Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

THESE

Présentée

À L'UNIVERSITE DE TLEMCEN
FACULTÉ DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE ET ELECTRONIQUE

Pour l'obtention du diplôme de

DOCTORAT ES SCIENCES

Spécialité : "Electronique Biomédicale"
Option: Télémédecine

Par

NEMMICHE Ahmed

ETUDE ET REALISATION D'UNE PLATEFORME DEDIEE A LA PRATIQUE TELE MEDICALE SOUS ARCHITECTURE DE COMMUNICATION LOCALE USB-HID ET DISTANTE TCP-IP

Soutenue le 14 janvier 2016 devant le Jury:

A. BESSAID	Professeur à l'Université de Tlemcen	Président
B. SALGUES	Professeur à l'université de Montpellier II	Examinateur
G. NASSAR	Professeur à l'université de Valencienne	Examinateur
M. BENABDELLAH	Professeur à l'Université de Tlemcen	Directeur de Thèse

ETUDE ET REALISATION D'UNE PLATEFORME DEDIEE A LA PRATIQUE TELE MEDICALE SOUS ARCHITECTURE DE COMMUNICATION LOCALE USB-HID ET DISTANTE TCP-IP

Auteur : NEMMICHE Ahmed
Prof. Responsable : BENABDELLAH Mohammed
Sujet proposé au sein du labo GBM

Dédicaces

A ma famille;

Mes collègues;

Et tous mes amis.

Remerciements

Je tiens à exprimer toute ma gratitude et mes remerciements à monsieur Mohammed BENABDELLAH, qui m'a par son assiduité, son engagement, son soutien et sa présence constante dans toutes les conditions accompagné dans ma formation, jusqu'à pouvoir exposer ma présente thèse. Le travail que voici n'aurait pas été possible sans le concours du groupe de Télémédecine sous la direction de monsieur Mohammed BENABDELLAH à savoir: le laboratoire de recherche du Génie Biomédical relevant du Département de Génie Électrique et Électronique de la Faculté de Technologie auprès de l'Université Abou-bekr Belkaïd de Tlemcen, qui m'ont facilité les recherches.

je continue par souligner toute ma reconnaissance aux membres du jury qui ont bien voulu assister à mon exposé et dont Monsieur Abdelhafid BESSAID professeur à l'université Abou-bekr Belkaïd de Tlemcen et président du juré. Les examinateurs Monsieur Bruno SALGUES professeur à l'université de Montpellier et Monsieur George NASSAR professeur à l'université de Valencienne.

Merci également à toutes les personnes dont l'amitié m'a apporté des moments de réconfort et d'encouragement nécessaires au bon déroulement de ce projet, notamment, le groupe de recherche en Télémédecine, pour sa disponibilité, son aide et son soutien à tout moment.

Résumé

La pratique télémédicale consiste à prélever sur le patient des informations multidimensionnelles et multi médias représentatives de son état physiopathologique, de les faire parvenir à un terminal informatique local dans un premier temps puis à un terminal informatique distant dans un deuxième temps.

Nous nous proposons dans le cadre de ce travail de mettre en œuvre :

1) La réalisation d'une plateforme télémédicale non invasive et non intrusive dédiée à l'acquisition simultanée et en temps réel de jusqu'à 16 signaux physiologiques unidimensionnels.

2) La réalisation d'une interface hardware construite autour des C.A.N. séries AD0830 d'Analog Device assurant la numérisation des signaux et du microcontrôleur 18F2550 de Microchip assurant leurs transferts du patient vers la poste local sous protocole USB avec les spécifications HID.

3) La réalisation d'une interface graphique sous environnement Visual Basic mettant à profit le composant Winsock compatible avec le système d'exploitation Windows et l'architecture client-serveur permettant la connexion des différents terminaux informatiques pour le transfert des données à travers les réseaux télé médicaux sous protocole TCP-IP.

Mots clés— Télémédecine, USB, HID, mcHID.dll, TCP-IP, Microcontrôleur, Firmware, VB6, Socket, Photopléthysmographie, Electrocardiographie, Spectrophotométries, Télé surveillance.

Abstract

The telemedical practice consists in taking on patient of multidimensional information and multi media representative of its physiopathological state, to do them to arrive to a local computer terminal initially then with a remote computer terminal in the second time.

We propose within the framework of this article to put in work:

1) The realization of a telemedical platform noninvasive and nonintrusive dedicated to simultaneous acquisition and in real-time of to 16 one-dimensional physiological signals.

2) The realization of an interface hardware built around the C.A.N series AD0830 of Analog Device ensuring the digitalization of the signals and the microcontroller 18F2550 of Microchip ensuring their transfers of the patient towards the local post office under protocol USB with specifications HID.

3) The realization of an graphical interface under environment Visual BASIC making profitable the component Winsock compatible with the operating system Windows and architecture customer-server allowing the connection of the various computer terminals for the transfer of the data through the telemedical networks under protocol TCP-IP.

Key words Télémédecine, USB, HID, mcHID.dll, TCP-IP, Microcontroller, firmware, VB6, socket, Photopléthysmographie, Electrocardiographie, Spectrophotométries.

ملخص

في هذه الأطروحة نقدم تطوير واجهة الإنسان ـ آلة للمعلومات الطبية والإتصالات ـ عن طريق USB- HID من جهاز المراقبة إلى جهاز الكمبيوتر وكذلك الإتصالات عن بعد بواسطة TCP IP . استعملت لغة البرمجة على الحاسوب لغة delphi

لمعالجة المعطيات على الحاسوب مثل التمثيل ، التخزين ، انقلاب ، الإزاحة وحساب البيانات الطبية Visual Basic التداخل البيني و كذلك لغة ASSEMBLEUR .

كلمات مفتاحية :

PPG– PTG الطب عن بعد USB HID TCP HP vr fc .

SOMMAIRE

Sommaire

I. Table des matières

LISTE DES FIGURES

Liste des figures

LISTE DES TABLEAUX

Liste des tableaux

Glossaire

Glossaire

A
ADL :Activities of daily living
AFSSAPS: **A**gence **f**rançaise de **s**écurité **sa**nitaire des **p**roduits de **s**anté
AHRI :Aware Home Research Initiative.
AID HOUSE: Assisted Interactive Dwelling House.
AIM: Advanced Informatics in Medicine.
AMT: **A**uto **M**esure **T**ensionnelle au domicile
ANTADIRE : Association Nationale de Traitement à Domicile des Insuffisances Respiratoires.
AOP : Amplificateur Opérationnel.
APA :Allocation Personnalisée d'Autonomie.
AP-HP : Assistance Publique - Hôpitaux de Paris
API: Application Programming Interface.
ARES : Advanced Routing and Edition Software
ARP :AdressResolution Protocol
ASH: adaptable smart home.
ASK :Amplitude Shift Keying
ATM : Asynchronous Transfer mode
B
BITC :Biomedical interactive Technology Center.
C
CEPT: Conférence des Administrations Européennes des Postes et Télécommunications
CESU: Chèque Emploi Service Universel
CFLHTA : Comité Français de Lutte contre l'Hypertension Artérielle
CHS: Health care System
CI: Capacité inspiratoire
CIDR : Classless Inter Domaine Routing
CLNP/CLNS :Connectionless Network Protocol/Connectionless Network Services.
CODEC: Codeur/Decodeur
CPT: Capacité pulmonaire totale
CR :Filtre passe haut
CRF : capacité résiduel fonctionnelle
CSMA/CD : Carrier SenseMiltiple Access/ Collision Detection
CV: Capacité vitale.
D
DCE**:** Data Communication Equipement
DEM: Débit expiratoire maximal

VEMS: Débit expiratoire maximal seconde
DF : Don't Fragment
DLL : Direct Link Library, bibliothèque de liaison dynamique
DNS :Domaine name system.
DoD :Departement of Defense
DPSK: Differential Phase Shift Keying
DSP :Densité spectrale de puissance
DTE: Data Terminal Equipement
DVM: débit ventilatoire maximal

E

ECG: ElectroCardioGramme.
EDF: Électricité de France
EHC: Electronic House call
EHPAD : Etablissements d'Hébergement de Personne Agées et Dépendantes.
EIB :Européen Installation Bus, aujourd'huitKonnex
EMD : Electronique medical record
ETCD: Équipement Terminal de Circuit de Données
ETTD: Équipement Terminal de Traitement de Données
ETSI: EuropeanTelecommunications Standards Institute

F

FDDI :Fiber Data Distributed Interface
FISSA : Force d'Intervention Sanitaire SatelliatireAuto-portée
FO2 :Fraction d'oxygène
FSK: Frequency Shift Keying
FTP: File Transfer Protocol.

G

GPRS**:** General Packet Radio Service
GPS: Global Positioning System.
GSM: Global System for Mobile Communication

H

HAD: Hospitalisation à Domicile
HiPPI: High Performance Parallel Interface
HIS: Habitat Intelligent pour la Santé.
HIS2 :HealthIntegrated Smart Home Information System
HomeRF**:** Home Radio Frequency
HTTP: Hypertext Transfer Protocol.

I

ICMP: Internet Control Message Protocol.
IEEE: Institute of Electrical and ElectronicEngineers.
IGMP : Internet Groupe Management Protocol
IP: Internet Protocol.
irDA**:I**nfrared data association.
ISO: International StandardizationOrganization

IUG: Interface Utilisateur Graphique

L

LAN: Local Area Network.
LON : Local Operational Network
LS : Liaison spécialisée

M

MAD: Maintien à Domicile
MAC: Media access control.
MAN : Metropolitan Area Network
MAQ: Modulation d'amplitude de deux porteuses en quadrature .
MARC: Centre de Recherche en Automatique Médicale
MDA: Modulation par Déplacement d'Amplitude
MDF :Modulation par Déplacement de Fréquence
MDP: Modulation par Déplacement de Phase
MDPD: Modulation par Déplacement de Phase Différentiel
MF :More Fragment
MIDAS: Miniature Intelligent DomiciliaryAlarm System.
MIP: **M**edical**I**nformatics**P**rogramme
MIT: Massachusetts Institute of Technology
MSS : Maximum Segment Size
MTU :Maximum Transfer Unit

N

NAT: Network Address Translator
NBMA : Non Broadcast Multiple Access
NCP :Network Control Program
NFS: Network File System.
NTIC: Nouvelles Technologies de l'Information et de la Communication.
RNIS: Réseau Numérique à Intégration de Service

O

OFDM: Orthogonal Frequency Division Multiplexing
OMS: Organisation Mondiale de la Santé.
ONU Optical Network Unit
ORL: Oto-rhino-laryngologie.
OSI: Open SystemsInterconnection.
OSPF : Open shortestPath First

P

PDA: Personnel Data Assistant.
PDU :Protocol Data Unit
PPG: Photopléthysmographie.
PPP: Point to Point Protocol.
PSK : Phase Shift Keying
PSTN : Public SwitchedTelephone Network
PTG: Phototachographie.

Q
QAM : Quadrature Amplitude modulation
R
RARP :Reverse AddressResolution Protocol
RC: Filtre passe bas
RCA: Rythmes Circadiens d'Activités
RIP : Routing Information Protocol
RFC: Request For Comments.
RTC: Réseau Téléphonique Commuté
RTCP: Real Time Control Protocal
RTP: Real Time Protocal
S
SAMU: Service d'Aide Médicale Urgente
SAP :Service Access Point
SEQ :SequenceNumber.
SID: Système d'Information Domotique-santé Intégré à Domicile
SLIP: Serial Line Interface Protocol.
SMS: Short Message Service.
SMTP: Simple Mail Transfer Protocol.
SNA: System Network Architecture
SpO2 : Saturation en oxygène.
T
TCP: Transport Control Protocol
TCP/IP: Transport Control Protocol/ Internet Protocol
TDM: Tomodensitomètre.
TELNET: Terminal Emulation
TFTP: Trivial FTP
TIC: Technologies de l'Information et de la Communication.
TLI: Transport Layer Interface.
TMBS: Telemedicine Base Station
ToIP: Telephony over TCP/IP
TOS: Type of Services.
TRMC: Taux de Rejection en Mode Commun.
TTL: Time to live.
U
UART: UniversalAsynchronousReceiverTransmitter.
UCLA : Université Californienne de Los Angeles.
UDP: User Datagram Protocol.
UMTS: Universal Mobile Telecom System.
V
VAD :Vivre à Domicile
VB: Visual Basic.
W

WAN: Wide Area Network.
WiMAX: WorldwideInteroperability for Microwave Access
WLAN**:** Wireless Local Area Network
Wi-Fi: Wireless Fidelity.
Winsock: Windows Sockets.
WOSA: Windows Open System Architecture
WSA: Windows Sockets API

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Les signaux physiologiques sont des grandeurs physiques prélevés sur le corps humain au moyen de capteurs appropriés. Ils sont détendeurs d'informations relatives à l'état physiopathologique du patient. Leur traitement revêt un caractère informationnel primordial, permettant d'éclairer le médecin dans son diagnostic et de le guider dans sa thérapeutique.

Le présent mémoire est réparti en six chapitres dans lesquels le premier fait une description des signaux physiologiques suivants : l'Electrocardiogramme (ECG) représentatif de l'activité électrique du myocarde, le Pneumotachogramme (PTG) représentatif de l'activité ventillatoire de l'appareil respiratoire et le Photopléthysmogramme (PPG) représentatif de l'efficacité de l'échangeur pulmonaire au niveau des territoires alvéolo-capillaires.

Le second chapitre traite la notion et le protocole de la transmission USB (universal Serial Bus) dans sa norme large. Pour notre travail, le coté HID a été privilégié. Ici nous faisons un rappel sur le concept communication via un port USB en détaillant les notions de l'attachement, les transactions, les transferts et l'énumération en passant par expliquer les descripteurs.

Le troisième chapitre a trait au principal composant utilisé à savoir le microcontrôleur de chez Microchip et en insistant sur la famille 18F.

Dans le chapitre quatre on trouvera les descriptions concernant les capteurs pour signaux physiologiques étant donné que le sujet traité est en rapport direct avec tout ce qui concerne capteurs, conditionnement, filtrage, acquisition et traitement

Le chapitre cinq est de loin le plus important étant donné que c'est la partie qui explique notre travail réalisé compte tenu du fait que notre contribution est axée surtout sur le coté réalisation pratique et tests.Dans cette partie nous présentons le hardware réalisé, sa description, ses éléments et son fonctionnement.
La carte électronique est principalement axée sur le microcontrôleur, il est donc

normal aussi de décrire le soft embarqué. Le programme est écrit entièrement en assembleur ce qui permet de contrôler au mieux la gestion des ressources.
La carte étant relié au PC, il était nécessaire d'implémenter un programme de gestion de la dite carte. Le programme écrit en Delphi permet de contrôler le flux des transactions entre la carte et le microcontrôleur d'une part et d'en faire l'exploitation à savoir le choix des signaux physiologiques recueillis, le tracé des signaux respectifs ainsi que la gestion de sauvegarde et 'archivage des données.

Le chapitre six est en fait la continuité du précèdent. Alors que le programme hôte sur PC décrit précédemment concerne le traitement en temps réel a fur et a mesure de l'acquisition des signaux physiologiques , il nous est paru indispensable d'y ajouter un autre programme (lui aussi en Delphi) qui permet de faire un traitement approfondi des résultats obtenus , ce traitement sera en fait accompli a partit des données archivées (calcul, statistiques ,comparaisons,…)

Enfin dans le septième et dernier chapitre ,il est question de la communication distante sous le protocole TCP/IP. Ainsi les signaux et résultats obtenus peuvent être envoyé a distance et destinés aux praticiens.

Chapitre I : SIGNAUX PHYSIOLOGIQUES

I Electrocardiographie

Compte tenu de son caractère anodin, non invasif et de la richesse des informations présentes dans le signal électrocardiographique, l'électrocardiographie représente une méthode de choix pour l'exploration du myocarde. Ce dernier fait partie du système cardiovasculaire. Sa principale fonction est d'assurer aux organes et aux tissus un flux sanguin adéquat, continu et sous pression suffisante afin de satisfaire à la fois leurs besoins énergétiques et leurs renouvellement cellulaire[21].

I.1 Physiologie du cœur

Le cœur est un muscle creux (figure I.1) dont la fonction est d'assurer la petite circulation (grâce au ventricule droit) et la grande circulation (grâce au ventricule gauche). Il doit remplir cette fonction sans défaillir une seul minute, de la naissance à la mort, jour et nuit, au rythme de 60 à 80 contractions par minutes. A chaque contraction, le cœur éjecte environ 5 à 6 litres de sang à la minute[22].

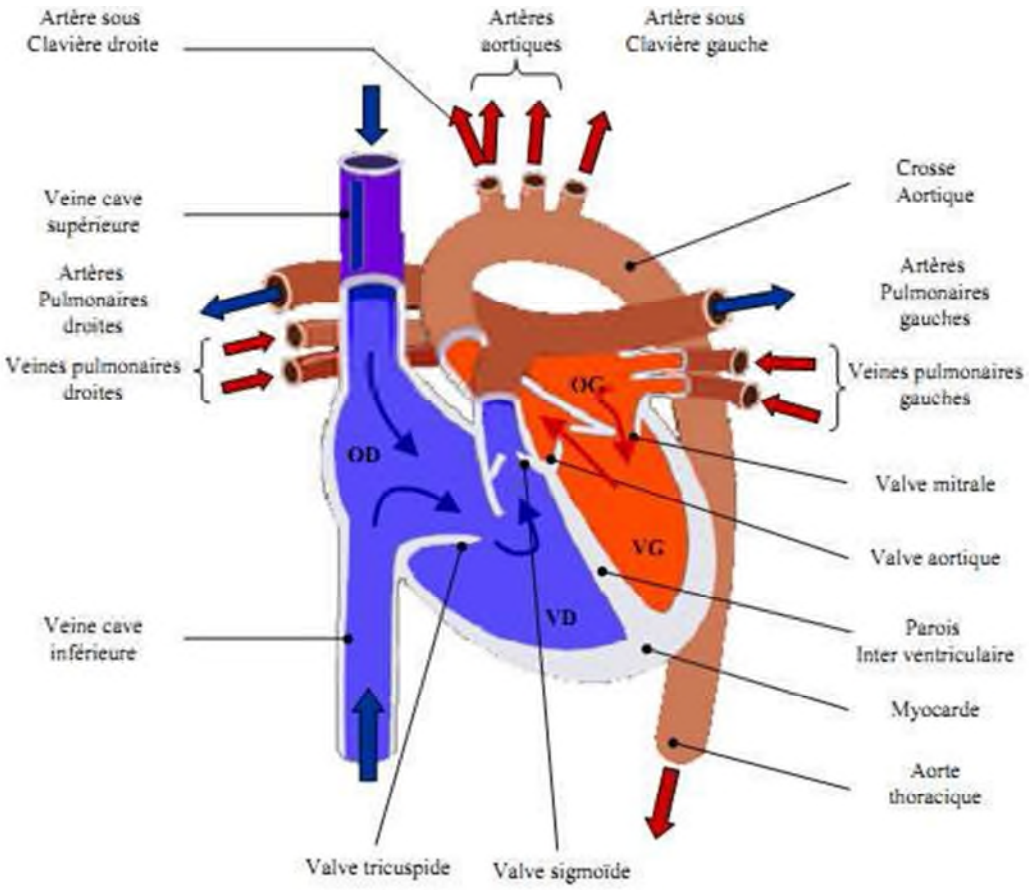


Figure I. 1: Anatomie du coeur

La physiologie du cœur est complexe. Son fonctionnement fait appel à des notions à la fois mécaniques (par son effet d'irrigation de l'organisme), électriques (par le fait qu'il soit activé et contrôlé par les influx nerveux) et chimiques (vu la mise en jeu de substances chimiques).
De plus, son étude doit tenir compte de son milieu intracellulaire et extracellulaire qui sont totalement indépendants. Chacun de ces milieux est électriquement neutre (la quantité des charges électronégatives est égale à celle des charges électropositives), mais leur composition est différente : il ya par exemple du calcium dans le milieu extracellulaire, et du potassium dans le milieu intracellulaire. Cette répartition de charges crée une différence de potentiel électrique de part et d'autre de la membrane qui sépare les deux milieux. Au niveau de la cellule myocardique, la différence de potentiel entre l'intérieur et l'extérieur, que l'on appelle potentiel de repos ou potentiel diastolique de l'ordre de - 90 mV (figure I.2). Quant au potentiel d'action, il est dû à une modification de la perméabilité de la membrane cellulaire aux ions de calcium, de potassium et de sodium. Il indique la contraction musculaire qui survient lorsqu'un stimulus suffisamment puissant est appliqué. Cela se traduit électriquement par une inversion brutale du potentiel de repos : l'extérieur de la cellule devient électronégatif par rapport à l'intérieur. Le retour au potentiel de repos se fait lentement.

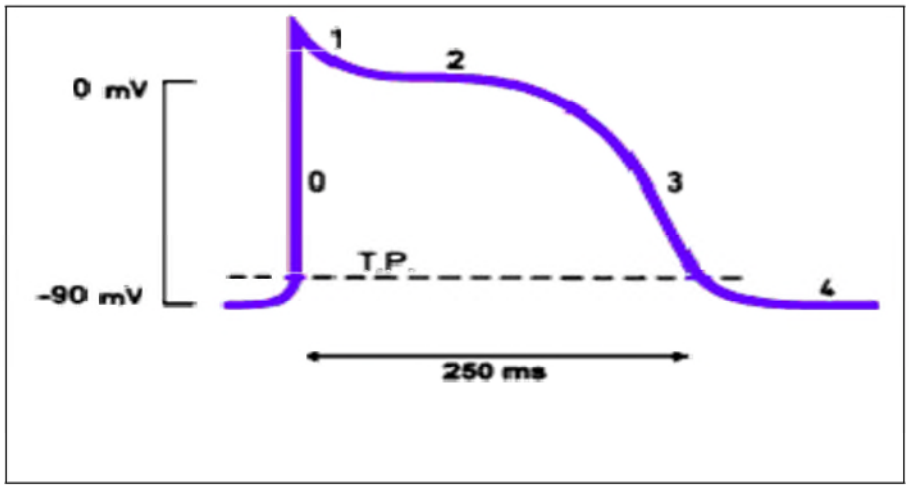


Figure I. 2:Dépolarisation et repolarisation d'une cellule

Toutes les cellules du muscle cardiaque agissent de la même façon. Cependant, les cellules appartenant au nœud sinusal de Keith et flack (figure I.3) se dépolarisent plus vite que les autres et vont imposer leur rythme de fonctionnement aux différentes régions myocardiques. La décharge du nœud sinusal de keith et flack est contrôlée par le système nerveux.

