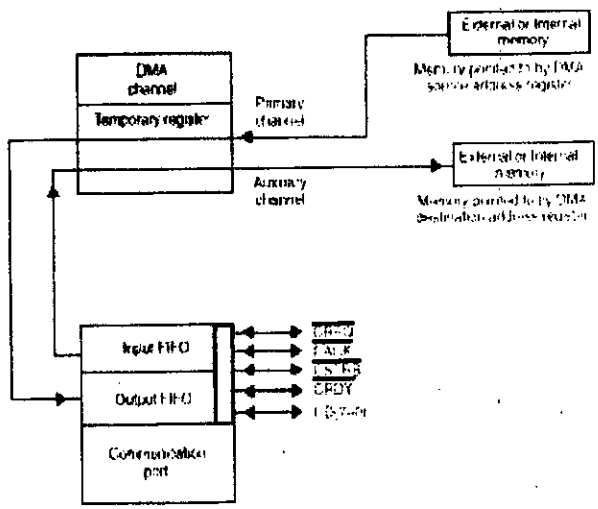



Figure A.6: Configuration de transfert des DMAs

*Structure du pipeline :*

Les cinq unités majeures du pipeline du 'C40 et leurs fonctions sont :

- Recherche « Fetch » : Chercher l'instruction dans la mémoire et réactualiser le pointeur PC.
- Décodage « Decode » : Déchiffrer le sens de l'instruction.
- Lecture « Read »: Lire l'opérande de la mémoire ( si nécessaire ).
- Exécution « Execute »: Effectuer l'opération.
- DMA coprocesseur : Lecture et écriture.

Une instruction de base a quatre niveaux : Recherche, décode, lecture et exécution. Le Tableau 3.2 illustre ces quatre niveaux du pipeline.

| Cycle | F | D | R | E |
|---|---|---|---|---|
| m-3 | W | | | |
| m-2 | X | W | | |
| m-1 | Y | X | W | |
| M | Z | Y | X | W |
| m +1 | W | Z | Y | X |

Tableau A.1 : Structure du pipeline

*Notes:*

1. *W, X, Y et Z représentent des instructions.*
2. *F, D, R, et E=Fetch, Decode, Read, et Execute respectivement.*

Les microprocesseurs standards effectuent toutes les parties d'une instruction avant d'aller à la suivante. Presque tous les DSPs chevauchent les instructions pour obtenir une meilleure vitesse. Le *chevauchement parfait* dans le pipeline est réalisé lorsque les quatre étapes s'exécutent en parallèle, ce qui est vraie au cycle m.

Pour le 'C40, les quatre niveaux du pipeline sont transparents. La présence d'un contrôleur interne intelligent assure que le code sera toujours exécuté comme il est écrit, sans anomalies dues au pipeline. Ce pipeline assure un développement facile sans prise en considération des opérations internes du processeur.

### 3.8. Outils de développement

Le développement des systèmes du traitement du signal en utilisant les D.S.P.s, est une tendance courante dans l'industrie électronique. Non pas uniquement l'existence d'une variété de chips DSP et cartes à choisir, mais aussi les outils software qui aident les ingénieurs à se lancer dans des applications plus complexes [9].

Plusieurs applications qui enveloppent l'installation d'un environnement de développement DSP. Le développement nécessite un computer host où le code est écrit et compilé, et un système DSP cible où le code est exécuté.

La Figure A.7 illustre le flux d'un développement software du TMS320 à virgule flottante. Les portions ambrées de la Figure montrent le chemin le plus commun pour un développement software, les autres portions sont en option.

Compilateur / assembleur / linker du TMS320C3x/4x :

La famille DSP TMS320 à virgule flottante du *Texas Instruments* supporte les développeurs DSP avec les outils de génération du code. Les compilateurs optimiseurs C du TI traduit ANSI - standard, et le fichier en langage C aux fichiers source en langage assembleur TMS320, qui seront l'entrée du TMS320 Assembler/linker [12].

Le TMS320 C3x/4x C compiler/Linker été désigné pour trois objectifs majeurs :

- Produire un code C compilé qui s'approche en performance du langage assembleur.
- Fournir une interface de programmation simple et accessible à l'environnement C run-time ; donc, les algorithmes DSP critiques qui demandent de grandes performances peuvent être implantés en langage assembleur. Etablissement d'un outil facile à utiliser pour le développement des applications DSP à haute performance en C.

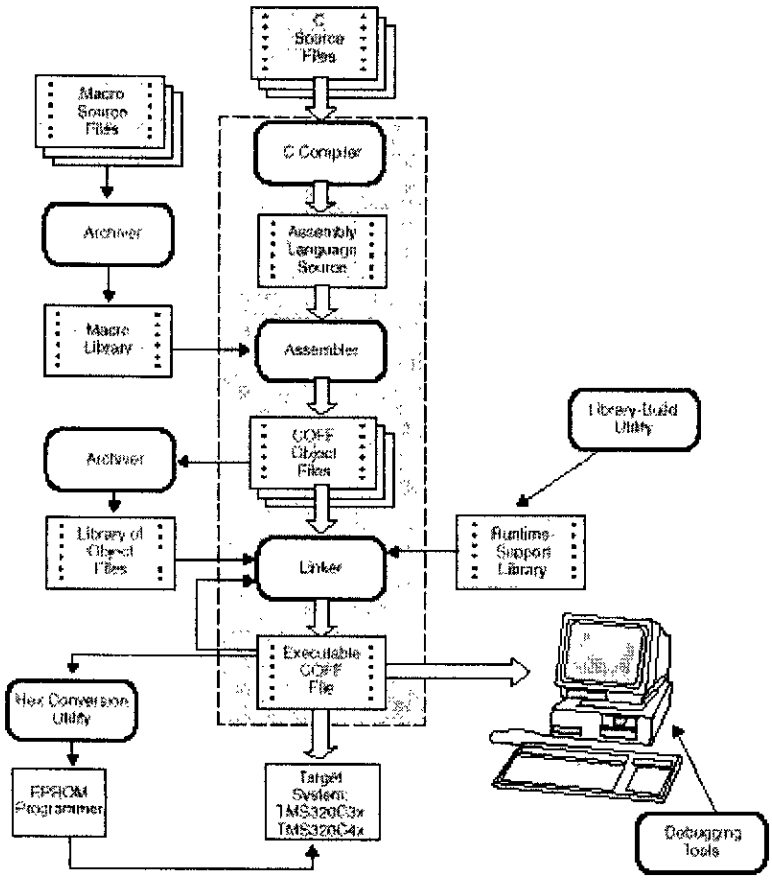



Figure A.7 : Outils de développement software du 'C40

Le compilateur C accepte le code source en C et produit un code source en langage assembleur, et forme une grande variété d'optimisation pour améliorer l'efficacité du code compilé.

L'ensemble du compilateur C contient:

➢ ***Un shell program*** qui fait appel aux programmes individuels avec les optimisations désirées. Le programme *shell* peut appeler le *parser*, *l'optimiseur*, le générateur de code, *l'interlist utility*, l'assembleur et le *linker.*

➢ ***Le parser*** lit le fichier source écrit en C, effectue des opérations de pré – traitement, vérifier la syntaxe, et produit un fichier intermédiaire qui sera utilisé comme entrée de l'optimiseur ou le générateur du code.

➢ L'***optimiseur*** lit le fichier intermédiaire généré par le ***parser*** et effectue les optimisations désirées pour améliorer la vitesse d'exécution du programme.

➢ ***Le générateur du code*** lit le fichier intermédiaire généré par le ***parser*** ou par l'***optimiseur*** et le converti en un fichier en code assembleur C4x.

➢ ***L'interlist utility*** introduit les commentaires existantes dans le code source en C dans le fichier source en langage assembleur.

➢ ***L'assembleur*** traduit le code source en langage assembleur en fichiers objet COFF en langage machine.

➢ ***L'archiver*** permet d'organiser les fichiers archives.

➢ ***Le run - time support archive*** est une collection de fichiers en C ou en langage assembleur qui implante les fonctions « ANSI standard run - time support ».

➢ ***Le peripheral control library*** est une collection de fichiers d'en – tête en C qui définissent les structures de données qui peuvent être utilisées pour accéder aux périphériques du C3x.

➢ **Le math assembly archive** est une collection de fichiers ( mathématiques ) en assembleur qui fournit une vitesse d'exécution améliorée par rapport au run – time support.

➢ **Le library-build utility** permet de construire une bibliothèque objet COFF à partir d'un archive code source.

➢ **Le linker** permet de combiner les fichiers objet COFF en un seul fichier exécutable de sortie COFF.

➢ **Le hex conversion utility** permet de convertir un fichier de sortie executable COFF en un fichier de format ASCII-Hex, Intel MSC-86, Extended Tektronix, Motorola-S ou TI- Tagged Object qui peut être utilisé comme entrée d'un programmateur d'EPROM.

Le simulateur du TMS320C4X :

Le simulateur '*C4X* est un programme qui utilise le processeur *host* et la mémoire pour effectuer la simulation des instructions niveau C du DSP C40. Il utilise le code objet produit par le marco Assembler/ linker ou le ANS ,C compiler et l'interface Debugger du TMS320 C4X standard .

Le simulateur fournit un software qui offre une capacité de debugging sur « XDS 510 » pour un 'C40 ou 'C44 plus une mémoire externe sans le Hardware DSP.

Le simulateur fournit la possibilité de développer des applications et des programmes de vérification en temps non - réel.

Chaque programme software du simulateur simule une opération 'C40 et permet le monitoring de l'état du processeur. *La vitesse de simulation typique est de l'ordre de centaines d'instructions par seconde sur un PC.* [9]

Les caractéristiques du simulateur 'C4x « communes pour tous les simulateurs TMS320 » sont:

➢ Exécution des programmes DSP sur un computer Host. Modification et inspection des registres.

➢ Modification et visualisation des données et la mémoire programme.

➢ Initialisation de la mémoire après le chargement du programme.

➢ Simulation des périphériques, caches, et timing pipeline. Extraction du cycle du timing pour l'analyse des performances.

➢ Breakpoints programmables sur :

Acquisition d'instruction, Ecriture/lecture mémoire, Conditions d'erreurs.

➢ Traces sur :

Accumulateur, Compteur programme, Registres auxiliaires, Single stepping des instructions.

Le simulateur offre les caractéristiques additionnelles suivantes pour l'interface Debugger TMS320 :

➢ Memory-mapped I/O peut être connecté à un fichier pour simuler les E/S, comme le port série synchrone. Le simulateur peut simuler les interruptions externes.