I.2 Propriétés du muscle myocardique

I.2.1 Excitabilité et la contractilité du myocarde

Au repos, la cellule myocardique est excitable et répond au stimulus par une dépolarisation (inversion des charges positives et négatives).

On appelle « seuil », l'intensité minimale du stimulus pour l'obtention de la dépolarisation. Pendant la durée du potentiel d'action, la cellule myocardique du ventricule n'est plus excitable : c'est la phase réfractaire absolue.

I.2.2 Automatisme

Les cellules du nœud de Keith et Flack donnent naissance à un stimulus, au rythme de 80 dépolarisations par minute. Il se propage dans tout le cœur par le faisceau de His et le réseau de Purkinje (figure I.3). Ce centre automatique prend la commande du cœur, et inhibe les autres centres de cet organe.

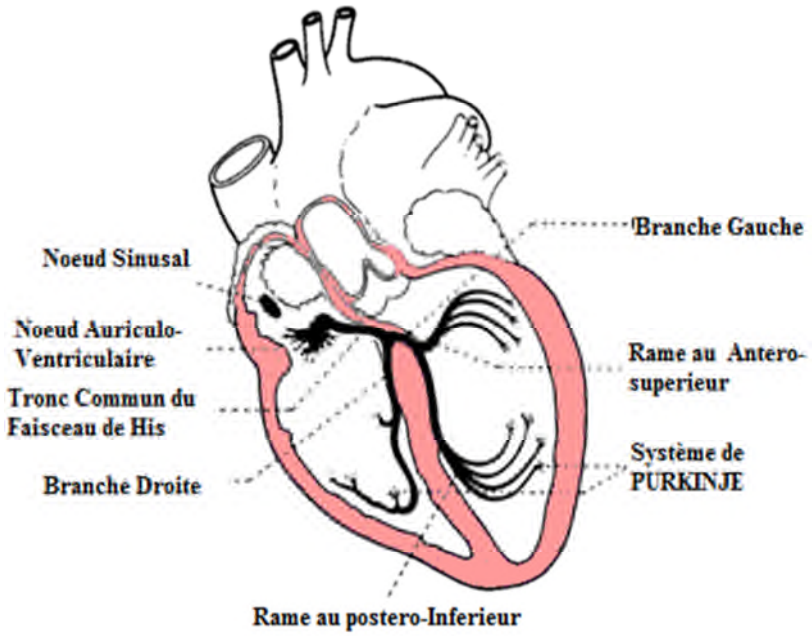


Figure I. 3: Innervation cardiaque

I.2.3 Cycle cardiaque

Le cycle cardiaque est dénommé révolution cardiaque. C'est l'ensemble des phénomènes qui se produisent de façon cyclique au niveau du cœur. Pendant ce cycle qui dure à peine une seconde, il se crée successivement des différences de pression dans les cavités cardiaques, provoquant l'ouverture ou la fermeture des différentes valves, et permettent l'éjection du sang ou le remplissage des cavités. Il se décompose en trois phases (figure I.4)[23]:

I.2.3.1 La diastole

C'est la phase de repos, pendant laquelle le cœur se remplit de sang en provenance de la veine cave et de la veine pulmonaire.

I.2.3.2 La systole auriculaire

C'est le début de la phase de contraction, au cours de laquelle les oreillettes chassent le sang dans les ventricules.

I.2.3.3 La systole ventriculaire

C'est la phase de contraction des ventricules, qui suit immédiatement la contraction des oreillettes.

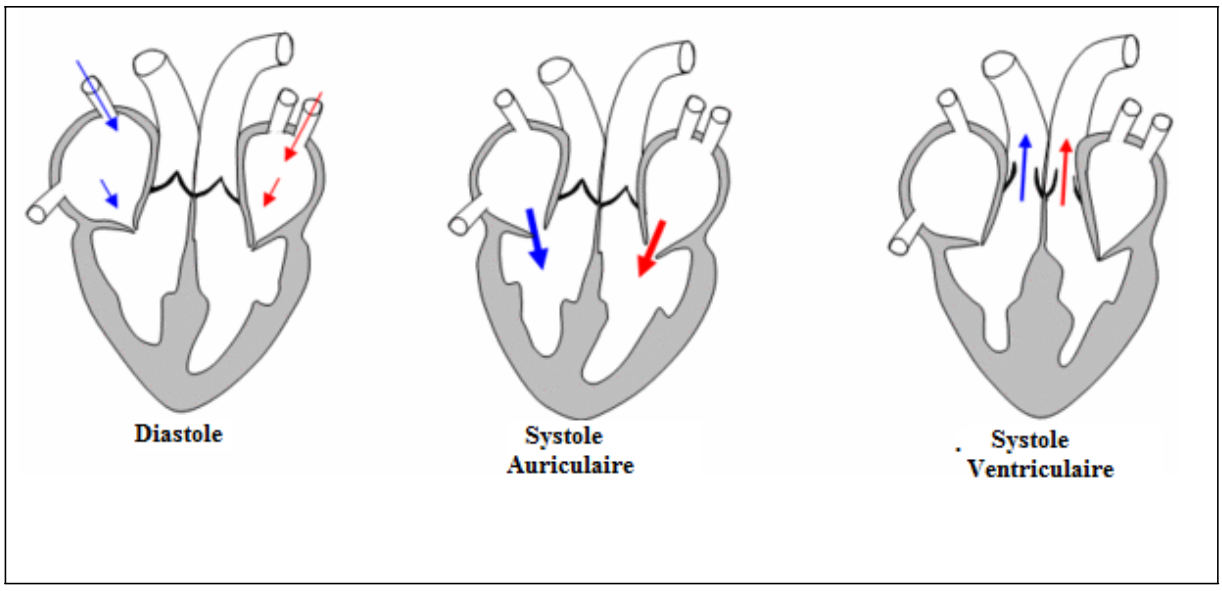


Figure I. 4: Circulation sanguine dans le cœur lors d'une révolution cardiaque.

I.3 Electrocardiogramme

L'électrocardiogramme est un signal physiologique représentatif de l'activité électrique du muscle cardiaque. Cette activité se propage dans le corps humain, jusqu'à la peau. Cela permet de faire des enregistrements de signaux électro-cardiographiques, au moyen d'électrodes positionnées sur plusieurs endroits de la surface du corps, en particulier au niveau du thorax.

I.4 Dérivations électrocardiographiques

La dérivation du courant d'action du cœur se fait toujours entre deux points reliés aux bornes de l'électrocardiographe. Selon les différentes dérivations, l'électrocardiogramme se présente différemment. C'est en confrontant ces différentes dérivations que l'on arrive à avoir une représentation significative. Il existe classiquement douze dérivations, donc douze points d'observation différents de l'activité électrique du cœur (figure I.5) [23].

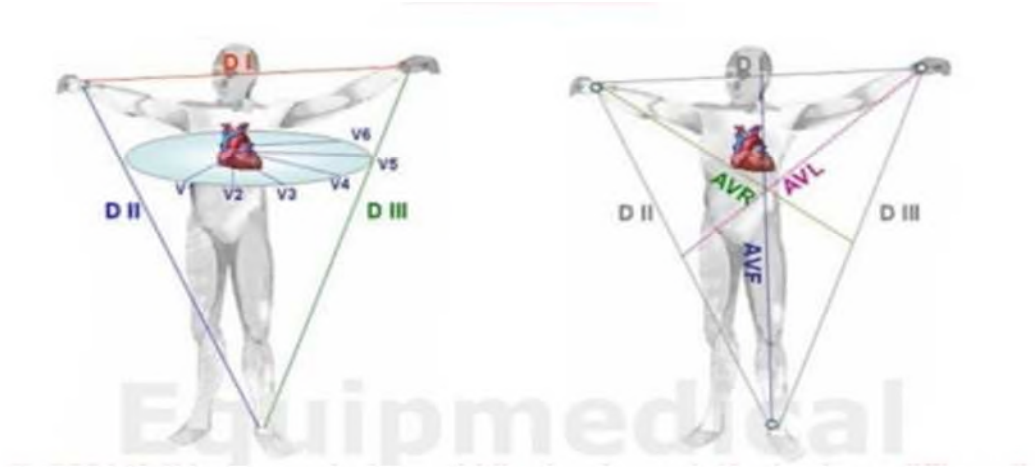


Figure I. 5: - Les 12 dérivations de l'ECG

I.4.1 Dérivations bipolaires: DI, DII, DIII

Les dérivations bipolaires : DI, DII, DIII sont représentées sur la figure I.6 :

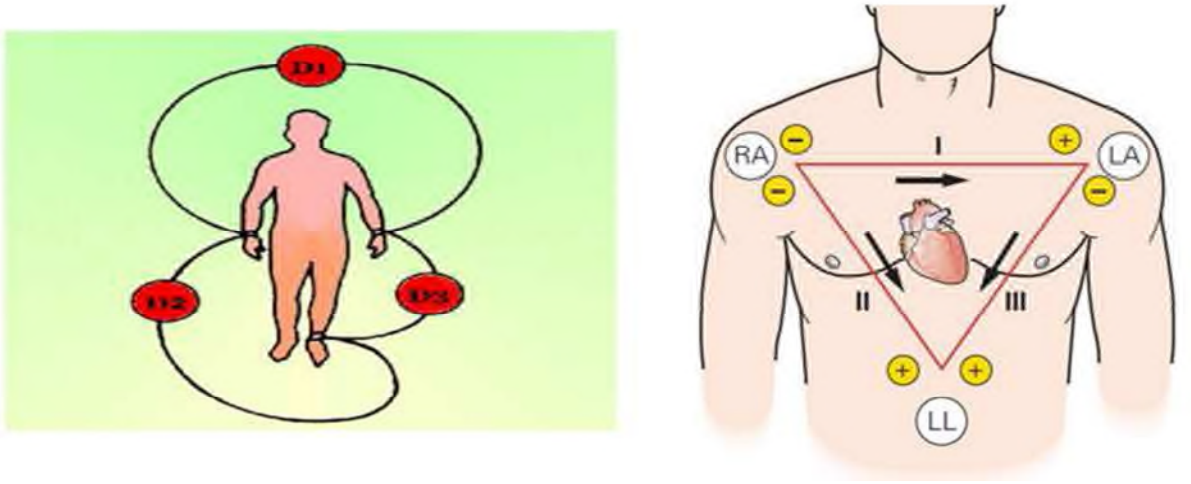


Figure I. 6: Dérivations bipolaires DI, DII, DIII.

DI reliant : poignet droit - poignet gauche.

DII reliant : poignet droit - jambe gauche.

DIII reliant : poignet gauche - jambe gauche.

Ces dérivations sont bipolaires puisqu'elles enregistrent une différence de potentiel entre deux électrodes par rapport à une électrode de référence.

DI explore la paroi latérale gauche du ventricule gauche.

DII suit la cloison inter-ventriculaire et l'apex.

DIII explore la région ventriculaire droite et diaphragmatique du cœur.

I.4.2 Dérivations unipolaires: aVR, aVL,aVF

Les dérivations unipolaires : aVR, aVL,aVF sont représentées sur la figure I.7 :

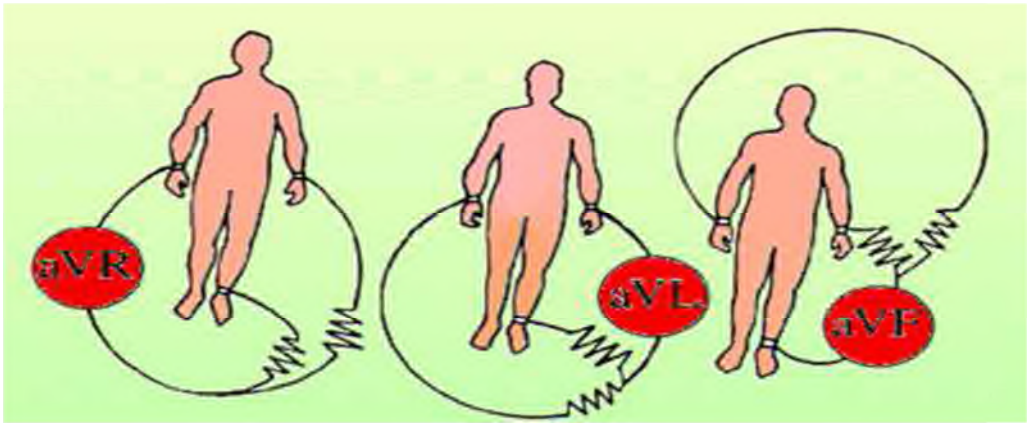


Figure I. 7: Dérivations unipolaires aVL, aVR, aVF.

Comme le montre la figure I.7, en plus de l'électrode de référence, les deux autres électrodes sont utilisées comme suit: la première explore les variations de potentiel d'un des membres. La deuxième est reliée soit à une borne de potentiel zéro (VR, VL, VF), soit aux deux autres membres réunis (aVR, aVL, aVF) tels que :

aVR (poignet droit): explore le potentiel endo-cavitaire.

aVL (poignet gauche) : explore la paroi latérale supérieure du ventricule gauche.

aVF (jambe gauche) : correspond à la partie inféro-diaphragmatique du cœur.

I.4.3 Dérivations unipolaires précordiales : V1, V2, V3, V4, V5, V6

Les dérivations unipolaires précordiales (figure I.8.) n'utilisent qu'une seule électrode exploratrice sur la paroi thoracique antérieure. Elles apportent six nouveaux aspects électriques de l'activité cardiaque. Elles sont définies comme suit :

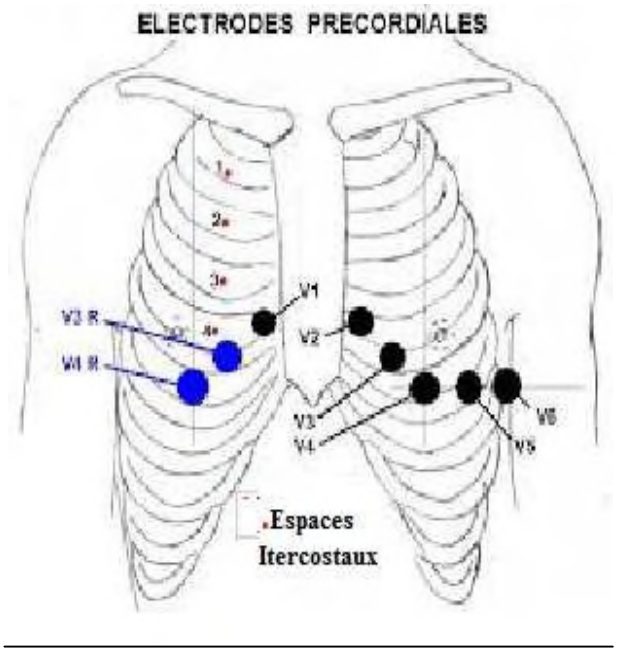


Figure I. 8: Dérivations unipolaires précordiales.

V1 – extrémité interne du 4^{e} E.I.C.D.

V2 – extrémité interne du 4^{e} E.I.C.G.

V3 – au milieu de la ligne V2, V4.

V4 – intersection de la ligne médio-claviculaire et 5^{e} E.I.C.G.

V5 – intersection de la ligne axillaire antérieure gauche et de l'horizontale passant par V4.

V6 – intersection de la ligne médio-axillaire gauche et de l'horizontale passant par V4.

On peut être amené à utiliser des électrodes précordiales postéro-gauches : V7, V8, ainsi que des unipolaires précordiales à droite du sternum :

V3R (right) symétrique à droite de V3 par rapport au sternum.

V4R (right), V4 sternum.

V1, V2, V3 explorent les cavités droites du cœur et la cloison inter-ventriculaire.

V4, V5, V6 explorent la paroi latérale du ventricule gauche et l'apex.

I.5 Tracé électrocardiographique sous la dérivation D1

Chaque partie du cœur (figure I.9) est le siège d'une dépolarisation (perte de potentiel de repos) et d'une re-polarisation ultérieure (récupération du potentiel de repos)[25].

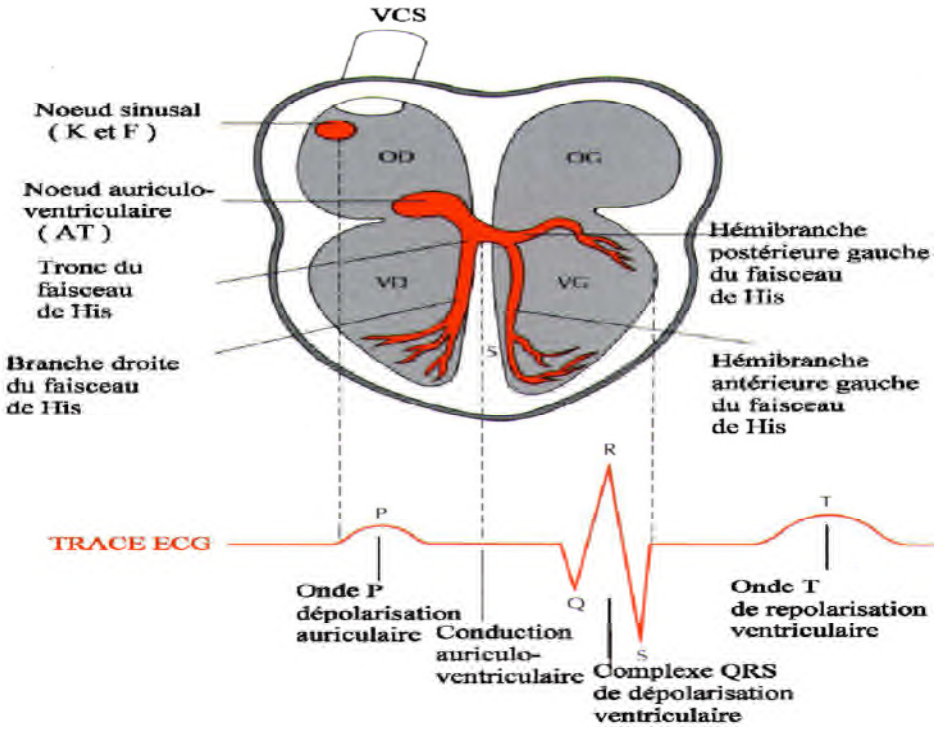


Figure I. 9: Production du signal ECG

La figure I.10 représente un tracé électrocardiographique normal sur la dérivation DI :

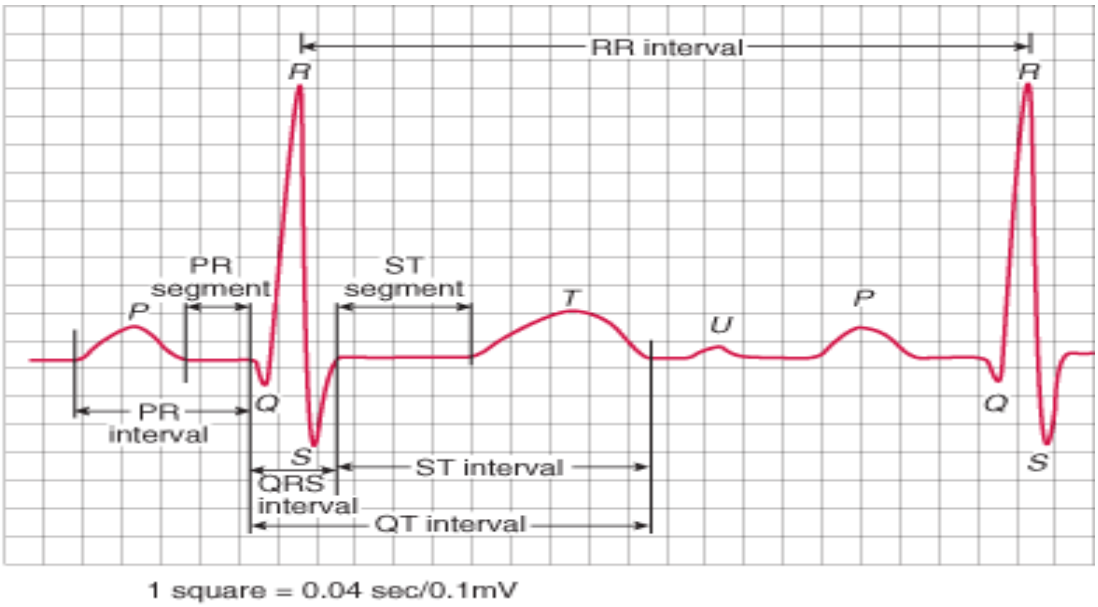


Figure I. 10: Tracé électrocardiographique

L'onde P : E lle traduit la contraction de l'oreillette, c'est une onde auriculaire de dépolarisation [22].Son amplitude ne dépasse pas 2mm en DII et sa durée est comprise entre 1.8 et 0.01 sec. Elle est habituellement positive dans toutes les dérivations sauf en VR. Elle peut être négative en DIII [24].