L'interface debugger du TMS320C40 :

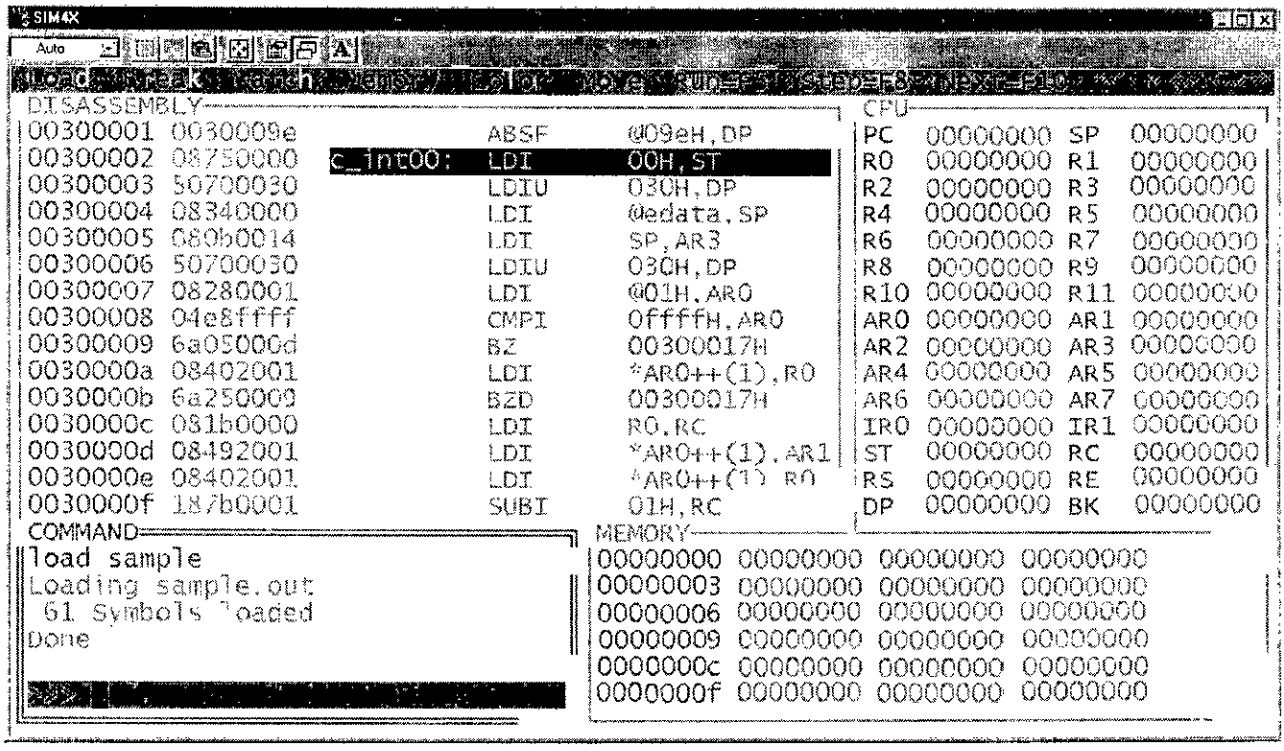



**Figure A.8 : La fenêtre de visualisation du debugger**

L'interface debugger TMS320 est utilisée avec tous les produits simulateur et émulateur et comprend le simulateur 'C4x et l'émulateur debugger 'C4x « XDS510/XD510WS ». La Figure 5.12 présente une vue générale de la fenêtre de visualisation du debugger qui est similaire à celle du simulateur.

Le debugger permet d'exécuter ou arrêter le processeur ; visualiser et modifier les registres, les valeurs de la mémoire et les variables en C, et visualiser l'assembleur et la source en C. Les caractéristiques du debugger sont :

➢ Possibilité de lancer ou arrêter le 'C4x avec single step, step over et run.

➢ Une fenêtre CPU qui visualise les valeurs des registres du DSP.

➢ Une fenêtre *watch* qui visualise les valeurs des variables en format désiré.

➢ Fenêtres de visualisation des vecteurs.

- Possibilité de changer les valeurs de données dans n'importe quelle fenêtre.
- Possibilité de définir des points d'arrêt software « break points » dans l'assembleur ou dans la source en C avec un click avec la souris.

preparation d'un programme DSP :

La Figure 5.13 illustre les étapes nécessaires pour préparer un programme pour le Debugging.

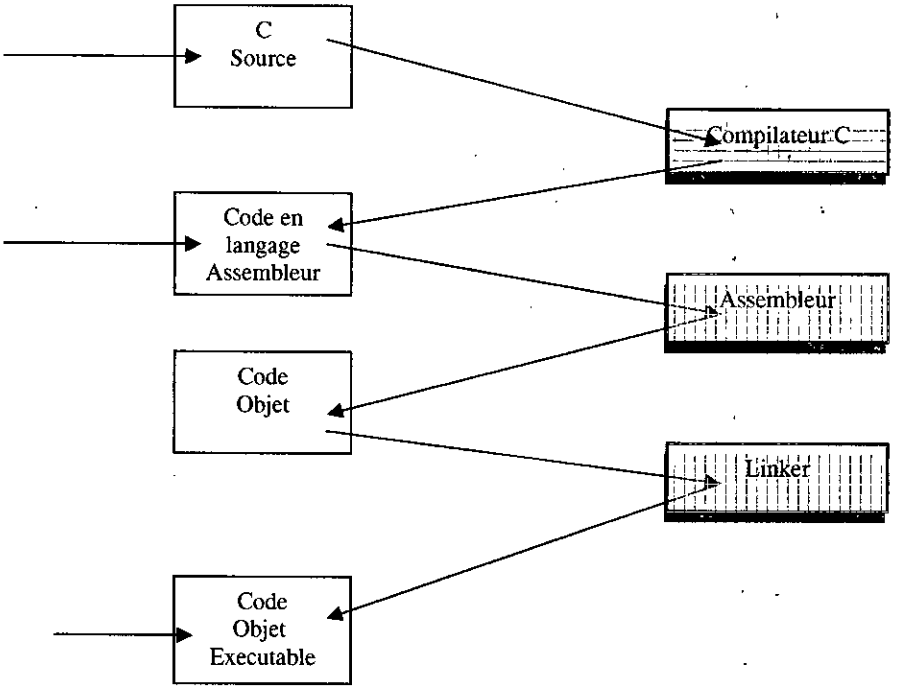



**Figure A.9 :** Préparation d'un programme DSP

Si on prépare un programme écrit en langage C:

- Compiler le programme en utilisant les options *–g* et *v40*.
- Assembler le programme assembleur résultant.
- Linker le fichier Objet résultant.