L'onde QRS : Elle traduit l'activité électrique du ventricule, c'est l'onde ventriculaire rapide de dépolarisation. En regard du ventricule gauche, elle regroupe une petite onde négative Q inférieure à 0.04s, une onde positive R plus ample et, éventuellement une petite onde négative S[22].Le complexe QRS est inférieur à 0.1s habituellement compris entre 0.06 et 0.08s[24].

L'onde T : C'est une onde lente de repolarisation ventriculaire. Elle survient après un léger intervalle de temps [22].Elle est asymétrique avec une partie ascendante lente et un versant descendant plus abrupt. Elle se développe en général dans le même sens que QRS par rapport à la ligne isoélectrique[24].

L'onde U: Elle succède à l'onde T. Habituellement positive, son origine est mal précisée. Il ne faut pas la confondre avec l'onde P. Elle est souvent bien visible en V3, V4[24].

Les tableaux I.1 et I.2 résument les différentes valeurs des paramètres de l'ECG d'un adulte en bonne santé :

	Onde P	**Intervalle PQ**	**Complexe QRS**	**Intervalle ST**	**Intervalle QT**	**Onde T**
Durée (sec)	0.08-0.1	0.12-0.2	0.08	0.20	0.36	0.2
Amplitude (mV)	(Pa) 0.25	Isoélectrique : 0	Q<0, R>0, S<0	Isoélectrique : 0	-	Ta>0

Tableau I. 1

Type d'onde	**Origine**	**Amplitude (mV)**	**Durée (sec)**
L'onde P	Dépolarisation articulaire	<= 0.2mV	Intervalle : P-R 0.12 – 0.22
L'onde R	Repolarisation et dépolarisation ventriculaire	1.60	0.07 – 0.1
L'onde T	Repolarisation des ventricules	0.1 – 0.5	Intervalle : Q-T 0.35 – 0.44
Intervalle S – T	Contraction ventriculaire		Intervalle : S-T 0.015 – 0.5

Tableau I. 2

Tableau I.1 –I.2 : Valeurs habituelles des différents paramètres caractérisant un battement cardiaque **[26].**

II Pneumotachographie

La pneumotachographie est l'une des méthodes les plus utilisées dans l'exploration fonctionnelle respiratoire.

La fonction respiratoire peut être définie comme l'ensemble des processus aboutissant aux échanges gazeux entre un organisme et son environnement. Chez l'homme la respiration se subdivise en quatre composantes : la ventilation, le débit sanguin, la diffusion et le contrôle ventilatoire. Les explorations fonctionnelles respiratoires comprennent classiquement [29] :

- la mesure des volumes pulmonaires et des débits ventilatoires forcés,
- la mesure des gaz du sang,
- l'étude de la mécanique respiratoire, de la fonction des muscles respiratoires et de la commande ventilatoire,
- la mesure de la capacité de transfert de l'oxyde de carbone,
- les épreuves d'exercice physique,
- le cathétérisme cardiaque droit.

II.1 Système respiratoire

Le système respiratoire (voies aériennes et poumons) et le système cardiovasculaire (cœur et vaisseaux sanguins) assurent, d'une part l'acheminement de l'oxygène O_2 jusqu'aux muscles et organes vitaux et d'autre part l'évacuation du dioxyde de carbone CO_2 (déchet musculaire issu de l'activité énergétique)[25]. Nous voyons donc que la fonction cardiaque et la fonction respiratoire sont intimement liées et que par conséquent leur étude conjointe au moyen d'un enregistrement simultané permet au clinicien de mieux appréhender les défaillances du système cardio-respiratoire.

Le système respiratoire est composé des voies aériennes supérieures (nez,bouche, larynx) et des voies aériennes inférieures (endo-trachéennes, broncho pulmonaire)(figure I.11).

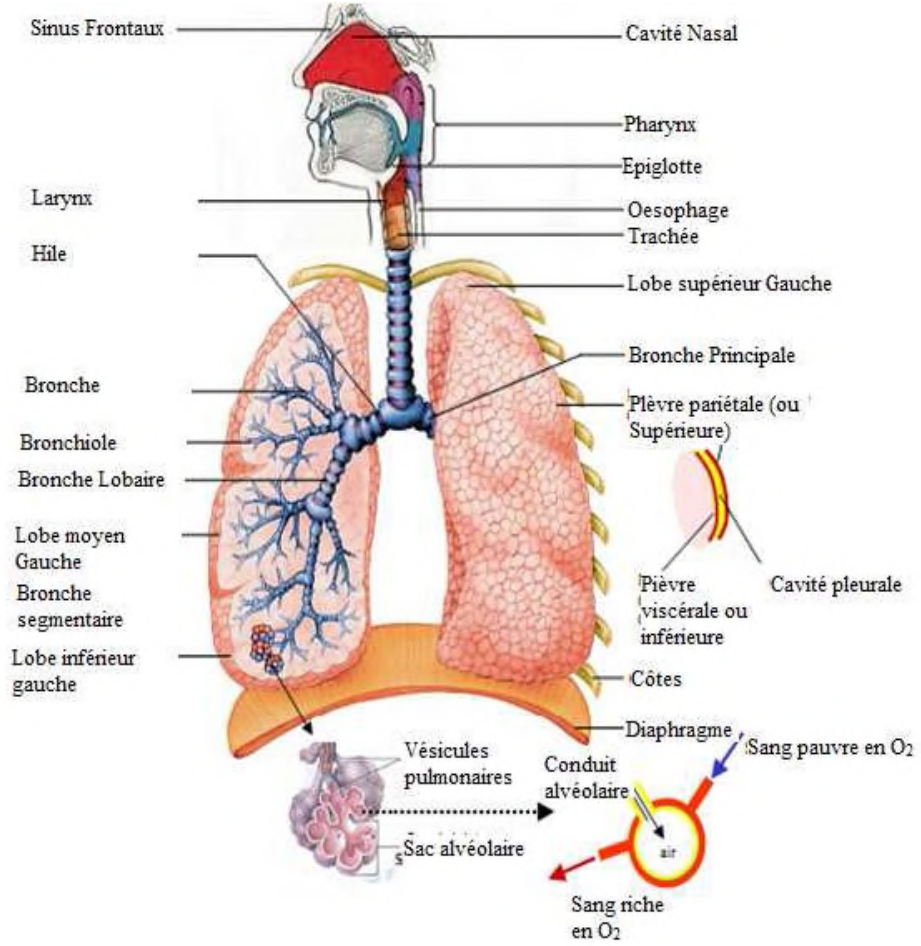


Figure I. 11: Appareil respiratoire

L'intérieur des poumons ressemble à un arbre à l'envers, appelé arbre bronchique : ce dernier est composé des bronches, ramifiés ensuite en bronchioles, qui à leur tour se ramifient en alvéoles.

Lors d'une inspiration, l'air ambiant est aspiré par les voies aériennes supérieures (nez-bouche) pour arriver jusqu'aux poumons. L'air est composé de 21% d'oxygène et de 79% d'azote (environ 0.03% de CO_2). Seul l'oxygène se retrouvera dans le sang, car l'organisme ne consomme pas l'azote [25].

Les alvéoles pulmonaires ressemblent à de petits ballonnets qui se remplissent d'air et présentent une paroi très fine au niveau de laquelle a lieu les échanges gazeux respiratoires. C'est donc une surface d'échange entre l'air et le sang. Le très grand nombre d'alvéoles (environ 200 millions) permet une surface totale d'échange absolument astronomique d'environ 100m². Les alvéoles se gonflent d'air à l'inspiration et se vident lors de l'expiration. La fine paroi des alvéoles est recouverte de très nombreuses et très fines petites veines : les

capillaires (5 à 10 fois plus fins qu'un cheveu), au travers de la paroi desquels se réalise le véritable échange gazeux (figure I.12).

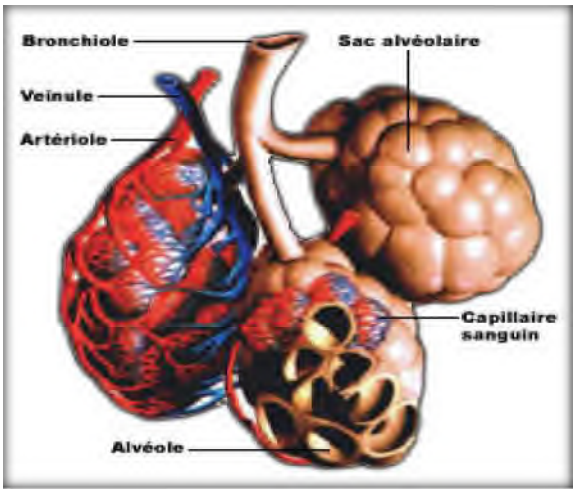


Figure I. 12: Les alvéoles pulmonaires

II.2 Physiologie de l'appareil respiratoire

La respiration correspond à deux mécanismes : l'inspiration qui fournit l'oxygène de l'air à l'organisme et l'expiration qui permet d'éliminer le CO_2.Cet échange gazeux se produit au niveau des poumons, dans les alvéoles pulmonaires grâce au phénomène de diffusion.

Le terme de respiration a aussi un autre sens lorsqu'il correspond aux réactions chimiques oxydatives à l'intérieur des cellules de l'organisme : c'est la respiration cellulaire (elle correspond à la consommation d'oxygène au niveau cellulaire pour dégrader le glucose ou les lipides en produisant du CO_2qui est un déchet de cette dégradation).

Le phénomène ventilatoire se fait grâce aux muscles respiratoires intercostaux et diaphragmatiques (muscle fin à la base des poumons qui sépare la cage thoracique de l'abdomen).

La respiration est un phénomène automatique et spontané. Au repos, le rythme ou fréquence respiratoire d'un adulte moyen est de 16 respirations par minute.

Chaque jour, un adulte inspire environ 8000 litres d'air (à raison de 0,5 litre d'air environ par inspiration).

II.3 Mécanique ventilatoire

Lors de l'inspiration, le diaphragme s'abaisse et les muscles des côtes se contractent, ce qui à pour effet d'augmenter le volume de la cage thoracique et ainsi de diminuer la pression dans les poumons. Cela crée une dépression faisant entrer l'air dans les poumons. C'est une phase active.

Lors de l'expiration, les muscles se relâchent (ceux des côtes et du diaphragme) qui baisseront la cage thoracique. Cela a pour effet d'augmenter la pression dans les poumons. L'air sera donc chassé vers l'extérieur. Cette phase est passive. Cette dernière phase peut devenir active s'il y a contraction des muscles abdominaux (figure I.13).

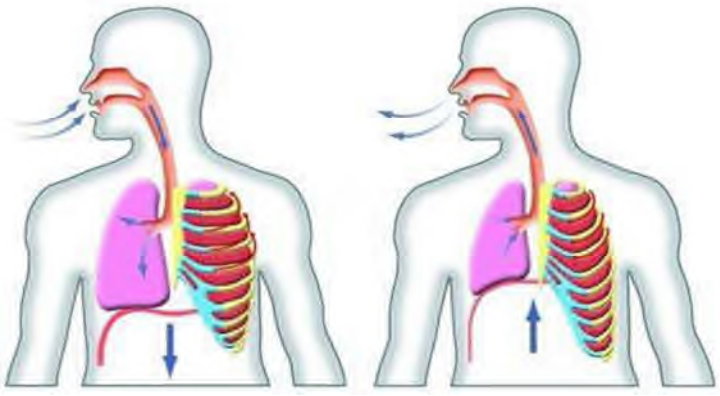


Figure I. 13: Inspiration et expiration

La figure I.14 représente le tracé pneumo-tachographique constitué d'une phase inspiratoire et d'une phase expiratoire lors d'un cycle respiratoire [31].

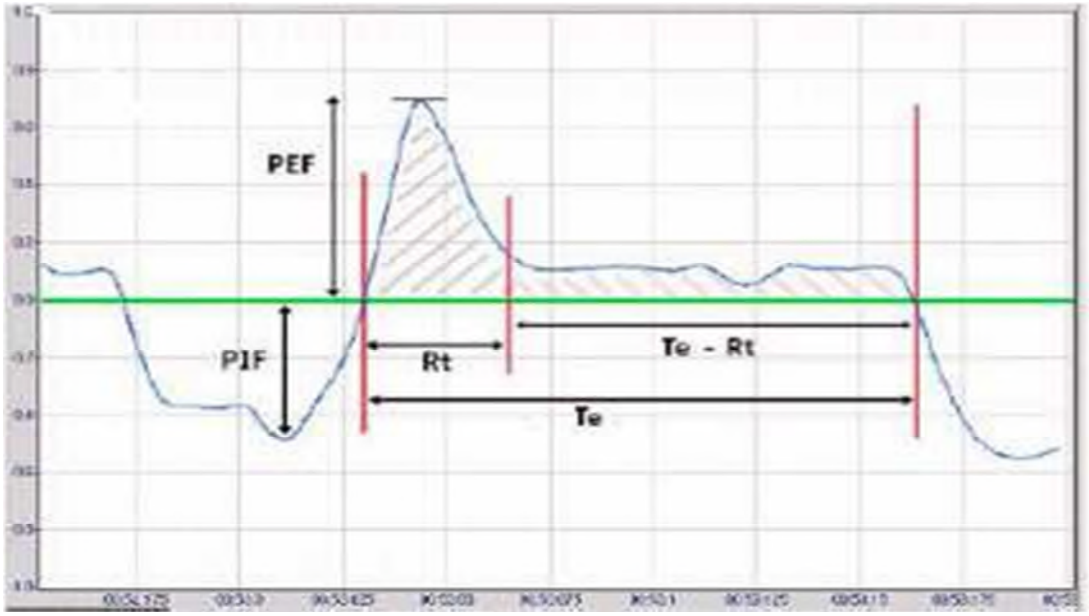


Figure I. 14: Représentation d'un cycle respiratoire.

Te : temps d'une expiration. Rt : Temps pour expirer 65% du volume total.
PIF: Pic maximum d'une inspiration. PEF : Pic maximum d'une expiration [29].

Lorsque l'organisme a des besoins plus importants en O_2, le cœur accélère son rythme pour maintenir un flux circulatoire adapté dans les deux sens.

II.4 Représentation spirographique de la respiration

La spirométrie est un test de mesure de la respiration. C'est le plus fréquent des tests de contrôle de la fonction pulmonaire. Elle consiste en une série d'examens des fonctions respiratoires, selon des paramètres et dans des conditions précises. Le but d'une spirométrie est de contrôler la fonction ventilatoire en mesurant les volumes d'air mobilisés par les mouvements respiratoires et les débits ventilatoire.

Ces tests visent à déterminer de manière relativement simple, à la fois les paramètres de différentes capacités pulmonaires, les volumes pulmonaires et les débits d'air (inspiration, expiration) d'un patient. Cela se fait dans le but de diagnostiquer certaines pathologies respiratoires (asthmes, BPCO entre autres) et de suivre leur évolution.

La spirométrie apporte des informations très précises concernant les maladies respiratoires, et spécialement les maladies dites restrictives et obstructive (broncho-pneumopathie chronique obstructive - BPCO).

Il existe deux principaux types de spirométrie: simple et forcée.

Les résultats sont représentés sous forme graphique et montrent l'évolution du volume d'air et de son débit en fonction du temps.

La figure I.15 montre les différents volumes et capacités pulmonaires explorés par pneumo-tachographie.

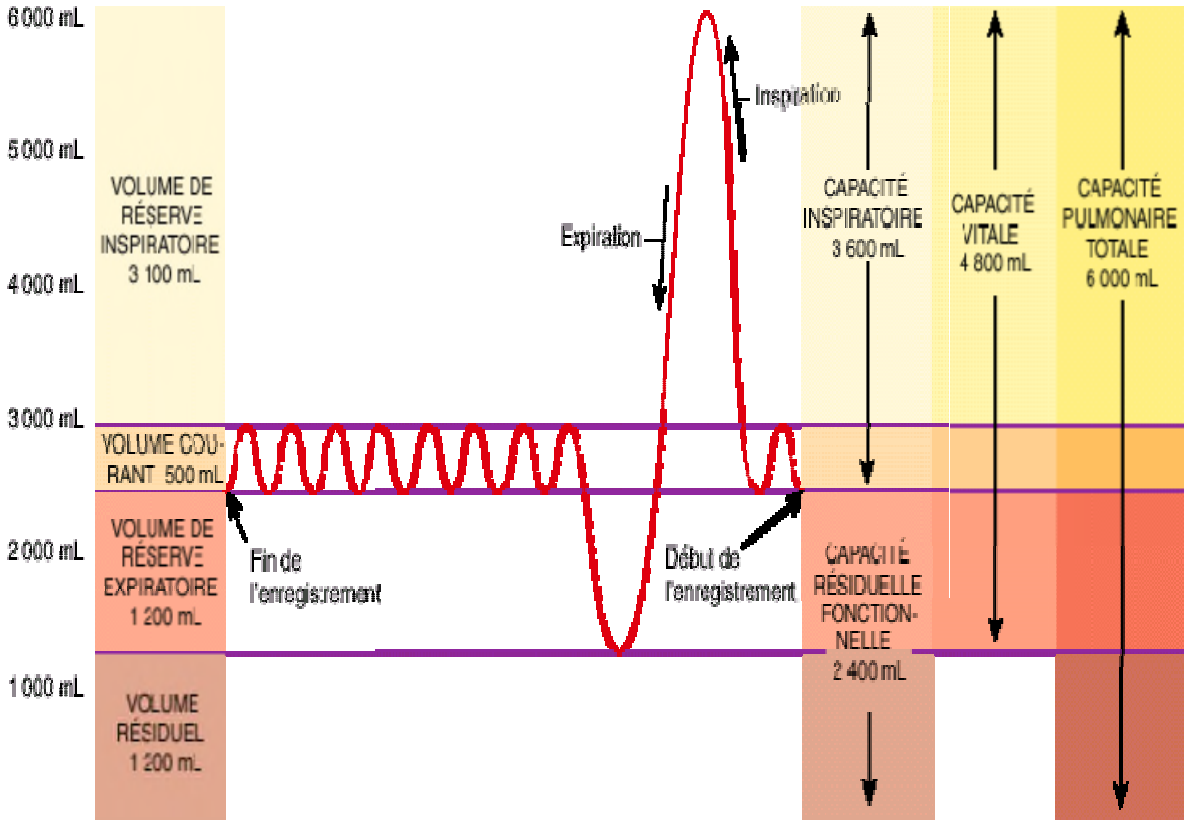


Figure I. 15: Volume et la capacité pulmonaire chez l'homme.

Le tableau I.3 résume les différents volumes misent en jeu par l'appareil respiratoire.

Volume	Abréviation	Définition
Volume courant	VC	Volume mobilisé à chaque cycle respiratoire pendant une respiration normale. Valeur : 0,5 l d'air (500 ml)
Volume de réserve inspiratoire	VRI	Volume maximum pouvant être inspiré en plus du VC à l'occasion d'une inspiration profonde. Valeurs : chez l'homme, 3,1 l + 0,5 l (VC) = **3,6 l** et chez la femme, 2 l + 0,5 l (VC) = **2,5 l**
Volume de reserve expiratoire	VRE	Volume maximum pouvant être rejeté en plus du volume courant à l'occasion d'une expiration profonde
Volume résiduel	VR	Volume d'air se trouvant dans les poumons à la fin d'expiration forcée. Autrement dit qu'il est impossible d'expirer. Il est impossible de mesurer ce volume avec des tests de spiromètre. Pour mesurer le VR, des tests plus sophistiqués, comme la méthode dilution à l'hélium ou la pléthysmographie, sont nécessaires

Tableau I. 3: – Différents volumes pulmonaires

T

Le tableau I.4 résume les différentes capacités pulmonaires misent en jeu par l'appareil respiratoire.

Capacité	Abréviation	Définition
Capacité vitale	CV	VRI + VC + VRE
Capacité inspiratoire	CI	VC + VRI
Capacité résiduelle fonctionnelle	CRF	VRE + VR
Capacité pulmonaire totale	CPT	CV+VR

Tableau I. 4: Capacités pulmonaires

II.5 Trajet de l'O_2 dans le sang

Avant de retourner dans le sang et durant la phase d'inspiration, l'O_2 va traverser deux minces parois: la paroi des alvéoles et la paroi des capillaires pulmonaires. Une fois dans le sang, l'O_2 est alors transporté en se fixant directement sur l'hémoglobine, une protéine contenue dans les globules rouges (hématies) du sang.

Le sang qui s'est chargé en oxygène au niveau des poumons va rejoindre la partie gauche du cœur. Une fois cette partie gauche remplie de sang, le cœur va se contracter et va éjecter ce sang dans une grosse artère (l'aorte), avant de rejoindre les autres vaisseaux sanguins. Ainsi le sang va passer des artères aux vaisseaux pour arriver enfin au niveau des cellules musculaires par le biais des capillaires musculaires (figure I.16).