Ce qui produit un fichier *objet* qui peut être chargé dans le Debugger.

Si on prépare un programme écrit en assembleur:

- Assembler le fichier assembleur source.
- Linker le fichier objet résultant.

Le debugger peut être provoquée en utilisant la commande :

***Emu4x*** pour l'emulateur, ou ***Sim4x*** pour le simulateur.

# Annexe B

# Bus VME

## B.1. Introduction au VMEBUS :

VMEbus est une architecture d'un système informatique. Le terme 'VME' est présenté par l'Eurocard VERSAMODULE et a été d'abord inventé en 1980 par le groupe de fabricants qui l'ont défini. Ce groupe était composé des gens de Motorola, Mostek et les sociétés Signetics qui coopéraient pour définir la norme. Le terme 'bus' est un terme générique décrivant un chemin de données d'ordinateur, de là le nom VMEbus.

D'autres définitions largement employées sont VERSABUS-E, VERSAmodule l'Europe et l'Européen VERSAMODULE. Cependant, le terme 'Eurocard' a tendance à aller mieux, comme VMEbus était à l'origine une combinaison de la norme électrique de VERSABUS et le facteur mécanique de l'Eurocard. La figure.B.1 montre un rack VME avec les alimentation +5V,+12V et -12V alors que la figure 2 montre la carte host XVME 653 équipée d'un Pentium II 200MHz et 64 Moctets de RAM.

VERSAbus a été d'abord défini par la Société Motorola en 1979 pour son microprocesseur 68000. Au commencement, il a rivalisé avec d'autres bus comme le Multibuste, STD l'Autobus, S-100 et le Q-autobus. Cependant, il est rarement employé.

Puisque beaucoup de travail a été déjà fait sur VERSABUS, il a été utilisé comme une structure pour la nouvelle norme. De plus, une norme mécanique basée sur le format d'Eurocard a été choisie. L'eurocard est un terme qui décrit approximativement une famille de produits basés autour du VACARME 41612 et IEC 603-2 standards de connecteur, l'IEEE 1101 standards de conseil de PC et le VACARME 41494 et IEC 297-3 standards de support. Quand VMEBUS a été développé, le format d'Eurocard avait été bien établi en Europe pendant plusieurs années. Un grand volume de matériel mécanique comme des cages de carte, des connecteurs et des sous-supports étaient disponible. Les pins et les supports de connecteur sont plus élastiques à l'usure mécanique que des connecteurs à circuit imprimés.

Le mariage de la spécification VERSABUS électrique et le format d'Eurocard a abouti à VMEBUS version A. Il a été sorti en 1981. La spécification VMEBUS a depuis été raffinée par des révisions B, C, C.1, IEC 821, IEEE 1014-1987 et ANSINITA 1-1994. L'ANSI, VITA, IEC et des standards IEEE sont importants pour la publicité de VMEBUS.

Depuis son introduction, le VMEbus a fournit des milliers de produits et a attiré des centaines des fabricants de cartes, de matériel mécanique, de logiciel et de chips d'interface de bus. Il continue à cultiver et soutenir des demandes (applications) diverses comme des commandes industrielles, militaires, des télécommunications, la bureautique et des systèmes d'instrumentation.

## B.2. Caractéristiques du VMEBUS :

La table B.1 montre les caractéristiques générales du VMEBUS sous les spécifications de VME64 et de VME64X.

**Table B.1. General VMEbus Features**

| Item | Specification | Notes |
|---|---|---|
| Architecture | Master/slave | |
| Transfer Mechanism | Asynchronous, with both multiplexed and non-multiplexed bus cycles. | There is no central synchronization clock. |
| Addressing Range | 16, 24, 32, 40 or 64-bit | Address path width selected dynamiclly |
| Data Path Width | 8, 16, 24, 32 or 64-bit | Data path width selected dynamiclly |
| Unaligned Data Transfers | Yes | Compatible with most popular microprocessors. |
| Error Detection | Yes | Using BERR* signal. |
| Parity Protection | No | There are no parity signals on the backplane, but parity protected boards are quite common. |
| Data Transfer Rate | 0 - 500+Mbyte/sec | |
| Interrupts | 7 levels | Priority interrupt system with 8, 16 or 32-bit STATUS/ID (interrupt vector). |
| Multiprocessing Capability | 1 - 21 processors | Flexible bus arbitration with true peer-to-peer multiprocessing. |
| System Diagnostic Capability | Yes | Using SYSFAIL* signal and VME64x test & maintenance bus. |
| Live Insertion Capability | Yes | Using optional standards. |
| Control & Status Registers (Plug & Play Support) | Yes | Under VME64 & VME64x |
| Mechanical Standard | 3U single-height Eurocard<br>6U double-height Eurocard<br>9U (optional standard) | 160 x 100 mm Eurocard<br>160 x 233 mm Eurocard<br>367 x 400 mm Eurocard |
| User Defined I/O | Yes | Through the Front Panel and P2/J2 User defined Pins |
| Maximum Number of Card Slots in Backplane | 21 | The number of cards is limited by how many boards, located on 0.8" centers, can be placed into a 19" rack panel. |

## B.3. Applications :

VMEbus est utilisé dans des applications très variées. Dans beaucoup de cas, le design d'un système VMEbus est conçu pour des applications spécifiques. Parmi les applications courantes il y a:

- Les contrôles industriels.
- Militaire:systèmes de contrôle des radars terre ou air, communications, avionique et beaucoup autres.
- L'aérospatiale: avioniques, systèmes de contrôle des vols par fil, contrôle de tests *du* vaisseau spatial, séquenceurs du compte à rebours du missile, et beaucoup autres. En 1998 l'Éclaireur de Mars a utilisé un ordinateur VMEbus pour contrôler l'opération du vaisseau spatial sur la planète Mars.
- Le contrôle du transport: ferroviaire.
- Telecom: équipement de commutation téléphonique, station pour téléphone cellulaire, uplink du satellite et down links et de commutation téléphone.
- La simulation: simulateur de vol, tremblement de terre et plusieurs systèmes de la simulation militaires.
- Médical: Imagerie CATSCAN, imagerie MRI et plusieurs systèmes acoustiques.

## B.4. Présentation du Rack et des cartes VMEbus :

La figure B.1 présente une image d'un châssis VME. Au niveau électrique, les valeurs des alimentations sont fixées à (+5V, + et -12V).

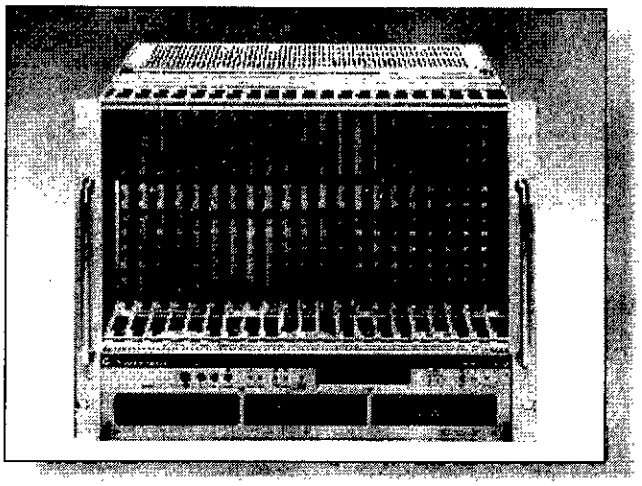


Figure B.10 : Châssis VME

Ce châssis VME comporte 21 emplacements pour des modules électroniques. Du point de vue informatique, il s'agit réellement d'un bus informatique. Des processeurs résident dans le VME et les cartes peuvent être maîtres ou esclaves. Le bus VME est un bus asynchrone, non multiplexé de 32 bits

d'adressage et 32 bits de données. Des extensions lui permettent d'aller jusqu'à 64 bits.

Le principe de base est une relation "maître-esclave". Un maître est un module actif (processeur, contrôleur d'interruption,...) qui transfère des données à destination ou en provenance d'un module passif appelé esclave (carte mémoire, carte d'entrée / sortie,...). L'esclave (ou la partie esclave d'un module) répond quand son adresse est reconnue sur le bus.

La figure B.2 présente une image de la carte host qui est équipée d'un Pentium II 200MHz avec 64Moctets implanté dans un environnement PC et interfacé à un bus VME via le circuit Universe de TUNDRA. Cette carte est insérée dans le rack VME sur le slot 1 qui est configuré comme maître et dans le reste des slots les cartes DSP TMS320C40 (voir la figure.B.3.) comme esclaves. Le Pentium de la carte host inspecte les FIFOs des cartes DSP s'il y a des données à lire et écrit aussi dans les FIFO des cartes DSPs si le Pentium veut envoyer des données vers le DSP TMS320C40 [14] [15].