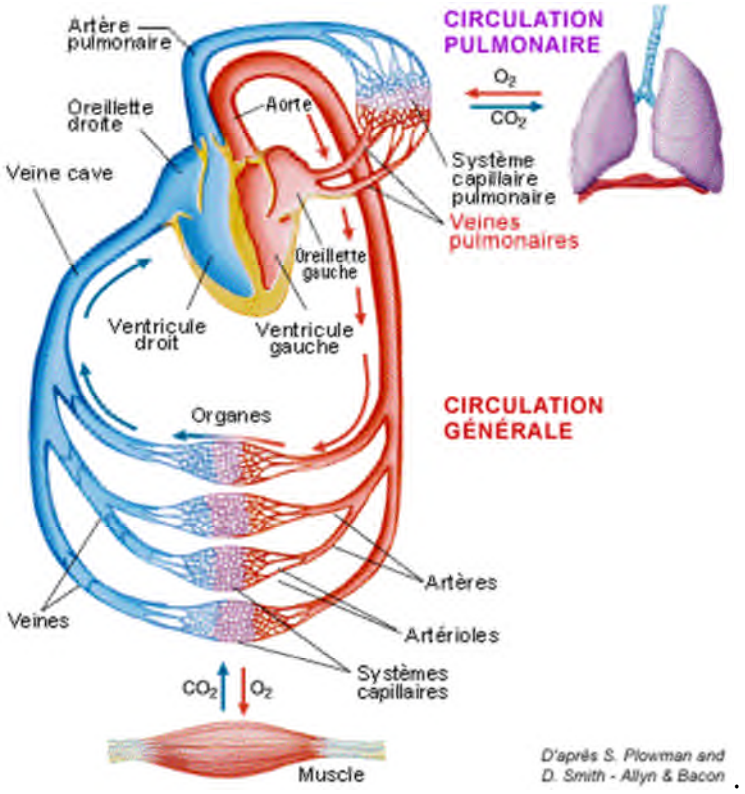


Figure I. 16: - Circulation du sang.

En effet, ces capillaires musculaires traversent les muscles et les approvisionnent en O_2. Comme ces derniers vaisseaux sont très fins, le sang circule plus lentement et l'oxygène va avoir le temps de se détacher de l'hémoglobine pour ensuite pénétrer librement dans les cellules musculaires. Une partie de cet oxygène reste toujours dans le sang, accroché à l'hémoglobine [25].

Une fois à l'intérieur de la fibre musculaire, l'oxygène est transporté par la myoglobine, une sorte d'hémoglobine intramusculaire. Celle-ci va amener l'oxygène dans la mitochondrie qui est une véritable " centrale énergétique", une structure organique à l'intérieur desquelles sont brûlés des nutriments (glucides, lipides, ou plus rarement acide aminé) dans un but énergétique. Les fibres musculaires contiennent des milliers de mitochondries qui assurent la combustion aérobie (en présence d'O_2) des nutriments. Les mitochondries assurent exclusivement les mécanismes énergétiques aérobies. Une fois dans la mitochondrie, l'oxygène va être utilisé pour brûler les nutriments et libérer ainsi de l'énergie nécessaire à la re-synthèse de l'ATP.

Dans les mitochondries il se produit la réaction suivante :

$$\text{Nutriment} + O_2 \longrightarrow CO_2 + H_2O + \text{Energie} \qquad \text{(I.1)}$$

Dans cette réaction, le mot nutriment désigne le carburant utilisé (glucose ou lipide ou plus rarement acide aminé), quant à l'énergie dégagée, elle va servir à la contraction du muscle (qui génère le mouvement).

La myoglobine est capable de stocker une petite quantité d'oxygène, ce qui lui permet lors d'un effort de fournir la mitochondrie sans attendre que le système cardiovasculaire ait atteint sa pleine mesure.

Après avoir approvisionné les muscles en O_2, le sang récupère le CO_2 produit durant la combustion des carburants énergétiques (nutriments) puis, il remonte jusqu'à la partie droite du cœur. Le sang est alors pauvre en O_2 et riche en CO_2. Une fois dans la partie droite du cœur, il va être expulsé vers les poumons où il va pouvoir rejeter le CO_2 dans l'air et se ré approvisionner en O_2. Le sang aura alors fait le tour de tout l'organisme [25].

L'oxygène est présent dans le sang sous deux formes :

-la forme dissoute (PaO_2) (environ : 2% du volume total d'oxygène).

-la forme combinée à l'hémoglobine (HbO_2), sachant que l'O_2 dissout est très faible par rapport à l'O_2 combiné.

III Signaux représentatifs de la saturation d'O_2

L'oxymétrie de pouls ou saturation en oxygène est une méthode de mesure non invasive de la saturation en oxygène de l'hémoglobine au niveau des capillaires sanguins. On parle de saturation pulsée en oxygène SpO_2 [31].

Les hématies ou globules rouges sont composées d'environ 33 % d'hémoglobine.

Chaque molécule d'hémoglobine porte quatre atomes de fer relié à une molécule d'oxygène (figure I.17).

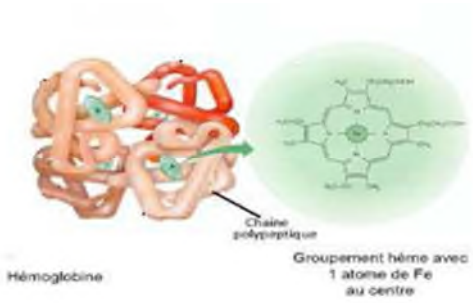


Figure I. 17: Composition d'une molécule de l'hémoglobine

Lorsque l'oxygène se lie au fer, le globule rouge se charge en oxygène et prend une couleur rouge vif. L'hémoglobine oxygénée s'appelle l'oxyhémoglobine. Cette forme de transport représente 98% du contenu total en O_2 dans le sang,

Lorsque l'oxygène est distribué aux tissus, le globule rouge est quasi déchargé en oxygène.

L'hémoglobine désoxygénée s'appelle, la déoxyhémoglobine.

L'oxymétrie de pouls s'est imposée comme un élément indispensable de la surveillance des patients. Aux yeux du personnel soignant, la SpO_2 mesurée par l'oxymétrie est le reflet fidèle de la saturation artérielle en oxygène (SaO_2), garante de la sécurité des patients [32].

Cette mesure de la saturation de l'O2 dans le sang vise à surveiller l'oxygénation du sang et à détecter, de manière fiable et précoce, toute hypoxie. Elle sert également à surveiller la fréquence cardiaque.

L'oxymétrie de pouls se base sur deux principes optiques :

III.1 **Pléthysmographie**

La pléthysmographie utilise la technologie de l'absorbance lumineuse pour reproduire les ondes produites par le sang pulsatile.

III.2 **Spectrophotométrie**

La spectrophotométrie mesure quantitativement l'absorption lumineuse à travers des substances données à des longueurs d'onde variées.

III.3 **Photoplethysmographie**

La photopléthysmographie est une technique d'exploration fonctionnelle vasculaire non invasive permettant de diagnostiquer des affections telles que la tachycardie ventriculaire qui est souvent secondaire à un infarctus du myocarde ou une lésion de fibrose dans une cardiomyopathie. Souvent une tachycardie est causée par une hypoxémie qui est la diminution de la quantité d'oxygène transportée dans le sang. La pression artérielle en O_2 diminue (synonyme généralement d'un mauvais échange entre les alvéoles pulmonaires et les capillaires sanguins).

La photopléthysmographie est une méthode optique qui permet de relever le signal photopléthysmographique PPG (figure I.18) [33]. Ce tracé découle de la modification de l'atténuation de l'énergie lumineuse transmise par les tissus sur lesquels la lumière a été appliquée [34]. Le principe de fonctionnement est détaillé dans le deuxième chapitre.

Figure I. 18: Forme de signal PPG [34]

IV **Conclusion**

L'exploration fonctionnelle du système cardio-respiratoire met en jeu trois signaux physiologiques : l'Electrocardiogramme caractérisant l'activité électrique du myocarde, le Pneumotachogramme caractérisant le système respiratoire et le Photopléthysmogramme caractérisant l'efficacité hémodynamique. Ils ont pour effets de diagnostiquer et de confirmer une affection, d'apprécier sa sévérité et de garantir l'efficacité du traitement.

En procédant au traitement numérique de ses signaux physiologiques, nous mettons à la disposition du médecin un plateau technique qui lui permettra de développer des protocoles d'exploration, capables de répondre à des problématiques spécifiques.

Dans ce premier chapitre, nous avons présenté les différents signaux physiologiques unidimensionnels, objet de notre traitement.

Dans le deuxième chapitre nous aborderons les principes physiques permettant la mise en œuvre de capteur physiologique et l'acquisition du signal électrique.

Chapitre 2 **NORME UNIVERSAL SERIAL BUS[19][20]**

I INTRODUCTION :

Le *Universal Serial Bus* (**USB**, en français *Bus universel en série*, dont le sigle, inusité, est *BUS*) est une norme relative à un bus informatique en transmission série qui sert à connecter des périphériques informatiques à un ordinateur. Le bus USB permet de connecter des périphériques *à chaud* (quand l'ordinateur est en marche) et en bénéficiant du *Plug and Play* (le système reconnaît automatiquement le périphérique). Il peut alimenter les périphériques peu gourmands en énergie (disques SSD en particulier).

Apparu en 1996, ce connecteur s'est généralisé dans les années 2000 pour connecter souris, clavier d'ordinateur, imprimantes, clés USB et autres périphériques bon marché sur les ordinateurs personnels.

L'USB s'est depuis déclinée en version 2 (0,48 Gb/s) et version 3 (5 Gb/s) puis en août 2013 est annoncé le standard 3.1 à 10 Gb/s.

L'USB a été conçu au milieu des années 1990 afin de remplacer les nombreux ports externes d'ordinateurs lents et incompatibles. Différentes versions de la norme ont été développées au fur et à mesure des avancées technologiques.

- En 1996, la première version de la norme, l'**USB 1.0**, est spécifiée par sept partenaires industriels (Compaq, DEC, IBM, Intel, Microsoft, NEC et Northern Telecom).
- En 1998, la version **USB 1.1** apporte des corrections et deux vitesses de communications : 1,5 Mbit/s (*faible vitesse*, ou Low Speed), et 12 Mbit/s (soit 1,5 Mo/s) (*pleine vitesse* ou Full Speed). En août 1998, Apple est le premier constructeur à uniquement proposer les ports USB (en remplacement des ports d'ancienne génération), avec la sortie de l'iMac G3, ce qui a fait décoller le marché des périphériques USB.
- En 2000, la version **USB 2.0** ajoute des communications à 480 Mbit/s (*haute vitesse* ou High Speed) (soit 60 Mo/s).
- En 2005, le **Wireless USB**, une version sans-fil de l'USB, est spécifiée par le *Wireless USB Promoter Group*. Elle promet 480 Mbit/s à une distance de 3 m et 110 Mbit/s à 10 m.
- En 2008, l'**USB 3.0** transmet à 5 Gbit/s (soit 600 Mo/s) (*vitesse supérieure* ou SuperSpeed) et 4,5 Watts. Les nouveaux périphériques disposent de connexions à 6 contacts au lieu de 4, mais la compatibilité ascendante des prises et câbles avec les versions précédentes est assurée. **La compatibilité descendante est impossible**, les câbles USB 3.0 de type B n'étant pas compatibles avec les prises USB 1.1/2.0, mais il existe des adaptateurs.
- Début 2010 introduction de l'USB 3 dans des produits grand public. Les prises femelles correspondantes sont signalées par une couleur bleue. Apparition aussi des prises femelles USB *rouges*, signalant une puissance électrique disponible supérieure, et appropriée au chargement rapide de petits appareils y compris (à condition de le paramétrer dans le BIOS ou l'EFI) lorsque l'ordinateur est éteint.
- Août 2013, l'**USB 3.1** est officialisé et promet des débits doubles de ceux de l'USB 3.0, soit 10 Gbits/s (1,2 Go/s). Le nouveau standard (câbles, interface) sera rétro compatible avec l'USB 3.0 et l'USB 2.0. Toutefois la connectique change, elle sera plus fine et n'imposera pas de sens de branchement. Pour tout de même permettre la connexion vers des connecteurs USB

2.0 et 3.0, le standard devra prendre en compte la possibilité d'avoir des adaptateurs passifs (à l'inverse des adaptateurs Lightning, le connecteur réversible qu'Apple a lancé avec l'iPhone 5 en 2012), pour garder une taille réduite et un coût de fabrication mesuré. Cette nouvelle connectique se nommera *Type C*.

II CARACTERISTIQUES GENERALES:

L'*Universal Serial Bus* est une connexion à haute vitesse qui permet de connecter des périphériques externes à un ordinateur (hôte dans la terminologie USB). Il permet le branchement simultané de 127 périphériques par contrôleur (hôte). Le bus autorise les branchements et débranchements à chaud (« *Hot-Plug* », sans avoir besoin de redémarrer l'ordinateur) et fournit l'alimentation électrique des périphériques sous 5 V, dans la limite de 0,5 A, soit 2,5 W.

D'un point de vue logiciel, le bus possède une topologie arborescente (dite également en étoile) : les feuilles de cet arbre sont les périphériques ; les nœuds internes sont des *hubs* qui permettent de greffer des sous-arborescences dans l'arborescence principale. On trouve dans le commerce ces *hubs* sous forme de petits boîtiers alimentés soit sur le bus, soit sur le secteur, et qui s'utilisent comme des multiprises. Certains périphériques intègrent également un *hub* (moniteurs, claviers…). Cependant, tout bus USB possède au moins un *hub* situé sur le contrôleur : le *hub racine*, qui peut gérer les prises USB de l'ordinateur. Le nombre de *hubs* connectés en cascade est limité : *hub racine* compris, il ne doit pas exister plus de sept couches dans l'arborescence (**figure 2.1**).

À plus bas niveau, il s'agit d'un anneau à jeton (ou *Token Ring*) : chaque nœud dispose successivement du bus. Il n'y a pas de collision de paquets comme en Ethernet, mais le nombre maximal de nœuds est prédéfini. Pour cette raison, l'USB n'est pas adapté aux communications réseau : l'apparition des "modems" ADSL USB était un moyen de diffuser l'ADSL à une époque où la plupart des PC bas de gamme disposaient du port USB mais pas d'Ethernet.

La version 1.x du bus peut communiquer dans deux modes : lent (1,5 Mbit/s) ou rapide (12 Mbit/s, soit 1,5 Mo/s) :

- le *mode lent* (« *Low Speed* ») permet de connecter des périphériques qui ont besoin de transférer peu de données, comme les claviers et souris ;
- le *mode rapide* (« *Full Speed* ») est utilisé pour connecter des imprimantes, scanners, disques durs, graveurs de CD et autres périphériques ayant besoin de plus de rapidité. Néanmoins il est insuffisant pour beaucoup de périphériques de stockage de masse (ce mode permet la vitesse « 10 X » des CD).

USB 2.0 introduit un troisième mode permettant de communiquer à 480 Mbit/s. Ce mode est appelé « *High Speed* ». Il est utilisé par les périphériques rapides : disques durs, graveurs… Mais en 2009, la plupart des périphériques ont une vitesse inférieure à ce que permet l'USB 2.0.

La dernière version, l'USB 3.0, comporte un quatrième mode (« *Super Speed* ») permettant de communiquer à 5 Gbit/s. Ce nouveau mode utilisant un codage des données de type 8b/10b, la vitesse de transfert effective des données est de seulement 4 Gbit/s (500 Mo/s).

Lorsque l'on parle d'un équipement USB, il est nécessaire de préciser la version de la norme (1.1, 2.0 ou 3.0) mais également la vitesse (Low, Full ou High Speed). Une clef USB spécifiée en USB 2.0 n'est pas forcément High Speed si cela n'est pas précisé par un logo « High Speed ».

Le bus USB reste plus lent que des interfaces internes comme PCI ou AGP ou SATA /e-Sata (dans sa version 1.x et 2.0).

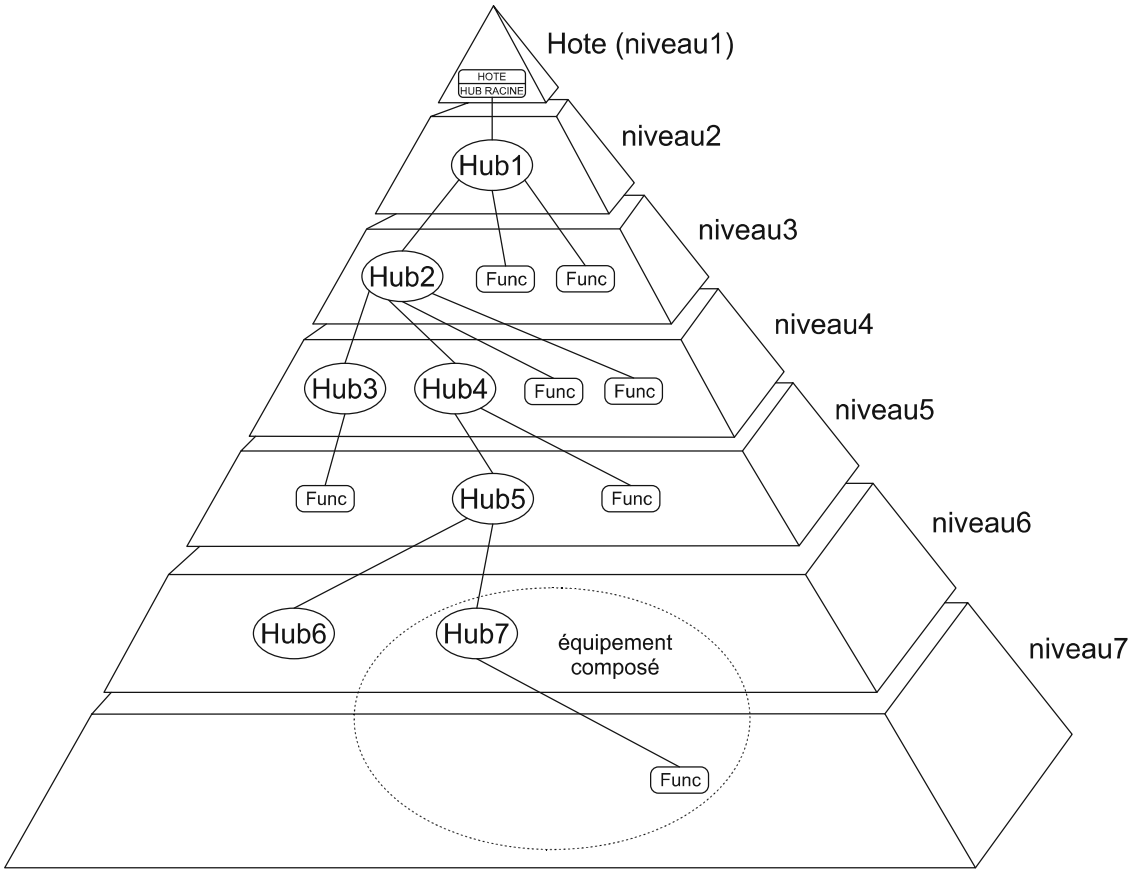


Figure II.1: Topologie du bus USB

L'USB, comme son nom l'indique est un Bus Série. Il utilise quatre fils isolés dont deux sont l'alimentation (+5V et GND). Les deux restants forment une paire torsadée qui véhiculent les signaux de données différentiels. Il utilise un schéma d'encodage NRZI (Pas de retour à Zéro inversé – figure 2.2) pour envoyer des données avec un champ sync de manière à synchroniser les horloges de l'Hôte et du récepteur.

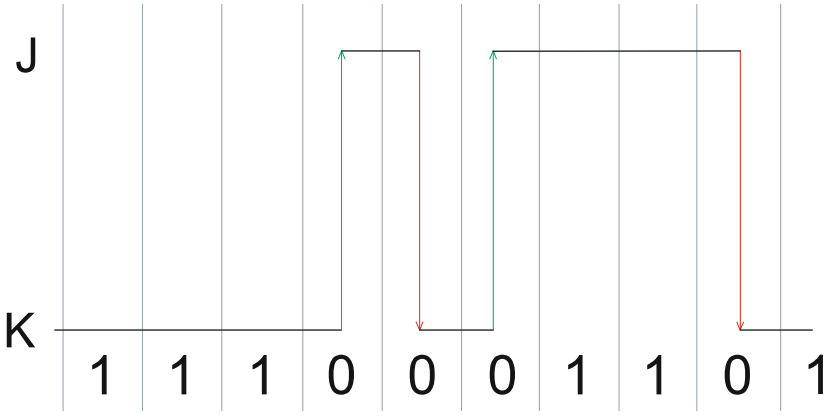


Figure II.2: Codage NRZI

L'USB supporte le système " plug'n play " branchement à chaud avec des drivers qui sontdirectement chargeable et déchargeable. L'utilisateur branche simplement l'appareil sur le Bus. L'Hôte détectera cet ajout, interrogera l'appareil nouvellement inséré et chargera le driver approprié pendant le temps qu'il faut au sablier pour clignoter sur votre écran assurant qu'un driver est installé pour votre

appareil. L'utilisateur final n'a pas besoin de se soucier des terminaisons, des termes tels que IRQs et adresses de ports, ou de la réinitialisation de l'ordinateur. Une fois que l'utilisateur a terminé, on peut simplement retirer le câble, l'Hôte détectera cette absence et déchargera automatiquement le driver.

Le chargement du driver approprié sera réalisé en utilisant une combinaison PID / VID (Interface Produit Machine / Vendeur Machine). On peut se procurer le VID au forum desfournisseurs USB en payant, ce qui est considéré comme un autre point de blocage par USB. On peut trouver le catalogue des tarifs réactualisés sur le site web des fournisseurs USB.

Une autre caractéristique intéressante de l'USB réside dans ces modes de transferts. L'USB soutient des transferts de contrôles, d'interruptions, en Bloc et Isochrone. Lorsque nous examinerons les autres modes de transferts ultérieurement, nous nous rendrons compte que l'Isochrone permet à un appareil de réserver une approximation définie de la bande passante avec un temps d'attente garanti. Ce système se révèle idéal dans les applications Audio et Vidéo ou l'encombrement peut susciter une perte de données ou une chute de trames. Chaque mode de transfert fourni au concepteur des compromis dans les domaines de la détection d'erreur et de la reprise, du temps d'attente garanti et de la bande passante.