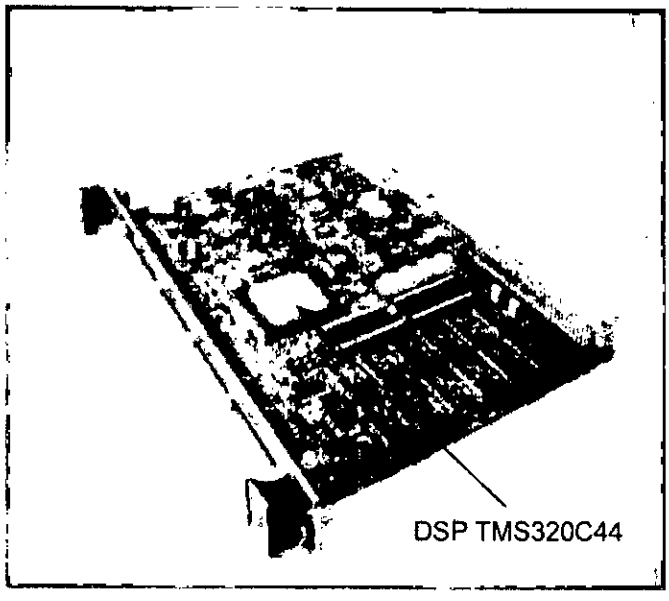



Figure B.3 : Carte DSP TMS320C44

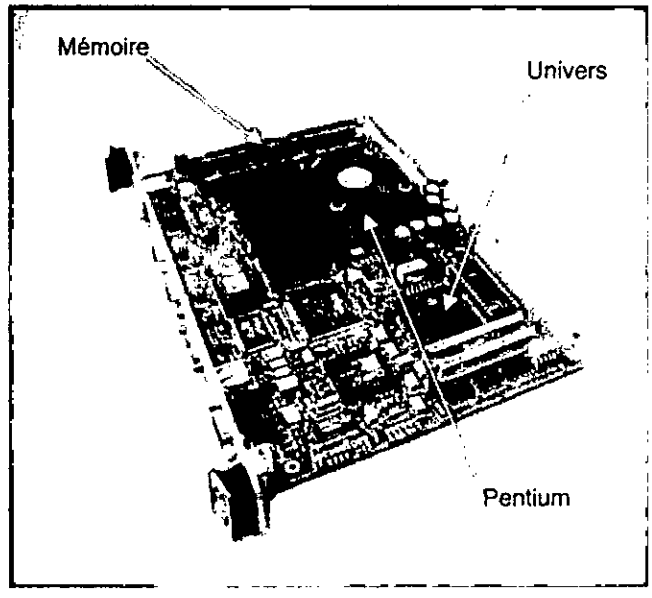



Figure .B.2 : Carte Host (Pentium II) XVME 653

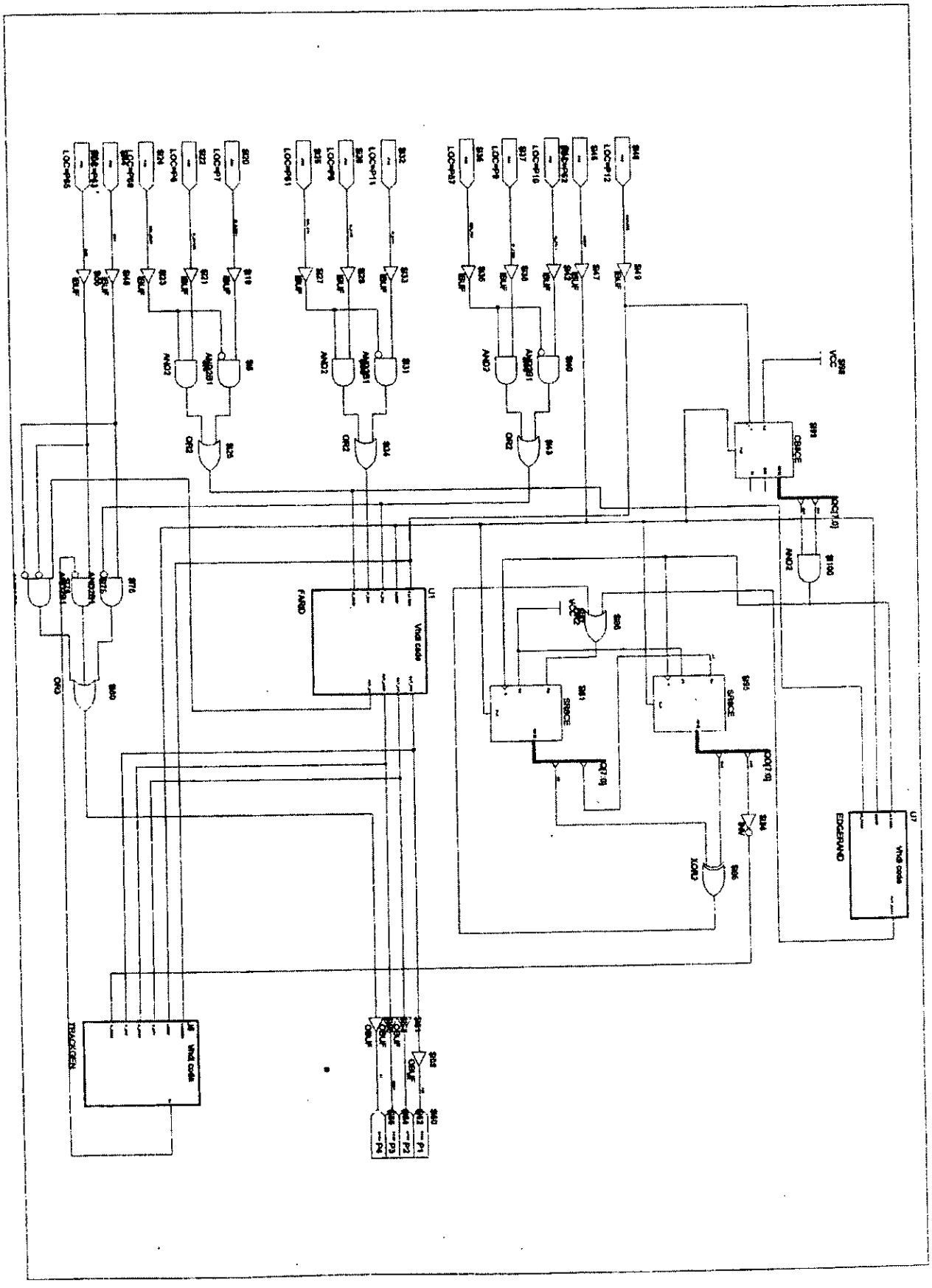
# Annexe C

```
****************************  Device Pin Out *****************************

Device : XC95108-7-PC84


                    I   I
          I  I  I   N   N  I
          N  N  N       _  N
          _  _  _   I   I  _
          C  P  P   N   O  C     N
          P  R  R G R   R  P  I  O  C  P  T  T  T  T  T  T  V  T  T  T
          I  F  F N D   D  I  F  R  P  R  I  I  I  I  I  I  C  I  I  I
          1  1  2 D 1   2  2  F  D  I  F  E  E  E  E  E  E  C  E  E  E
          ------------------------------------------------------------
         /11 10  9  8  7  6  5  4  3  2  1 84 83 82 81 80 79 78 77 76 75 \
CLK1MHZ | 12                                                          74 | TIE
    TIE | 13                                                          73 | VCC
    TIE | 14                                                          72 | TIE
    TIE | 15                                                          71 | TIE
    GND | 16                                                          70 | TIE
    TIE | 17                                                          69 | TIE
    TIE | 18                                                          68 | TIE
    TIE | 19                                                          67 | TIE
    TIE | 20                                                          66 | TIE
    TIE | 21                      XC95108-7-PC84                      65 | INR2
    VCC | 22                                                          64 | VCC
    TIE | 23                                                          63 | INR1
    TIE | 24                                                          62 | RESET
    TIE | 25                                                          61 | SEL_CPI
    TIE | 26                                                          60 | GND
    GND | 27                                                          59 | TDO
    TDI | 28                                                          58 | SEL_NORD
    TMS | 29                                                          57 | SEL_PRF
    TCK | 30                                                          56 | TIE
    TIE | 31                                                          55 | TIE
    TIE | 32                                                          54 | TIE
        \ 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 /
          ------------------------------------------------------------
          T  T  T  T  T  V  T  T  T  G  T  T  T  T  T  T  G  T  T  T  T
          I  I  I  I  I  C  I  I  I  N  I  I  I  I  I  I  N  I  I  I  I
          E  E  E  E  E  C  E  E  E  D  E  E  E  E  E  E  D  E  E  E  E


Legend : TIE  = Tie pin to GND or board trace driven to valid logic level
         VCC  = Dedicated Power Pin
         GND  = Dedicated Ground Pin
         TDI  = Test Data In, JTAG pin
         TDO  = Test Data Out, JTAG pin
         TCK  = Test Clock, JTAG pin
         TMS  = Test Mode Select, JTAG pin
  PROHIBITED  = User reserved pin
```



```
XACT: version C.16                                    Xilinx
Inc.
                                        Fitter Report
   Design Narre: allou
Fitting Status: Successful          Date: 1-21-2001, 2:41AM

***************************** Resource Summary ************

Design          Device              Macrocells      Product
Terms Pins
Narre           Used                Used            Used
Used
allou           XC95108-7-PC84      80 /108 ( 74%) 244/540
45%)   17 /69   ( 24%)

PIN RESOURCES:

Signal Type     Required      Mapped |Pin Type Used
Remaining

Input               12           12       I/O
14        49
Output               1            1    1 GCK/IO
3         0
Bidirectional :      3            3       GTS/IO
0         2
GCK                  1            1       GSR/IO
0         1
GTS                  0            0
GSR                  0            0    I

         Total      17           17

GLOBAL RESOURCES:


Signal 'CLKlMHZ' mapped onto global clock net GCK3.
Global output enable net(s) unused. Global set/reset
net(s) unused.

POWER DATA:
```