III CARACTERISTIQUES MECANIQUES:

Tous les appareils ont une connexion amont vers l'hôte et tous les hôtes ont une connexion aval vers l'appareil. Les connecteurs amont et aval ne sont pas interchangeables mécaniquement, éliminant ainsi les connexions de rebouclage interdite aux Hubs comme pour un port aval connecté à un port aval. Il y a généralement 2 types de connecteurs, appelé type A et type B présenté ci-dessous.

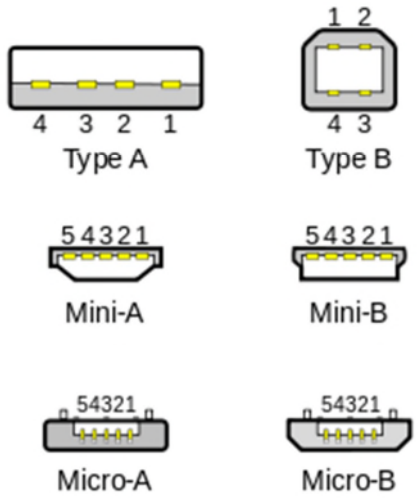


Figure II.3: Différents types de connecteurs USB

Les prises mâles de type A sont toujours tournés vers l'Amont. Les prises femelles de type A setrouveront généralement sur les hôtes et les Hubs. Par exemple, les prises femelles de type A sont courantes sur les cartes mères des ordinateurs et les Hubs. Les prises mâles de type B sont toujours connecté vers l'aval et par conséquent les prises femelles de type B se trouvent sur les appareils.

Il vient de paraître une spécificité On-the-Go qui ajoute la fonctionnalité pair à pair (peer to peer) à l'USB. D'où l'introduction des hôtes USB dans les téléphones mobiles et les agendas électroniques, de même qu'une particularité pour les prises mâles mini A, les prises femelles mini A et les prises femelles mini A-B. Tout porte à croire que nous devrions être inondés de câbles mini USB et d'une gamme de câbles convertisseurs de la taille mini à standard.

Fonction	Couleur	Numérode broche pour les types A et B	Numéro de broche pour le type mini B
Alimentation +5 V (VBUS)	Rouge	1	1
Données (D-)	Blanc	2	2
Données (D+)	Vert	3	3
Masse (GND)	Noir	4	5

Tableau II. 1: Brochage des différents connecteurs USB

On utilise des couleurs standards pour les fils intérieurs des câbles USB de façon à faciliter l'identification des fils d'un constructeur à un autre (tableau II.1). La normalisation précise les différents paramètres électriques pour les câbles.

IV CARACTERISTIQUES ELECTRIQUES:

Comme nous en avons déjà discuté, l'USB utilise une paire de transmission différentielle pour les données. Celle-ci étant codé en utilisant le NRZI et est garni de bits pour assurer les transitions adéquates dans le flot de données. Sur les appareils à vitesse basse et pleine un '1' différentiel est transmis en mettant D+ au-dessus de 2,8V grâce à une résistance de 15k ohms relié à la masse et D- en dessous de 0,3V avec une résistance de 1,5k ohms relié à 3,6V. D'autre part un différentiel '0' correspond à D- plus grand que 2,8V et D+ inférieur à 0,3V avec les mêmes résistances de rappel état haut/bas adéquates.

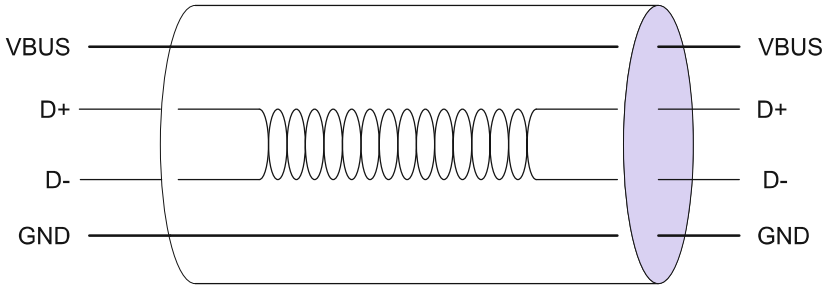


Figure II.4: . Câble USB

Le récepteur défini un différentiel '1' avec D+ plus grand de 200 mV que D- et un différentiel '0' avec D+ plus petit de 200 mV que D-. La polarité du signal est inversée en fonction de la vitesse du BUS. En conséquence les états référencés par les termes 'J' et 'K' sont utilisés pour signifier les niveaux logiques. En vitesse basse, un état 'J' est un différentiel '0'. En vitesse haute, un état 'J' est un différentiel '1'.

Les émetteurs / récepteurs USB comprendront à la fois des sorties différentiels et uniques (non complémentaires). Certains états de BUS USB sont indiqués par des signaux à **sorties uniques** (*single ended zero*) sur D+, D- ou les deux. Par exemple un zéro à sorties uniques ou **SE0** peutêtre utilisé pour signifier la réinitialisation d'un appareil s'il est maintenu plus de 10 ms. On génèreun **SE0** en maintenant D+ et D- en position basse (inférieur à 0,3V). Les sorties uniques etdifférentielles sont importantes d'être notées si vous utilisez un émetteurs / récepteurs et un FPGAcomme appareil USB. Vous ne pouvez pas vous contenter simplement d'échantillonner la sortiedifférentielle.

Le BUS basse et pleine vitesse a une impédance caractéristique de 90 Ohms +/-15%. Il est donc important d'observer la documentation technique lorsque vous sélectionnez les résistances des caractéristiques électriques séries pour D+ et D- afin d'équilibrer l'impédance. Toute documentation technique devrait spécifier ces valeurs et tolérances.

V IDENTIFICATION DE LA VITESSE:

Un appareil USB doit indiquer sa vitesse en mettant soit D+ ou D- à 3,3V. Un appareil pleine vitesse, représenté plus bas utilisera une résistance de rappel rattaché à D+ pour se signaler comme tel. Ces résistances de rappel à l'extrémité de l'appareil seraient aussi utilisées par l'hôte ou Hub pour détecter la présence d'un appareil connecté à son port. Sans résistance de rappel, l'USB suppose qu'il n'y a rien de connecté au BUS. Certains appareils possèdent cette résistance intégrée sur le silicium, pouvant être connecté ou non sous commande micro programmée, d'autres exigent une résistance externe.

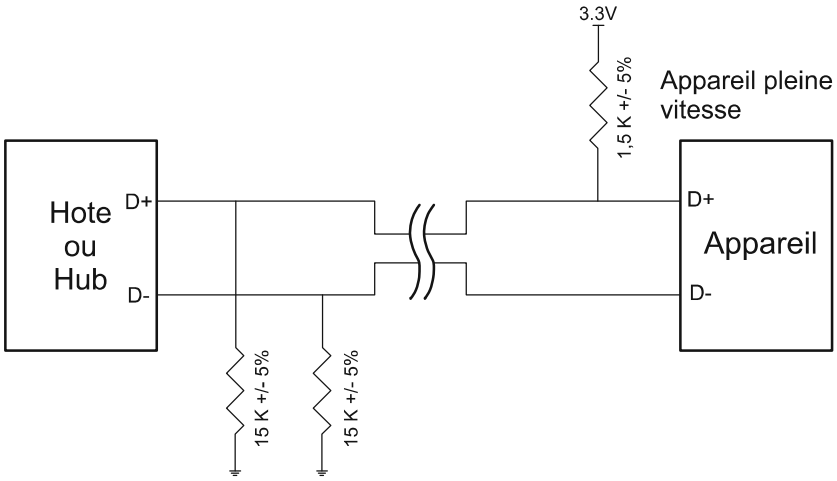


Figure II.5: Appareil pleine vitesse avec résistance de rappel état haut branché sur D+

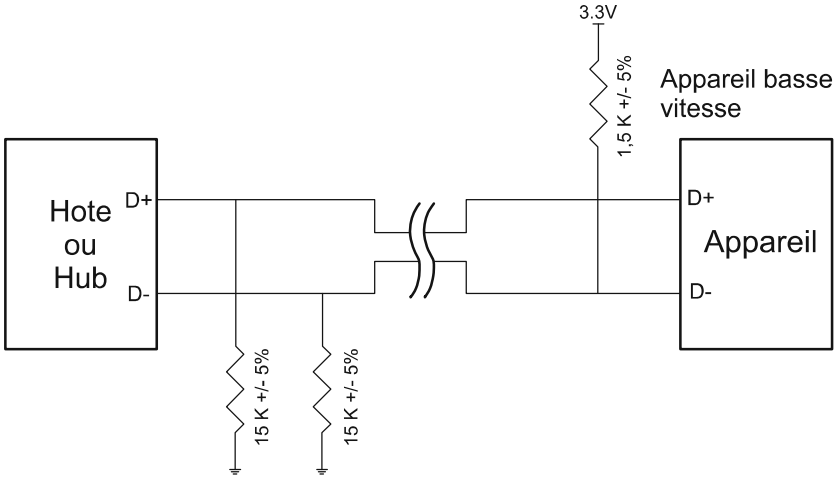


Figure II.6: Appareil pleine vitesse avec résistance de rappel état haut branché sur D-

Vous noterez que nous n'avons pas inclus d'identification de vitesse pour le mode haute vitesse Les appareils haute vitesse démarreront dès qu'ils seront connectés en tant qu'appareils pleine vitesse (1,5 kOhms à 3,3V). Une fois qu'il sera attaché il émettra un **Chirp** à haute vitesse pendant la réinitialisation et établira une connexion à grande vitesse si le Hub le supporte. Si l'appareil fonctionne en mode haute vitesse, alors la résistance de rappel est retirée pour équilibrer la ligne.

Toutefois un appareil haute vitesse ne peut pas supporter le mode basse vitesse. Il devraitseulement supporter le mode pleine vitesse nécessaire en début de connexion, ensuite le mode haute vitesse s'il réussit par être négocié. Un appareil face aval à détection d'anomalies pour l'USB 2.0 (Hub ou Hôte) doit supporter les 3 modes, Haute, pleine et basse vitesse.

VI L'ALIMENTATION USB (V_{BUS}):

Un des avantages de l'USB réside dans ces appareils alimentés par le Bus. Ceux-ci obtiennent leur alimentation à partir du Bus et ne demande aucune prise externe et câble additionnel. Cependant beaucoup de gens se focalisent sur cette option sans prendre en compte au préalable tous les critères nécessaires.

Un appareil USB précise sa consommation électrique exprimée en unité de 2mA dans le descripteur de la configuration que nous examinerons plus tard. Un appareil ne peut pas augmenter sa consommation électrique plus qu'il n'est précisé pendant l'énumération, même s'il perd de la puissance externe. Il existe 3 classes de fonctions USB :

- Les fonctions alimentées par le Bus à basse puissance (Low-Power)
- Les fonctions alimentées par le Bus à haute puissance (High-Power)
- Les fonctions auto alimentées (Self-powered)

Les fonctions alimentées par le Bus à basse puissance tirent toute leur puissance de V_{Bus} et nepeuvent en tirer que la charge d'une unité. La spécification USB défini une charge d'unité à 100mA. Les fonctions alimentées par le Bus à basse puissance peuvent aussi être conçues pour travailler à une tension de V_{BUS} tombant à 4,4V et montant à un maximum de 5,25V mesuré à la prise amont de l'appareil. Pour beaucoup d'appareil 3,3V, des régulateurs LDO sont obligatoires.

Les fonctions alimentées par le Bus à haute puissance tireront toute leur puissance du Bus et ne pourront tirer plus d'une unité de charge jusqu'à ce qu'elles aient été configurées, après quoi elles pourront tirer cinq unités de charge (500mA max) pourvu que cela soit demandé dans son descripteur. Les fonctions du Bus à Haute puissance doivent être capables d'être détectées et énumérées à un minimum de 4,40V. Lorsqu'elles fonctionnent à pleine charge, un V_{BUS} minimum de 4,75V est précisé avec un maximum de 5,25V. Une fois de plus, ces mesures sont prises à la prise mâle amont.

Les fonctions auto alimentées peuvent tirer jusqu'à une unité et faire dériver le reste de leur alimentation d'une source extérieure. Si cette source extérieure venait à manquer, il doit y avoir des réserves en place de manière à ne pas tirer plus d'une unité de charge du Bus. Les fonctions auto alimentées sont plus faciles à concevoir au niveau de la spécification car il ne peut guère y avoir de problèmes en ce qui concerne l'alimentation électrique. La charge alimentée par le Bus à une seule unité permet la détection de l'énumération d'appareil sans avoir besoin d'une alimentation principale/secondaire du secteur.

Aucun appareil USB, qu'il soit alimenté par le Bus ou bien auto alimentée ne peut piloter V_{Bus} surson port face amont. Si V_{Bus} est perdu, l'appareil a une durée de 10 secondes pour retirer l'alimentation des résistances de rappel de D+/D- utilisées pour l'identification de la vitesse.

Les autres considérations de VBUS sont l'appel de courant qui doit être limité. Ceci est souligné dans le paragraphe 7.2.4.1 des spécifications USB et est souvent laissé de côté. L'appel de courant est soutenu au niveau de la capacité de tête de votre appareil entre V_{BUS} et la masse. Les spécifications précisent par conséquent que la capacité de découplage maximum que vous pouvez avoir sur votre appareil est de 10µF. Quand vous déconnectez l'appareil après que le courant soit passé par le câble USB inductif, une grande tension de retour peut se produire sur l'extrémité ouverte du câble. Pour éviter ceci, on recommande une capacité de découplage V_{BUS} minimum de 1µF.

Pour l'appareil habituel alimenté sur un Bus, celui-ci ne peut drainer plus de 500 mA, ce qui est fort raisonnable.

VII COURANT DE VEILLE:

Le mode Veille est obligatoire sur tous les appareils. Pendant son temps d'action, d'autrescontraintes surviennent. Le courant maximum de veille est proportionnel à l'unité de charge. Pour un appareil d'une unité de charge (par défaut) le courant de veille maximum est de 500µA. Ceci comprend le courant dû aux résistances de rappel sur le Bus. Au niveau du Hub, D- et D+ possèdent des résistances de rappel niveau bas de 15 kOhm. Pour des raisons de consommation électrique, la résistance de rappel niveau bas de l'appareil est montée en série avec la résistance de rappel niveau haut de 1,5 kOhm, totalisant ainsi une charge de 16,5 kOhm sur VTERM habituel de 3,3V. Par conséquent cette résistance draine un courant de 200 µA avant même que l'on commence.

Beaucoup de développeur demande au forum des fournisseurs USB, quelles sont les complications si l'on dépasse cette limite? On comprend bien, que la plupart des Hôtes et des Hubs n'ont pas la possibilité de détecter de surcharge de cette importance et ainsi si vous drainez peut-être 5 mA ou même 10 mA cela devrait aller, tout en gardant à l'esprit qu'à la fin de la journée, votre appareil a enfreint la spécification USB. Toutefois en fonctionnement normal, si vous essayez de dépasser les 100 mA ou la charge permise qui vous est indiqué, alors attendez-vous à ce que le Hub ou Hôte le détecte et déconnecte votre appareil, dans l'intérêt de l'intégrité du Bus.

Bien sûr, ces problèmes de conception peuvent être évités si vous choisissez de concevoir un appareil auto alimenté. Les courants de veille peuvent ne pas être d'un grand intérêt pour les ordinateurs de bureau mais avec l'introduction de la spécification On-The-Go, nous allons commencer à voir des Hôtes USB construit dans les téléphones mobiles et agendas électroniques. La consommation électrique utilisée à partir de ces appareils affectera défavorablement la durée de vie de la batterie.

VIII ACCES AU MODE VEILLE:

Un appareil USB entrera en veille lorsqu'il n'y a aucune activité sur le Bus pendant plus de 3ms. Il dispose ensuite de 7ms de plus pour éteindre l'appareil et ne prendre que le courant de veille désigné, ne prenant ainsi que le courant de veille nominal à partir du Bus 10ms après que l'activité du Bus ce soit arrêté. Afin de le maintenir connecté à un Hub ou à un Hôte mis en veille, l'appareil doit encore fournir de l'alimentation à ces résistances de rappel de sélection de vitesse pendant le mode veille.

L'USB possède un démarrage de trames de bits (frame packet) ou bien d'entretien qui sontenvoyés périodiquement sur le Bus. Ceci empêche un Bus inutilisé d'entrer dans le mode veille en l'absence de données.

- Un Bus haute vitesse enverra une trame toutes les 125.0 µs ±62.5 ns.
- Un Bus pleine vitesse enverra une trame toutes les 1.000 ms ±500 ns.
- Un Bus basse vitesse aura un dispositif d'entretien qui est un EOP (End Of Packet ou Fin De Paquet) toutes les 1ms simplement en l'absence de données basse vitesse.

Le terme veille Global (Global Suspend) est utilisé lorsque le Bus USB entier entre collectivementdans le mode veille. Cependant les appareils sélectionnés peuvent être mis en veille en ordonnantau Hub sur lequel l'appareil est aussi connecté. On fait référence à cette opération comme mode"veille sélective".

L'appareil reprendra son fonctionnement quand il recevra tout signal qui n'est pas en attente. Si unappareil possède une mise en service de réveil retardé, alors il devra signaler à l'Hôte de reprendreà partir du mode veille.

IX LES PROTOCOLES USB:

Contrairement à la RS232 et des interfaces sérielles similaires où le format des données envoyées n'est pas défini, l'USB est composé de plusieurs couches de protocoles. En fait la plupart des Circuits Intégrés contrôleur d'USB s'occuperont de la couche inférieure, la rendant ainsi presque invisible au regard du concepteur final.

Chaque transaction USB consiste d'un

- Paquet Jeton (Token) (en tête définissant ce qu'il attend par la suite)
- Paquet DATA optionnel (contenant la " charge utile " (payload))
- Paquet d'Etat (utilisé pour valider les transactions et pour fournir des moyens de
- corrections d'erreurs).

Comme nous en avons déjà discuté, l'USB est un Bus géré par l'hôte. L'hôte initie toutes les transactions. Le premier paquet, aussi appelé Jeton est produit par l'hôte pour décrire ce qui va suivre et si la transaction de données sera en lecture ou écriture et ce que sera l'adresse de l'appareil et la terminaison désignée. Le paquet suivant est généralement un paquet de données transportant la " charge utile " et est suivi par un paquet " poignée de mains " (handShaking), signalant si les données ou le jeton ont été reçus correctement ou si la terminaison est bloquée, ou n'est pas disponible pour accepter de données.

X LES CHAMPS DE PAQUET USB ORDINAIRES:

Les données sur le BUS USB sont transmises avec le bit LSB en premier. Les paquets USB se composent des champs suivants :

X.1 Le champ SYNC :

Tous les paquets doivent commencer avec un champ Sync. Le champ Sync fait de 8 bits de long pour la basse et pleine vitesse ou 32 bits pour la haute vitesse est utilisé pour synchroniser l'horloge du récepteur avec celle de l'émetteur. Les 2 derniers bits indiquent l'endroit où le champ PID commence.

X.2 Le champ PID :

PID signifie Paquet ID. Ce champ est utilisé pour identifier le type de paquet qui est envoyé. Le **tableau II.2**montre les valeurs possibles.

Groupe	Valeur PID	Identificateur Paquet
Token	0001	OUT Token
Jeton	1001	IN Token

	0101	SOF Token
	1101	SETUP Token
Data **Données**	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake **Poignée** **De mains**	0010	ACK Handshake
	1010	NAK Handshake
	1110	STALL Handshake
	0110	NYET(No Response Yet)
Special	1100	PREamble
	1100	ERR
	1000	Split
	0100	Ping

Tableau II. 2: Les différentes valeurs du PID

SOF = Start of Frame; Début de Trame

SETUP = Installation, configuration

ACK = Acknowledge; Validation

NAK = No Acknowledge; Pas de validation

STALL = Bloqué

PREamble = Synchroniseur initial

Split = Partager, Fractionner

Ping = S'assurer d'une bonne connexion

Il y a 4 bits pour le PID, toutefois pour s'assurer qu'il a été reçu correctement, les 4 bits sont complémentéset répétés faisant un PID de 8 bits au total. Le format résultant figure ci-dessous.

PID_0	PID_1	PID_2	PID_3	$nPID_0$	$nPID_1$	$nPID_2$	$nPID_3$

Figure II.7: Structure du champ PID

X.3 Le champs ADDR :

Le champ adresse détermine à quel appareil le paquet est destiné. Sa longueur de 7 bits, lui permet de supporter 127 appareils. L'adresse 0 n'est pas valide, tant qu'un appareil qui n'a pas encore d'adresse attribuée, doit répondre aux paquets envoyés d'adresse 0.

$Addr_0$	$Addr_1$	$Addr_2$	$Addr_3$	$Addr_4$	$Addr_5$	$Addr_6$

Figure II.8: Structure du champ ADDR

X.4 Le champs ENDP :

Le champ de terminaison est composé de 4 bits, autorisant 16 terminaisons possibles. Les appareils basse vitesse, toutefois peuvent seulement avoir 2 terminaisons additionnelles au-dessus du canal de communication par défaut (4 terminaisons maximales).

$Endp_0$	$Endp_1$	$Endp_2$	$Endp_3$

Figure II.9: Structure du champ ENDP

X.5 Le champs CRC :

Les Contrôles à Redondance Cyclique sont exécutés sur les données à l'intérieur du paquet de charge utile. Tous les paquets jetons ont un CRC de 5 bits tandis que les paquets de données ont un CRC de 16 bits.

X.6 Le champs EOP :

Fin de Paquet. Signalé par une sortie unique zéro (SE0) pendant une durée approximative de 2 bits suivie par un " J " d'une durée de 1 bit.

XI LES TYPES DE PAQUET USB:

L'USB a quatre types différents de paquet. Les paquets jetons indiquent le type de la transaction qui va suivre, les paquets de données contiennent la charge utile, les paquets " poignée de mains " sont utilisés pour valider les données ou rapporter les erreurs et les paquets début de trame (SOF) indiquent le commencement d'une nouvelle trame.

XI.1 Les paquets jetons :

Il y a 3 sortes de paquets Jetons :

- **In :** Informe l'appareil USB que l'hôte veut lire des informations.
- **Out** : Informe l'appareil USB que l'hôte veut envoyer des informations.
- **Setup :** Utilisé pour commencer les transferts de commande.

Les paquets jetons doivent se conformer au format suivant :

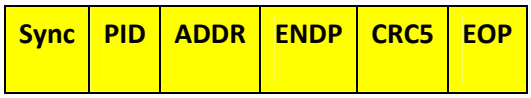


Figure II.10: Structure d'un paquet jeton

XI.2 Les paquets de données :

Il y a 2 sortes de paquets de données, chacun étant capable de transmettre plus de 1024 octets de données :

- Data0
- Data1

Le mode haute vitesse définit 2 autres PIDs de données, DATA2 et MDATA. Les paquets de données ont le format suivant :

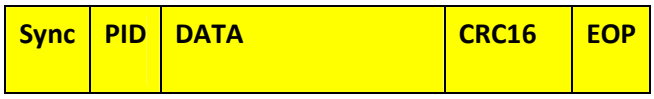


Figure II.11: Structure d'un paquet de données

La taille maximale de données " charge utile " pour les appareils basse vitesse est de 8 octets.La taille maximale de données " charge utile " pour les appareils pleine vitesse est de 1023 octets. La taille maximale de données " charge utile " pour les appareils haute vitesse est de 1024 octets. Les données doivent être envoyées en multiple d'octets.

XI.3 Les paquets " poignée de mains " :

Il y a 3 sortes de paquets " poignée de mains " qui font simplement partie du PID :

- **ACK** : validant que le paquet a été reçu correctement.
- **NAK :** rapporte que temporairement l'appareil ne peut ni envoyer ou recevoir des données. Aussi utilisé pendant les transactions d'interruptions pour avertir l'hôte qu'il n'a pas de données à envoyer.
- **STALL** (Bloqué) : L'appareil se retrouve dans un état qui va exiger l'intervention de l'hôte.

Les paquets " poignée de mains " ont le format suivant :

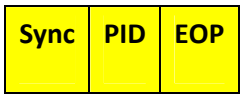


Figure II.12: Structure d'un paquet « poignée de mains »

XI.4 Les paquets début de trame (SOF) :

Le paquet SOF composé d'une trame de 11 bits est envoyé par l'hôte toutes les 1ms ± 500ns sur un bus pleine vitesse ou bien toutes les 125µs ± 0,0625µs sur un bus haute vitesse.

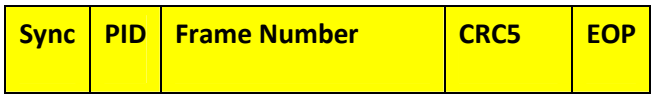


Figure II.13: Structure d'un paquet début de trame

XII LES FONCTIONS USB:

Quand nous pensons à un appareil USB, nous pensons à un périphérique USB, mais un appareil USB peut signifier un appareil émetteur / récepteur USB sur l'hôte ou périphérique, un HUB USB ou un circuit intégré contrôleur d'hôte ou un appareil périphérique USB. Par conséquent le standard fait références aux fonctions USB qui peuvent être considérées comme appareil USB qui fournissent une possibilité ou une fonction comme une imprimante, un lecteur Zip, un scanner, un modem ou un autre périphérique.

La plupart des fonctions USB manipulent les protocoles USB bas niveau jusqu'à la couche transaction dans le silicium.

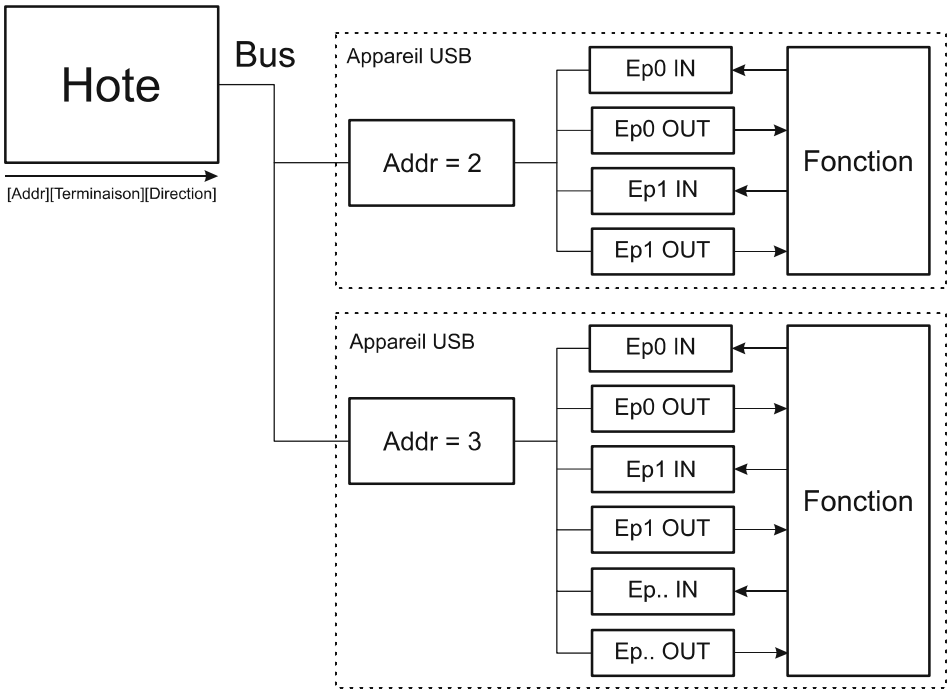


Figure II.14: . Exemple de fonction USB

La plupart des fonctions auront des séries de tampons (buffers), généralement de 8 octets de long. Chaque tampon appartiendra à une terminaison EP0 IN, EP0 OUT etc... Supposons par exemple, que l'hôte envoie une demande de descripteur d'appareil. La fonction matérielle lira le paquet d'installation et déterminera à partir du champ adresse si le paquet est pour lui-même, et si c'est le cas, il copiera la " charge utile " du paquet de données suivant au tampon de la terminaison appropriée, dictée par la valeur dans le champ de la terminaison du jeton d'installation. Il enverra ensuite un paquet " poignée de mains " pour valider la réception de l'octet et génèrera une interruption interne à l'intérieur du semi-conducteur / microcontrôleur pour la terminaison appropriée signalant qu'il a reçu un paquet. C'est en principe déjà intégré dans la matière (silicium).

Le logiciel a maintenant une interruption, et doit lire le contenu du tampon de terminaison et analyser la demande de descripteur d'appareil.

XIII LES TERMINAISONS (ENDPOINT):

Les terminaisons peuvent être décrites comme émetteurs ou récepteurs de données. Du fait que le bus est régi par l'hôte, les terminaisons se présentent à la fin de la chaîne de communications sur la fonction USB. Au niveau de la couche logicielle, le pilote (driver) logiciel de votre appareil va envoyer, par exemple, un paquet à vos appareils EP1. A la sortie de l'hôte, la donnée aboutira au tampon EP1 OUT. Votre microprogramme pourra alors lire à loisir cette donnée. S'il veut retourner la donnée, la fonction ne peut pas simplement écrire sur le BUS comme celui-ci est contrôlé par l'hôte. Par conséquent il écrit la donnée dans EP1 IN qui s'installe dans le tampon jusqu'à ce que l'hôte envoie un paquet IN à cette terminaison demandant la donnée. Les terminaisons peuvent être aussi considérées comme l'interface entre le matériel de l'appareil de fonction et le microprogramme s'exécutant sur ce même appareil.

Tous les appareils doivent prendre en charge la terminaison zéro. C'est la terminaison qui reçoit la totalité de la commande des appareils et des demandes d'états pendant l'énumération et tant que l'appareil est opérationnel sur le bus.

XIV LES CANAUX DE COMMUNICATIONS (PIPES):

Tandis que l'appareil envoie et reçoit des données sur une succession de terminaisons, le logiciel client transfère des données à travers des canaux de communications. Un canal de communication (Pipe) est une connexion logique entre l'hôte et les terminaisons. Les canaux de communications auront aussi un ensemble de paramètres qui leur seront associés tels que : combien de bande passante leur est allouée, quel type de transfert (Commande, Bloc, Iso ouInterruption) ils utilisent, la direction du flux de données et les tailles maximales du paquet / tampons. Par exemple le canal de communication par défaut est un canal bidirectionnel composé d'une terminaison zéro IN et d'une terminaison zéro OUT avec un type de transfert de commande.

L'USB définit 2 types de canaux de communications :

- **Les flux de données (Stream Pipes)** n'ont pas de format USB défini. Cela veut dire que vous pouvez envoyer n'importe quelle sorte de données par un flux de données et que vous pouvez rapporter les données de sorties à l'autre extrémité. Les données circulent séquentiellement et ont une direction prédéfinie, soit en entrée soit en sortie. Les flux de données supporteront les types de transferts en Bloc, Isochrone et d'Interruptions. Les flux de données peuvent soit être contrôlés par l'hôte ou l'appareil.
- **Les canaux de messages (Message Pipes)** ont un format USB défini. Ils sont contrôlés par l'hôte et sont initiés par une demande émanant de l'hôte. Les données sont ensuite transférées dans la direction voulue, dictées par la demande. Par conséquent les canaux de messages permettent aux données de circuler dans les deux directions mais ne prendront seulement en charge que les transferts de commande.

XV LES TYPES DE TERMINAISONS (ENDPOINTS):

La spécification du Bus Série Universel définit 4 types de transferts ou de terminaisons :

- Transferts de commande
- Transferts d'interruption
- Transferts Isochrone
- Transferts en Bloc (BULK)

XV.1 Les transferts de commande:

Les Transferts de commande sont régulièrement utilisés pour les opérations de commande et d'état. Ils sont essentiels pour installer un appareil USB avec toutes les fonctions d'énumération qui seront exécutés en utilisant les Transferts de commande. Ils surviennent généralement en paquets directs et par rafales qui sont initiés par l'hôte et utilisent le meilleur rendement de livraison. La longueur du paquet du Transfert de commande pour appareil basse vitesse doit être de 8 octets, les appareils pleine vitesse autorise une taille de paquet de 8, 16, 32 ou 64 octets et les appareils haute vitesse doivent avoir une taille de paquet de 64 octets.

Un Transfert de commande peut avoir plus de 3 étapes.

XV.1.1 L'étape d'installation (Setup Stage) :

Elle se passe lorsqu'une demande est envoyée. Elle est composée de 3 paquets. Le jeton (token) d'installation envoyé le premier est celui qui contient l'adresse et le numéro de la terminaison. Le paquet de données est envoyé après et a toujours un type PID de Data0 et inclut un paquet d'installation qui détaille le type de la demande. Nous détaillerons le paquet d'installation plus tard. Le dernier paquet est une poignée de mains utilisé pour valider la bonne réception ou pour indiquer une erreur. Si la fonction reçoit correctement la donnée d'installation (CRC et PID etc...Ok) elle répond avec ACK, sinon elle ignore la donnée et n'envoie pas un paquet de poignée de mains. Les fonctions ne peuvent pas émettre un paquet STALL ou NAK en réponse à un paquet d'installation.

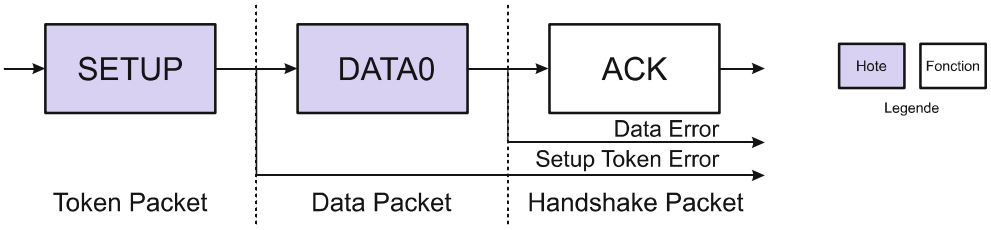


Figure II.15: Etape d'installation

XV.1.2 L'étape de données (Data Stage) :

Elle est facultative consiste en un ou plusieurs transferts IN (Entrée) ou OUT (sorties). La demande d'installation indique la quantité de données qui doit être envoyée dans cette étape. Si elle dépasse la taille maximale du paquet, les données seront envoyées en plusieurs transferts, chacune ayant la longueur maximale du paquet à l'exception du dernier paquet.

L'étape de données comporte 2 scénarios différents selon la direction du transfert de données :

- **IN** (Entrée): Quand l'hôte est prêt à recevoir les données de commande, il émet un jeton (token) IN. Si la fonction reçoit le jeton IN avec une erreur, c'est-à-dire que le PID ne correspond pas avec les bits inversés du PID, il ignore donc le paquet. Si le jeton est reçu correctement, l'appareil peut soit répondre avec un paquet de données contenant les données de commande à envoyer, soit un paquet "d'arrêt" signalant que la terminaison a eu une erreur soit un paquet NAK signalant à l'hôte que la terminaison fonctionne, mais provisoirement n'a pas de données à envoyer.
- **OUT** (Sortie): Quand l'hôte a besoin d'envoyer à l'appareil un paquet de données de commande, il émet un jeton OUT suivi par un paquet de données contenant les données de commande comme "charge utile" (payload). Si une partie du jeton OUT ou du paquet de données est altéré alors la fonction ignore le paquet. Si le buffer de terminaison de la fonction était vide et qu'il a cadencé les données dans le buffer de terminaison, il produit un ACK avisant l'hôte qu'il a bien reçu les données. Si le buffer de terminaison n'est pas vide à cause du traitement du paquet précédent, alors la fonction retourne un NAK. Toutefois si la terminaison comporte une erreur et que son bit "halt" ai été positionné, elle retourne un STALL (Bloqué).

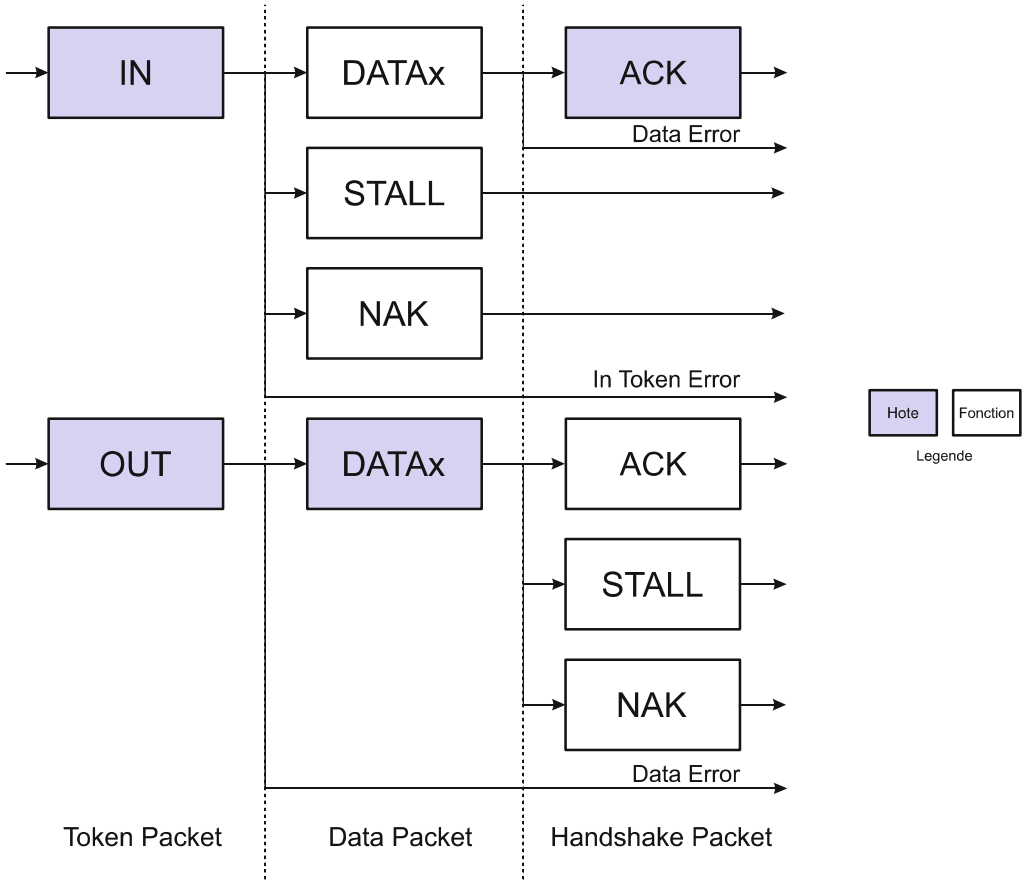


Figure II.16: Etape de données

XV.1.3 L'étape d'état (Status Stage) :

Elle rend compte des états de l'ensemble des demandes et cette fois encoreselon la direction du transfert. Le rapport d'état est toujours réalisé par la fonction.

- **IN** (Entrée): Si l'hôte envoie un (ou des) jeton(s) IN pendant l'étape de données pour recevoir des données, alors l'hôte doit valider la bonne réception de ces données. Ceci est réalisé par l'hôte qui envoie un jeton OUT suivi par un paquet de données de longueur nul. La fonction peut maintenant rendre compte de son état dans l'étape poignée de mains. Un ACK indique que la fonction a achevé la commande et qu'elle est maintenant prête à accepter une autre commande. Si une erreur s'est produite pendant l'exécution de cette commande, alors la fonction émettra un STALL (Bloqué). Toutefois si la fonction continue l'exécution, elle retourne un NAK indiquant à l'hôte la nécessité de répéter l'étape d'état ultérieurement.

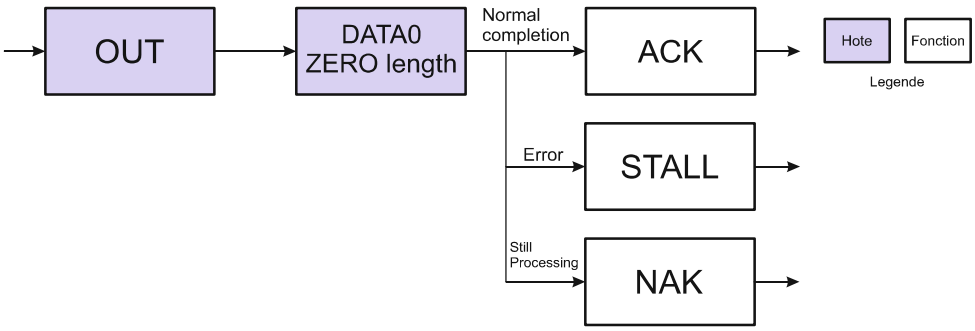


Figure II.17: . Etape d'état (entrée)

- **OUT** (Sortie):Si l'hôte envoie un (ou des) jeton(s) OUT pendant l'étape de données pour transmettre des données, la fonction validera la bonne réception des données en envoyant un paquet de longueur nul en réponse à un jeton IN. Toutefois si une erreur intervenait, cela déboucherait sur un STALL ou si la fonction était encore occupée à traiter les données, cela déboucherait sur un NAK demandant à l'hôte de retenter l'étape d'état ultérieurement.

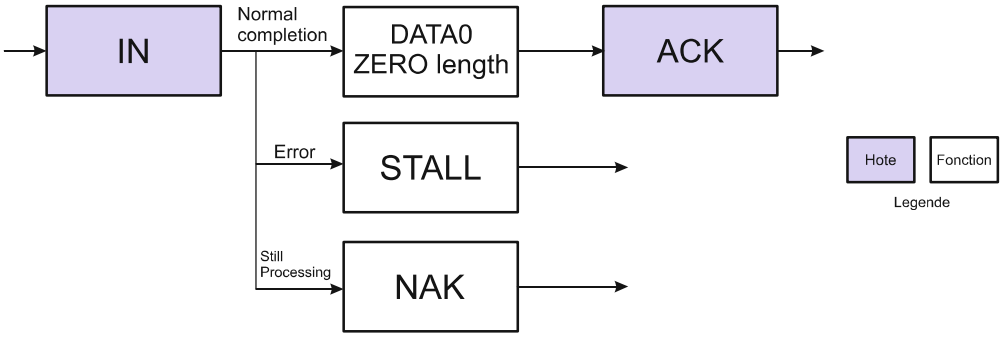


Figure II.18: Etape d'état (sortie)

Pour illustrer ce que nous venons de voir concernant les différentes étapes qui composent le transfert de commande, nous allons expliquer ceci par un exemple.

Supposonsque l'hôte veuille demander un descripteur d'appareil pendant l'énumération. Les paquets qui sont envoyés sont les suivants:

L'hôte enverra un jeton d'installation disant à la fonction que le paquet suivant est un paquet d'installation. Le champ adresse fixera l'adresse de l'appareil à qui l'hôte a demandé le descripteur. Le numéro de la terminaison devrait être un zéro, indiquant une ligne défectueuse. L'hôte enverra alors un paquet DATA0. Celui-ci aura 8 octets de "charge utile" (payload) correspondant à la Demande de Descripteur d'Appareil comme souligné au chapitre 9 de la spécification USB. La fonction USB annoncera alors que le paquet d'installation a été lu correctement et sans aucune erreur. Si le paquet était reçu altéré, l'appareil ignorerait tout simplement ce paquet. L'hôte renverra alors le paquet après un court délai.

1.Jeton d 'installation	Sync	PID	ADDR	ENDP	CRC5	EOP	Adresse et N° de terminaison
2.Paquet DATA0	Sync	PID	DATA0		CRC16	EOP	Demande de Descripteur d 'Appareil
3.Poignée de mains ACK	Sync	PID	EOP				L 'appareil valide le paquet d 'installation

Les 3 paquets ci-dessus représentent la première transaction USB. L'appareil USB décodera maintenant les 8 octets reçus, et déterminera que c'était une demande de Descripteur d'Appareil. L'Appareil tentera d'envoyer le Descripteur d'Appareil, qui sera la transaction USB suivante.