There are 80 macrocells in high performance mode (MCHP). There are 0 macrocells in low power mode (MCLP). There are a total of 80 macrocells used (MC).

# Bibliographie

[1] H. Mehalli et F. Settou, " Détecteur à taux de fausse alarme constante (TFAC/CFDAT)" Projet fin d'étude, juin 1995.

[2] M. Hammadouche, M. Barkat, M. Khodja "Analyse of the clutter map CFAR in Weibull clutter" Signal Processing 80, 2000, pp 117-123.

[3] Technique d'ingénieur, traité d'électronique.

[4] H.W. Cole "Understanding Radar ", Blackwell Scientific Publications, Oxford.

[5] Description technique de la station radar ASRx.

[6] Description technique de l'interrogation IFF de la station radar ASRx.

[7] L. Thourel "Les radars de veille modernes", Sofdal et Masson Cie, Paris,1965.

[8] M.I. Skolnik "Radar handbook", Mc Graw Hill, Inc, 1970.

[9] Texas Instruments, Inc, "TMS320 Floating-Point DSP Assembly language tools User's guide" (literature number SPRU035B).

[10] Texas Instruments, Inc, "TMS320C4x User's Guide" (literature number SPRU063).

[11] Texas Instruments, Inc, "Parallel Processing with the TMS320C4x" (literature number SPRA031).

[12] Texas Instruments, Inc, "TMS320 Floating-Point DSP Optimizing C Compiler User's Guide" (literature number SPRU034).

[13] Texas Instruments, Inc, "TMS320 Family Development Support Reference Guide" (literature number SPRU011).

[14] XYCOM,Inc,"XVME-653 Single-Slot VMEbus Pentium MMX Processor Module", XYCOM 1998.

[15] VMEbus Interface Components Manual, TUNDRA, Spring 1996.

[16] W.D. Peterson "VMEbus Handbook", Edition 4,1997.

[17] "IFF DSPVME Card User's Guide".

[18] "DET DSPVME Card User's Guide".

[19] Xilinx, Inc., "The Programmable logic Data Book," Xilinx, 2100 Logic Drive, San Jose, CA 95124, USA, 1998.

[20] Atmel Corporation, Integrated Circuit Data Book, Atmel Corporation, San Jose, CA, USA, 1997.

[21] Lucent Technologies, "ORCA OR3Cxx and OR3Txxx Series Field-Programmable Gate Arrays Data Sheet," Lucent Technologies, Microelectronics Group, 1997.

[22] Actel, ACT Family FPGA Databook, Actel Corporation, Sunnyvale, CA, USA, 1997.