1.Jeton IN	Sync	PID	ADDR	ENDP	CRC5	EOP	Adresse et N° de terminaison
2.Paquet DATA0	Sync	PID	DATA0		CRC16	EOP	Les 8 premiers octets du Descripteur d 'appareil
3.Poignée de mains ACK	Sync	PID	EOP				L 'hôte valide le paquet
1.Jeton IN	Sync	PID	ADDR	ENDP	CRC5	EOP	Adresse et N° de terminaison
2.Paquet DATA1	Sync	PID	DATA1		CRC16	EOP	Les 4 derniers octets + du remplissage
3.Poignée de mains ACK	Sync	PID	EOP				L 'hôte valide le paquet

Dans ce cas nous assumons que la taille maximale de "charge utile" (payload) est de 8 octets. L'hôte envoie le jeton IN, disant à l'Appareil qu'il peut maintenant envoyer des données pour cette terminaison. Comme la taille maximale du paquet est de 8 octets, nous devons scinder les 12 octets du Descripteur d'Appareil en morceaux avant de pouvoir les envoyer. Chaque morceau doit être de 8 octets excepté la dernière transaction. L'hôte validera chaque paquet de données que nous lui enverrons.

Une fois que le Descripteur d'Appareil est envoyé, il s'ensuit une transaction d'état. Pour le cas où les transactions seraient réussies, l'hôte enverra un paquet de longueur Nul indiquant que l'ensemble de transactions était correct. La fonction répond alors à ce paquet de longueur Nul indiquant son état.

1.Jeton OUT	Sync	PID	ADDR	ENDP	CRC5	EOP	Adresse et N° de terminaison
2.Paquet DATA0	Sync	PID	DATA0		CRC16	EOP	Paquet de longueur Nulle
3.Poignée de mains ACK	Sync	PID	EOP				L 'appareil valide la transaction entière

XV.2 Les transferts d'interruption:

Celui qui eut une expérience de demandes d'interruption sur microcontrôleurs saura que les interruptions sont générées par l'appareil. Toutefois sous USB, si un appareil demande l'attention de l'hôte, il doit attendre que l'hôte l'interroge avant de signaler qu'il a besoin d'une attention urgente.

Les transferts d'interruption sont caractérisés par :

- Temps de retard (ou Latence) garanti
- Ligne de flux - Unidirectionnel
- Détection d'erreur et nouvel essai sur période suivante

Les transferts d'interruptions sont communément non périodiques, les petits appareils qui ont initié la communication ont besoin de temps de retard limité.

- La taille maximale de "charge utile" (payload) pour des appareils basse vitesse est de 8 octets
- La taille maximale de "charge utile" (payload) pour des appareils pleine vitesse est de 64 octets
- La taille maximale de "charge utile" (payload) pour des appareils haute vitesse est de 1024 octets

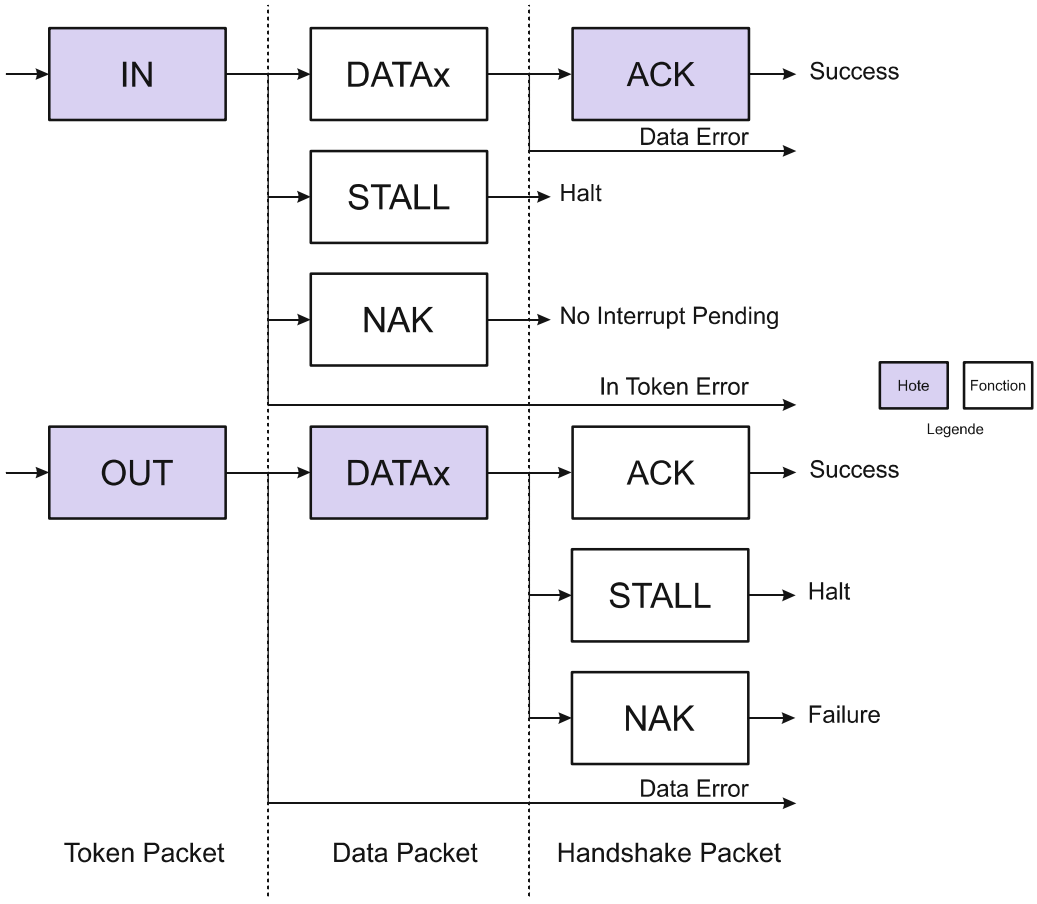


Figure II.19: Transaction d'interruption d'entée (IN) et de sortie (Out)

- **IN:** L'hôte interrogera périodiquement la terminaison d'interruption. Le taux d'interrogation est spécifié dans le descripteur de terminaison qui sera examiné plus tard. Chaque interrogation obligera l'hôte à envoyer un jeton IN. Si le jeton IN est altéré, la fonction ignore le paquet et continue la surveillance du Bus pour de nouveaux jetons.

Si une interruption a été mise en attente par l'appareil, la fonction enverra un paquet Data contenant des données ayant rapport à l'interruption quand il recevra le jeton IN. Sur des réceptions au niveau de l'hôte, celui-ci retournera un ACK. Toutefois si les données sont altérées, l'hôte ne mentionnera aucun état. Si, d'autre part, une condition d'interruption n'était pas présente quand l'hôte interrogeait la terminaison d'interruption avec un jeton IN, alors la fonction signale cet état en envoyant un NAK. Si une erreur se produisait sur cette terminaison, un STALL (Bloqué) serait envoyé en réponse à un jeton IN.

- **OUT:** Quand l'hôte veut envoyer à l'appareil les données d'interruptions, il émet un jeton OUT suivi par un paquet Data contenant les données d'interruption. Si une partie du jeton OUT ou du paquet Data est altéré alors la fonction ignore le paquet. Si le tampon de terminaison de la fonction était vide et qu'il ait cadencé les données dans le tampon de terminaison il émettrait un ACK prévenant l'hôte qu'il a reçu correctement les données. Si le tampon de terminaison n'est pas vide à cause du traitement d'un paquet précédent, alors la fonction retourne un NAK.

Toutefois si une erreur se produisait à cause de la terminaison et que son bit d'arrêt (Halt) ait été positionné, elle renverrait un STALL (Bloqué).

XV.3 Les transferts Isochrones:

Les Transferts Isochrones se produisent continuellement et périodiquement. Ils contiennent généralement des informations à durée de vie critique, tel des trains de données audio ou vidéo. S'il y avait un retard ou une reprise de données dans un flot de données audio, alors on pourrait s'attendre à de l'audio par intermittence contenant des signaux transitoires.

Les transferts isochrones sont caractérisés par :

- Un accès garanti à la bande passante USB.
- Un temps d'attente limité.
- Des flux de données - Unidirectionnel.
- La détection d'erreur via le CRC, mais sans reprise ou garantie de livraison.
- Seulement les modes pleines et haute vitesse.
- Pas de données de basculement (basculage, cachées, de commutation).

La taille maximale de données en " charge utile " (payload) est spécifiée dans le descripteur de terminaison d'une terminaison Isochrone. Cela peut être un maximum de 1023 octets pour un appareil pleine vitesse et 1024 octets pour un appareil haute vitesse. Comme la taille maximale de données en " charge utile " va effectuer des exigences de bande passante du Bus, il est prudent de spécifier une taille de " charge utile " modérée. Si vous utilisez de grandes " charges utiles " il peut être aussi plus intéressant de spécifier une série d'interfaces alternatives avec des tailles de charges utiles Isochrones variables. Si pendant l'énumération l'hôte ne peut pas valider votre terminaison

Isochrone nommée à cause des restrictions de bande passante, il reste une solution de repli préférable à un échec complet. Les données qui ont été envoyées sur une terminaison Isochrone peuvent être inférieures à la taille pré négociée et peuvent varier en longueur de transaction en transaction.

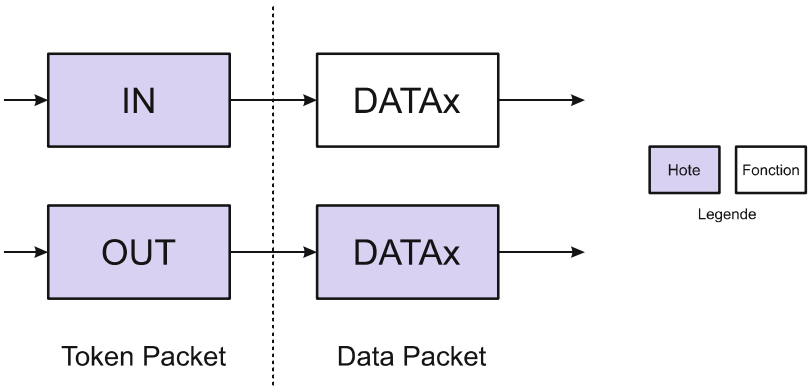


Figure II.20: Transaction Isochrone d'entée (IN) et de sortie (Out)

Les transactions Isochrones n'ont pas d'étape de poignée de mains et ne peuvent pas rendre compte des erreurs ou des conditions STALL / HALT ((Bloqué) / Arrêt).

XV.4 Les transferts En Bloc (BULK):

Les Transferts en Bloc peuvent être utilisés pour de grandes quantités de données sporadiques. De tels exemples pourraient inclure un travail d'impression envoyé à une imprimante ou une image provenant d'un scanner. Les Transferts en Bloc se prémunissent de correction d'erreurs sous la forme d'un champ CRC16 sur les données " charge utile " et sur les mécanismes de détection et de retransmission d'erreurs qui assure la transmission et la réception de données de manière infaillible.

Les Transferts en Bloc utiliseront une bande passante de réserve non attribuée sur le Bus après que toutes les autres transactions aient été allouées. Si le Bus est occupé avec de l'Isochrone et/ou de l'interruption, les données en bloc peuvent alors s'écouler doucement sur le Bus. En conséquence, les transferts en bloc devraient seulement être utilisés pour des communications insensibles au temps du fait de la non garantie du temps d'attente.

Les transferts en bloc sont caractérisés par :

- Utilisés pour de grandes quantités de données sporadiques.
- Détection d'erreurs via le CRC, avec la garantie de livraison.
- Pas de garantie de bande passante ou du temps d'attente minimum.
- Des flux de données - Unidirectionnel.
- Seulement les modes pleines et haute vitesse.

Les Transferts en Bloc sont seulement supportés par des appareils pleine et haute vitesse. Pourdes terminaisons pleine vitesse, la longueur maximale du paquet en Bloc est soit 8, 16, 32 ou 64octets. Pour des terminaisons haute vitesse, la longueur maximale du paquet peut aller jusqu'à512 octets. Si la charge utile des données est inférieure à la taille maximale du paquet, elle n'apas besoin d'être

remplie avec des zéros. Un transfert en Bloc est considéré comme completquand il a transféré la quantité exacte de données demandées, ou un paquet inférieur à la taillemaximale de la terminaison, ou un paquet de longueur zéro.

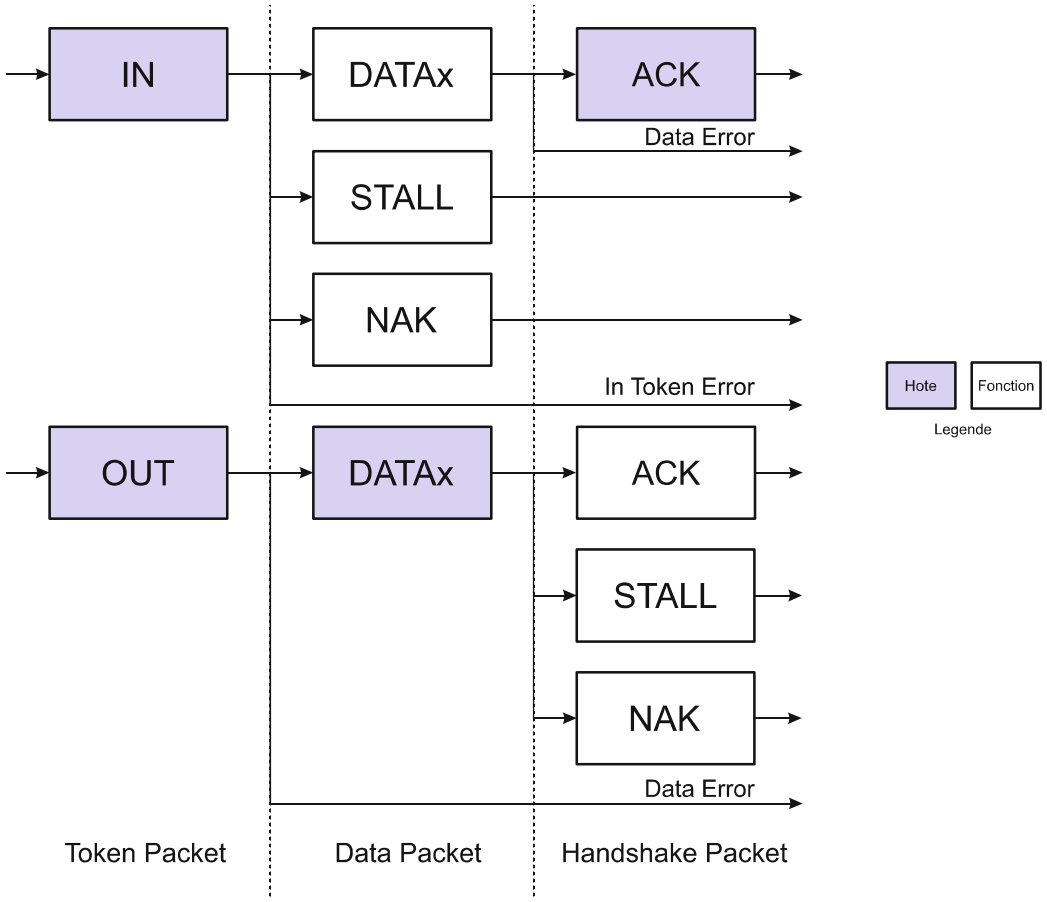


Figure II.21: Transaction en bloc d'entée (IN) et de sortie (Out)

- **IN:** Quand l'hôte est prêt à recevoir des données en Bloc, il émet un jeton IN. Si la fonction reçoit le jeton IN avec une erreur, il ignore le paquet. Si le jeton est reçu correctement, la fonction peut soit répondre avec un paquet DATA contenant les données en Bloc à envoyer ou bien un paquet Stall signalant que la terminaison a eu une erreur ou un paquet NACK signalant à l'hôte que la terminaison travaille, mais provisoirement n'a pas de données à envoyer.
- **OUT:** Quand l'hôte veut envoyer à la fonction un paquet de données en Bloc, il émet un jeton OUT suivi par un paquet DATA contenant les données en Bloc. Si une partie du jeton OUT ou du paquet DATA est altérée, alors la fonction ignore le paquet. Si le tampon de terminaison de la fonction est vide et qu'il a cadencé les données dans le tampon de terminaison, il émet un ACK prévenant l'hôte qu'il a reçu correctement les données. Si le tampon de terminaison n'est pas vide à cause du traitement d'un paquet précédent, alors la fonction retourne un NAK. Toutefois si la terminaison a eu une erreur et que son bit d'arrêt a été positionné, elle retourne un Stall (Bloqué).

XVI LA GESTION DE LA BANDE PASSANTE:

L'hôte est responsable de la bande passante du Bus. Elle est faîte par l'énumération (dénombrement) lors de la configuration des terminaisons Isochrones et d'interruptions et durant le fonctionnement du Bus. La spécification place des limites sur le Bus, l'autorisant à allouer plus de 90% pour des transferts périodiques (Interruption et Isochrone) sur un Bus pleine vitesse. Sur des Bus hautes vitesses, cette limitation se réduit à moins de 80% d'une micro-trame qui peut être allouée pour des transferts périodiques.

Aussi vous pouvez assez rapidement voir que si vous avez un Bus hautement saturé avec des transferts périodiques, les 10% restants sont laissés pour les transferts de contrôle et une fois qu'ils ont été attribués, les transferts en Bloc prendront ce qui reste.

XVII LES DESCRIPTEURS USB:

Tous les appareils USB ont une hiérarchie de descripteurs qui détaillent pour le compte de l'hôte des informations l'instruisant sur la nature de l'appareil, qui l'a réalisé, quelle version USB il supporte, de combien de manières il peut être configuré, le nombre de terminaisons et leurs types etc…

Les descripteurs les plus courants sont :

- Les Descripteurs d'Appareils.
- Les Descripteurs de Configurations.
- Les Descripteurs d'Interfaces.
- Les Descripteurs de Terminaisons.
- Les Descripteurs de Chaînes.

Les appareils USB ne peuvent avoir qu'un seul descripteur d'appareil. Le descripteur d'appareil inclut des informations qui précisent la révision USB à laquelle l'appareil se soumet, les Identificateurs d'Appareils du produit et du constructeur utilisés pour charger les pilotes logiciels appropriés et le nombre possible de configurations que l'appareil peut avoir. Le nombre de configurations indique combien de ramifications de descripteurs de configurations sont appelées à suivre.

Le descripteur de configuration précise des valeurs comme la quantité de puissance qu'utilise cette configuration particulière, si l'appareil est auto-alimenté ou alimenté par le bus et le nombre d'interfaces qu'il possède. Quand un appareil est énuméré, l'hôte lit les descripteurs d'appareils et peut décider de la configuration à valider. Il peut seulement valider une configuration à la fois.

Par exemple, il est possible d'avoir une configuration d'alimentation de bus de grande puissance et une configuration auto-alimentée. Si l'appareil est branché à un hôte possédant une alimentation électrique secteur, le pilote logiciel de l'appareil choisira peut-être de permettre la configuration d'alimentation du bus de grande puissance tolérant ainsi que l'appareil soit alimentée sans être relié au secteur, cependant s'il est connecté à un laptop (ordinateur portatif) ou à un organiseur personnel, il pourra valider la seconde configuration (auto-alimentée) exigeant de l'utilisateur de brancher son appareil sur un point d'alimentation (secteur).

Les paramètres de configurations ne sont pas limités aux différences d'alimentations. Chaque configuration pourrait être alimentée de la même façon et drainer le même courant, et avoir cependant des combinaisons de terminaisons et d'interfaces différentes. Toutefois il faut tenir

compte que la modification de la configuration exige l'arrêt de toute activité sur chaque terminaison. Tandis que l'USB propose cette flexibilité, très peu d'appareils possèdent plus d'une configuration.

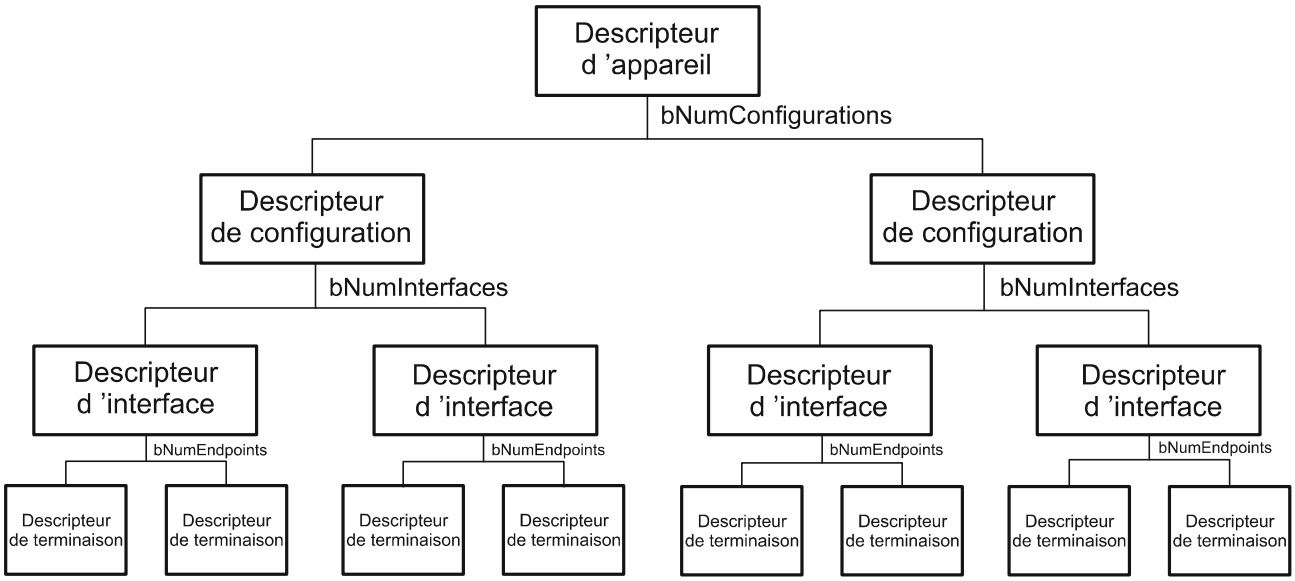


Figure II.22: Hiérarchie des différents descripteurs USB

Le descripteur d'interface peut être vu comme un "en tête" ou un regroupement de terminaison à l'intérieur d'un groupe fonctionnel accomplissant une seule fonctionnalité de l'appareil. Par exemple, vous pouvez avoir un appareil multifonctions : fax / scanner / imprimante. Le descripteur d'interface 1 pourra décrire les terminaisons de la fonction fax, le descripteur d'interface 2, la fonction scanner et le descripteur d'interface 3 la fonction imprimante. Contrairement au descripteur de configuration, il n'y a pas de limitations à avoir une seule interface validée à la fois. Un appareil peut avoir un ou plusieurs descripteurs d'interfaces validées en même temps.