[23] Altera, Databook, Altera Corporation, San Jose, CA, USA, 1997.

[24] H. Schmit, "Incremental Reconfiguration for Pipelined Applications," Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, CA, 1997.

[25] W. Luk, N. Shirazi and P.Y.K. Cheung, "Modelling and Optimising Run-Time Re-configurable Systems," Proceedings of the IEEE Symposiumon FPGAs for Custom Computing Machines, Napa, CA, 1996.

[26] S. Guccione, "List of FPGA-based Computing Machines," Personal work,

http://www.io.com/~guccione/HW_list.html, 1997.

[27] S. Ludwig, "Hades — Fast Hardware Synthesis Tools and a Reconfigurable Co-processor," PhD Thesis, ETH, Zürich, 1997.

[28] E. Lemoine and D. Merceron, "Run Time Reconfiguration of FPGA for Scanning Genomic Databases," Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines, Napa, CA, 1995.

[29] J.-C. Michelou et Cyril Müller, "Espion de bus PCI sur carte coprocesseur reconfigurable," Rapport de stage, Ecole Polytechnique, 1997.

[30] J.D. Hadley and B.L. Hutchings, "Design Methodology for Partially Reconfigured Systems," IEEE Symposium on FPGAs for Custom Computing Machines, Napa, CA, 1995.

[31] P. Lysaght and J. Dunlop, "Dynamic Reconfiguration of FPGAs," More FPGAs: Proc. 1993 Intl. Workshop on Field-Programmable Logic and Applications, 1993.

[32]L. MOLL," Applications des Mémoires Actives Programmables," Thèse doctorat, École Polytechnique, 1997.

[33] Motorola, "MPA1000 Programmable Arrays Data Sheet," Motorola Semiconductor Technical Data, 1997.

[34] S. C. Wong, H. C. So, J. H. Ou and J. Costello, A 5000-gate CMOS EPLD with multiple logic and interconnect arrays,in IEEE 1989 Custom Integrated Circuits Conference, paper 5.8, 1989.

[35] G.V. Trunk, "Range Resolution of targets using automatic Detectors" Naval Research Laboratry, 1978.

[36] M. C. Stevens "Secondary surveillance radar", Artech House, London, 1988.

[37] "Rapport technique MTD II", Lincoln Laboratry, MIT, Lexington, MA.

[38] Scott, B.Shonnon,F.Font : Visual Basic; Secret d'experts, Simon & Schuster Mac millan, Paris, 1996.

[39] "Langage C++ ",Simon & Schuster Mac millan, Paris, 1996.

[40] "Borland C++ Builder".

[41] F. de Coulon, "Théorie et traitement des signaux".

Site Internet consultés :

http://www.ti.com. (DSP TMS320C4x)

http://www.xilinx.com. (FPGA )

http://www.vita.com. (Bus VME)

**ملخص:**
العمل المقدم في هذه المذكرة يدخل في إطار تطوير محطة رادارية. في هذا العمل قمنا بانجاز نظام لاستخراج الاقترة (النقاط) الرادارية الثانوية التي يكون فيها عملية ترقيم إشارة تعارف صديق عدو, استخراج الاقترة و عرضها. عمليات الترقيم الاستخراج مثبتة على بطاقة (DSP TMS320C40) على VMEBUS. عملية العرض مكتوبة بلغة (Borland C++Builder) بنتيوم تحت نظام التشغيل ويندوز. بنتيوم مشكل و موضوع كقائد في نظام لاستخراج و تكمن مهمته في قراءة المعطيات الرادارية المبعوثة من بطاقة (DSP TMS320C40), الكتابة في هذه البطاقة (DSP TMS320C40) المعطيات الرادارية في إحداثيات الأهداف المعينة للاستجواب و الحالة الوظيفية للرادار في وقت حقيقي.

**كلمات مفتاحيه:** تعارف, DSP TMS320C40, مستخرج الاقترة, MTD-II, Bus VME

**Résumé**

Le travail présenté dans cette thèse rentre dans le cadre de modernisation d'une station radar. Dans ce travail nous avons réalisé un système d'extraction de plot radar secondaire dont la tâche est la numérisation du signal d'identification ami ou ennemi, extraction des plots et la visualisation de ces plots. Les fonctions de numérisation et d'extraction sont implantées sur une carte DSP TMS 320C40 sur bus VME. La fonction de visualisation écrite avec le langage Borland C++Builder est implantée sur un processeur Pentium sous Windows. Le Pentium est configuré en maître dans le système d'extraction et il a pour rôle la lecture des données radar envoyées par la carte DSP TMS320C40, l'écriture dans la carte DSP TMS320C40 des données sur les coordonnées des cibles à interroger et l'état de fonctionnement du radar en temps réel.

**Mots Clés :** DSP TMS320C40, Extracteur de Plot, MTD-II, IFF et Bus VME

**Abstract:**

Work presented in this thesis goes in the setting of modernization of a radar station. In this work we achieved a system of secondary radar plot extraction whose task is the digitalization of the identification friend or foe signal, extraction of plots and the visualization of these plots. Functions of digitalization and extraction are implanted on a DSP card TMS320C40 on VME bus. The written visualization function with the language Borland C++Builder is implanted on a processor Pentium under Windows. The Pentium is configured in master in the system of extraction and it has for role the reading of the data radar sent by the card DSP TMS320C40, the writing in the card DSP TMS320C40 of data on co-ordinates of targets to interrogate and the state of working of the radar in real time.

**Key words:** DSP TMS320C40, Plot Extractor, MTD II, IFF and Bus VME.