Les descripteurs d'interfaces ont un champ **bInterfaceNumber** précisant le numéro de l'interface et un **bAlternateSetting** qui autorise l'interface à modifier ses paramètres au vol. Par exemple on peut avoir un appareil avec 2 interfaces, interface 1 et interface 2. Interface 1 à **bInterfaceNumber** mis à 0 indiquant qu'il est le premier descripteur d'interface et un **bAlternativeSetting** de 0.

L'interface 2 aura un **bInterfaceNumber** mis à 1 indiquant qu'il est la seconde interface et un**bAlternativeSetting** de 0 (par défaut). On pourra donc y faire entrer un autre descripteur, comprenant lui aussi un **bInterfaceNumber** mis à 1 indiquant qu'il est la seconde interface, mais cette fois le **bAlternativeSetting** mis à 1, indiquant que ce descripteur d'interface peut représenter un paramètre alternatif à celui de l'autre descripteur d'interface 2.

Quand cette configuration est validée, les 2 premiers descripteurs d'interfaces avec **bAlternativeSettings** égal à 0 sont utilisés. Toutefois pendant le fonctionnement, l'hôte peut envoyer une demande imposée SetInterface (Sélection d'Interface) à l'interface 1 avec un alternative setting (paramètre alternatif) à 1 pour valider l'autre descripteur d'interface.

Cela est plus avantageux que d'avoir 2 configurations, dans le sens où l'on peut transmettre des données via l'interface 0 tandis que l'on change les paramètres de terminaisons associés à l'interface 1 sans affecter l'interface 0.

Chaque descripteur de terminaison est utilisé pour spécifier le type de transfert, la direction, l'intervalle d'interrogation et la taille maximale de paquet pour chaque terminaison. La terminaison 0, la terminaison de commandes par défaut est toujours supposée être une terminaison de commandes et en tant que tel ne possède jamais de descripteur.

Tous les descripteurs relèvent d'un schéma commun. Le premier octet précise la longueur du descripteur, tandis que le second octet indique le type de descripteur. Si la longueur du descripteur est plus petite que ce que définit la spécification, alors l'hôte doit l'ignorer. Toutefois si la taille est plus grande que prévue, l'hôte ignorera les octets supplémentaires et ne commencera à rechercher le prochain descripteur qu'à la fin de celui-ci.

XVII.1 Le descripteur d'appareil (Device Descriptor):

Le descripteur d'appareil d'un appareil USB représente l'appareil en entier. En conséquence un appareil USB ne peut avoir qu'un seul descripteur d'appareil. Il donne des informations élémentaires et pourtant fondamentales sur l'appareil telles la version USB supporté, la taille maximale de paquet, les identificateurs du constructeur et produits et le nombre de configurations possiblesque peut avoir l'appareil.

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille du descripteur en octets (12 octets)
1	bDescriptorType	1	Constante	Descripteur d'appareil (0x01)
2	bcdUSB	2	BCD	Numéro de spécification USB auquel l'appareil doit aussi se soumettre.
4	bDeviceClass	1	Classe	Code classe (class) (assigné par USB org) Si égal à 0, chaque interface précise son propre code classe Si égal à 0xFF, le code classe est précisé par le constructeur. Autrement le champ est un code classe valable.
5	bDeviceSubClass	1	Sous Classe	Code sous classe (assigné par USB org)
6	bDeviceProtocol	1	Protocole	Code protocole (assigné par USB org)
7	bMaxPacketSize	1	Nombre	Taille maximale de paquet pour la terminaison Zéro. Les tailles conformes

				sont 8, 16, 32, 64
8	idVendor	2	ID	IDentification du fournisseur (assigné par USB org)
10	idProduct	2	ID	IDentification du produit (assigné par USB org)
12	bcdDevice	2	BCD	Numéro de version de l'appareil
14	iManufacturer	1	Index	Index de descripteur de chaîne du fabricant
15	iProduct	1	Index	Index de descripteur de chaîne du produit
16	iSerialNumber	1	Index	Index de descripteur de chaîne du numéro de série
17	bNumConfigurations	1	Entier	Nombre de configurations possible

Tableau II. 3: Format du descripteur d'appareil

Le champ **bcdUSB** rapporte la version USB la plus haute que peut supporter l'appareil. La valeur est en binaire codé décimal avec un format de 0xJJMM ou JJ est le numéro de version de poids fort, M le numéro de version de poids faible et N correspond au numéro de sous-version c'est à dire USB 2.0 est inscrit comme 0x0200, USB 1.1 comme 0x0110 et USB 1.0 comme 0x0100.

Les champs **bDeviceClass, bDeviceSubClass** et **bDeviceProtocol** sont utilisés par le système d'exploitation pour trouver un pilote logiciel de classe pour votre appareil. Habituellement seul le bdeviceclass est positionné au niveau de l'appareil. La plupart des spécifications de classe choisissent de s'identifier au niveau de l'interface et en conséquence positionnent le bdeviceclass à 0x00. Cela permet à un appareil de supporter plusieurs classes.

Le champ **bMaxPacketSize** rapporte la taille maximale pour la terminaison zéro. Tous les appareils doivent supporter la terminaison zéro.

Le champ **idVendor** et **idProduct** sont utilisés par le système d'exploitation pour trouver un pilote logiciel pour votre appareil. L'identification constructeur (vendor ID) est assignée par USB-IF.

Le champ **bcdDevice** a le même format que **bcdUSB** et est utilisé pour fournir un numéro deversion d'appareil. Cette valeur est assignée par le développeur.

Trois descripteurs de chaines de caractères existent pour fournir des détails du fabriquant, du produit et du numéro de série. Il n'y a pas d'obligation à avoir des descripteurs de chaines de caractères. Si aucun descripteur de chaines de caractères n'est présent, il faudrait utiliser un index de zéro.

Le champ **bNumConfigurations** définit le nombre de configuration que l'appareil peut supporter et savitesse normale d'exécution.

XVII.2 Le descripteur de configuration (Configuration Descriptor):

Un appareil USB peut avoir plusieurs configurations différentes alors que la majorité des appareils sont simples et n'en ont qu'une. Le Descripteur de Configuration précise la façon dont l'appareil est alimenté, quelle est sa consommation électrique maximale, le nombre d'interfaces qu'il possède. Il est donc possible d'avoir 2 configurations, quand il est alimenté par le bus et une autre quand il est alimenté par le secteur. Comme ceci est un "en tête " de descripteur d'interface, il est aussi possible d'avoir une configuration utilisant un mode de transfert différent de celui d'une autre configuration.

Une fois que toutes les configurations ont été examinées par l'hôte, celui-ci enverra une instruction **SetConfiguration** avec une valeur différente de Zéro qui correspondra à la valeur **bConfiguration** de l'une des configurations. Elle est utilisée pour sélectionner la configuration voulue.

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille du descripteur en octets (9 octets)
1	bDescriptorType	1	Constante	Descripteur de configuration (0x02)
2	wTotalLength	2	Nombre	Longueur totale en octets de données renvoyées.
4	bNumInterfaces	1	Nombre	Nombre d'interfaces
5	bConfigurationValue	1	Nombre	Valeur à utiliser comme argument pour sélectionner cette configuration
6	iConfiguration	1	Index	Index de descripteur de chaines de caractères décrivant cette configuration.
7	bmAttributes	1	Bitmap	D7 Réservé, mis à 1 (USB 1.0 alimenté par le bus) D6 Auto alimenté D5 Activation d'une station a distance D4..0 Réservé. Mis à 0
8	bMaxPower	1	mA	Consommation électrique maximale en unités de 2 mA.

Tableau II. 4: Format du descripteur de configuration

Lors de la lecture du descripteur de configuration, il renvoie la hiérarchie ou l'arborescence complète de configuration qui inclut toute interface apparentée et les descripteurs de terminaisons. Le champ **wTotalLength** indique le nombre d'octets dans la hiérarchie.

Le champ **bNumInterfaces** indique le nombre d'interface présent pour cette configuration.

Le champ **bConfigurationValue** est utilisé par la demande SetConfiguration pour sélectionner cette configuration.

Le champ **iConfiguration** est un index de descripteur de chaine de caractère décrivant la configuration dans un format lisible par l'homme.

Le champ **bmAttributes** précise les paramètres d'alimentation pour la configuration. Si un appareil est auto-alimenté, il positionne D6. Le bit D7 était autrefois utilisé par l'USB 1.0 pour indiquer un appareil alimenté par le bus, ceci est maintenant réalisé par le champ **bMaxPower**. Si un appareil utilise l'énergie du bus, que ce soit un appareil alimenté par le bus ou un appareil auto-alimenté, il doit rapporter sa consommation électrique dans le champ **bMaxPower**. Les appareils peuvent aussi prendre en charge l'activation d'une station à distance qui permettra à l'appareil de réveiller l'hôte quand celui-ci est en mode veille.

Le champ **bMaxPower** définit la consommation électrique maximale que l'appareil peut prendre du bus. Elle est donnée en unités de 2 mA, jusqu'au chiffre maximum de 500 mA environ. La spécification permet à un appareil alimenté par un bus de forte puissance de ponctionner jusqu'à 500 mA à partir de Vbus. Dans le cas où un appareil perd son alimentation externe, il ne doit pas ponctionner plus que ce qui est indiqué dans **bMaxPower**. Il devrait échouer sur toutes opérations nécessitant l'alimentation externe.

XVII.3 Le descripteur d'interface (Interface Descriptor):

Le Descripteur d'Interface peut être vu comme un " en tête " ou un regroupement de Terminaisons dans un groupe fonctionnel exécutant une simple fonction pour l'appareil.

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille du descripteur en octets (9 octets)
1	bDescriptorType	1	Constante	Descripteur de configuration (0x04)
2	bInterfaceNumber	1	Nombre	Numéro d'interface
3	bAlternateSetting	1	Nombre	Valeur utilisée pour sélectionner une configuration de remplacement
4	bNumEndpoints	1	Nombre	Nombre de terminaisons utilisées pour cette interface.
5	bInterfaceClass	1	Class	Code Classe (assigné par USB org)
6	bInterfaceSubClass	1	SubClass	Code Sous Classe (assigné par USB org)
7	bInterfaceProtocol	1	Protocol	Code du Protocole (assigné par USB org)
8	iInterface	1	Index	Index du descripteur de chaine décrivant cette interface.

Tableau II. 5: Format du descripteur d'interface

Le champ **bInterfaceNumber** indique l'index du descripteur d'interface. Il devait être pointé à Zéro, et incrémenté une fois pour chaque nouveau descripteur d'interface.

Le champ **bAlternativeSetting** peut être utilisé pour préciser les interfaces de remplacement. Ces interfaces alternatives peuvent être sélectionnées par la demande SetInterface.

Le champ**bNumEndpoints** indique le nombre de terminaisons utilisé par l'interface. Cette valeur devrait exclure la terminaison Zéro et est utilisée pour indiquer le nombre de Descripteurs de terminaisons à suivre.

Les champs **bInterfaceClass, bInterfaceSubClass** et **bInterfaceProtocol** peuvent être utilisés pour préciser les classes prises en compte (par exemple : HID, Communications, mémoire de masse etc…).

Ceci permet à plusieurs appareils d'utiliser des " drivers " (pilote logiciel) de classe évitant le besoin d'écrire des " drivers " spécifiques pour votre appareil.

Le champ **iInterface** permet d'avoir une description textuelle de l'interface.

XVII.4 Le descripteur de terminaison (Endpoint Descriptor):

Les Descripteurs de Terminaison sont utilisés pour décrire les terminaisons autres que la terminaison zéro. La terminaison zéro est toujours censée être une terminaison de commande et est configuré avant que n'importe quel autre descripteur ne soit sollicité. L'Hôte utilisera l'information renvoyée par ces descripteurs pour déterminer les besoins de bande passante du bus.

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille du descripteur en octets (7 octets)
1	bDescriptorType	1	Constante	Descripteur de terminaison (0x05)
2	bEndpointAdress	1	Terminaison	Adresse de terminaison Bits 0..3b Numéro de terminaison Bits 4..6b Réservés. Mis à 0. Bit 7 Direction ; 0 = Sortie, 1 = Entrée (ignoré pour les terminaisons de commande).
3	bmAttributes	1	Bitmap	Bits 0..1 Type de transfert 00 = Commande 01 = Isochrone 10 = par Bloc 11 = Interruption Bits 2..7 sont réservés Si terminaison Isochrone : Bits 3..2 = Type de synchronisation (mode Iso) 00 = Pas de synchronisation 01 = Asynchrone 10 = Adaptif

				11 = Synchrone Bits 5..6 = Type d'utilisation (mode iso) 00 = Terminaison de données 01 = Terminaison de retour (Feedback) 10 = Renvoi explicite de terminaison de données 11 = Réservé
4	wMaxPacketSize	2	Nombre	Taille maximale du paquet que cetteterminaison est capable d'envoyer ou derecevoir
6	bIntervale	1	Nombre	Intervalle de temps pour interroger lestransferts de données de la terminaison. Valeur en nombre de trames. Ignoré pour lesterminaisons par Bloc et de commande. Pourle mode Isochrone il doit être égal à 1 et lechamp peut valoir de 1 à 255 pour desterminaisons d'interruptions.

Tableau II. 6: Format du descripteur de terminaison

Le champ **bEndpointAddress** indique quelle terminaison ce descripteur décrit.

Le champ**bmAttributes** précise le type de transfert. Cela peut être soit des transferts de type Commande, Interruption, Isochrone ou par Blocs. Si une terminaison Isochrone est précisée, des attributs supplémentaires peuvent être sélectionnés tel que la synchronisation et les types d'utilisations.

Le champ **wMaxPacketSize** indique la taille maximale de charge utile pour cette terminaison.

Le champ **bInterval** est utilisé pour préciser cet intervalle d'interrogation de certains transferts. L'unité est exprimé en trames équivalent ainsi à 1 ms pour des appareils basse / pleine vitesse et 125 µs pour des appareils haute vitesse.

XVII.5 Le descripteur de chaine de caractères (String Descriptor):

Les Descripteurs de chaînes fournissent une information lisible pour l'homme et sont optionnels. S'ils ne sont pas utilisés, tout champ d'index de descripteurs de chaînes doit être mis à zéro indiquant qu'il n'y a pas de descripteur de chaîne disponible.

Les chaînes de caractères sont codées au format Unicode et les produits peuvent être prévuspour comprendre les diverses langues. L'index de chaîne zéro devra retourner une liste delangues acceptée. On peut trouver une liste USB d'identification de langue dans Universal SerialBus Language Identifiers (LANGIDs) version 1.0 (Identificateurs de langue sur BUS SérieUniversel version 1.0).

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille du descripteur en octets
1	bDescriptorType	1	Constante	Descripteur de chaîne (0x03)
2	wLANGID[0]	2	Nombre	Langue acceptée code zéro(par exemple 0x0409 Anglais, U.S.)
4	wLANGID[1]	2	Nombre	Langue acceptée code Un(par exemple 0x0C09 Anglais, Australien)
n	wLANGID[n]	2	Nombre	Langue acceptée code x(par exemple 0x0407 Allemand - Standard)

Tableau II. 7: Format du descripteur de chaîne Zéro

Le descripteur de chaînes ci-dessus montre le format du descripteur de chaîne zéro. L'Hôte devralire ce descripteur pour déterminer quelles langues sont disponibles. Si une langue est acceptée,elle peut être référencée en envoyant l'Identification de la langue dans le champ **wIndex** à lademande de Get Descriptor(String).

Toutes les chaînes de caractères à venir tiennent dans le format ci-dessous.

Décalage	Champ	Taille	Valeur	Description
0	bLength	1	Nombre	Taille du descripteur en octets
1	bDescriptorType	1	Constante	Descripteur de chaîne (0x03)
2	bString	n	Unicode	Textes codés unicode

Tableau II. 8: Format du descripteur de chaîne

XVIII LE PAQUET D'INSTALLATION:

Tout appareil USB doit répondre aux paquets d'installation sur le canal de communication par défaut. Les paquets d'installation sont utilisés pour la détection et configuration de l'appareil et véhiculent des fonctions courantes telles que la mise en place de l'adresse de l'appareil USB, la demande d'un descripteur d'appareil ou la vérification de l'état d'une terminaison.

Un hôte USB conforme s'attend à ce que toutes les requêtes soient traitées dans une périodemaximale de 5 secondes. Il précise aussi des temps plus stricts pour des requêtes particulières.

Les requêtes standards d'appareils sans étage de données doivent être accomplies en 50 ms.

Les requêtes standards d'appareils avec un étage de données doivent commencer à renvoyer lesdonnées 500 ms après la requête.

- Chaque paquet de données doit être envoyé dans les 500 ms de la transmission réussie du paquet précédent.
- L'étage d'état doit être accompli dans les 50 ms après la transmission du dernier paquet de données.

L'instruction **SetAdress** (qui contient une phase de données) doit traiter l'instruction et retourner l'état dans ces 50 ms. L'appareil a donc 2 ms pour changer d'adresse avant que la prochaine requête ne soit envoyée.

Chaque requête commence avec un paquet d'installation de 8 octets qui a le format suivant :

Décalage	Champ	Taille	Valeur	Description
0	bmRequestType	1	Bitmap	D7 Direction de transfert dephase de données : 0 = Hôte vers l'appareil 1 = Appareil vers hôte D6..5 Type : 00 = Standard 01 = Classe 10 = Constructeur 11 = Réservé D4..0 Destinataire 0 = Appareil 1 = Interface 2 = Terminaison 3 = Autre 4 ..31 Réservés
1	bRequest	1	Valeur	Requête
2	wValue	2	Valeur	Valeur
4	wIndex	2	Index ou Décalage	Index
6	wLength	2	Compteur	Nombres d'octets à transférer s'ily a une phase de données

Tableau II. 9: Format du paquet d'installation

Le champ **bmRequestType** déterminera le sens de la requête, le type de la requête et le destinataire désigné. Le champ **bRequest** indique la requête formulée. Le champ **bmRequestType** est généralement analysé et l'exécution est raccordée à un numéro d'identificateurs comme un gestionnaire de requête d'un appareil standard, un gestionnaire de requête d'une interface standard, un gestionnaire de requête de terminaison standard, un gestionnaire de requête d'un appareil de

classe, etc…. La façon d'exécuter le paquet d'installation vous appartient. D'autres pourraient choisir d'exécuter en premier **bRequest** puis de déterminer le type et le destinataire basé sur chaque requête.

Les requêtes standards sont communes à tous les appareils USB. Les requêtes de classe sont communes aux classes de drivers (pilote logiciel).Par exemple, tout appareil étant conforme à la classe HID aura un ensemble commun derequêtes spécifiques de classe. Ils diffèreront d'un appareil conforme à la classe communicationet diffèreront encore d'un appareil conforme à la mémoire de masse (de grande capacité destockage).

Et pour finir, les requêtes définies par le constructeur. Ce sont des requêtes que vous pouvezattribuer en tant que concepteur d'appareil USB. Elles sont normalement différentes d'unappareil à un autre, mais dépendent de votre réalisation et de votre imagination.

Une requête commune peut être dirigé vers des destinataires différents et basée sur lesfonctions différentes exécutée par le destinataire. Par exemple une requête standard **GetStatus**peut être dirigée sur l'appareil, l'interface ou la terminaison. Quand elle est dirigée sur unappareil, cette dernière retourne des drapeaux (flags) indiquant l'état de la station d'activation et sil'appareil est auto-alimenté.

Toutefois si la même requête est dirigée sur l'interface il renvoie toujours Zéro, ou bien si elleétait dirigée sur une terminaison, l'appareil retournerait le drapeau Halt pour la terminaison.

Les champs **wValue** et **wIndex** permettent le passage de paramètres avec la requête. Le champ **wLength** est utilisé pour préciser le nombre d'octets à transférer au cas où il existerait unephase de donnée.

La section 9.4 de la spécification USB détaille les requêtes " d'appareils standard " exigées quidoivent être appliquées pour chaque appareil USB. Le standard fournit un tableau uniqueregroupant des articles par requête. Considérant que la plupart des microprogrammesanalyseront le paquet d'installation par destinataire, nous choisirons de séparer les requêtes enfonction du destinataire pour une étude et une application plus simple.

XVIII.1 Les requêtes d'appareil standard:

Il y a actuellement 8 requêtes d'appareil standard, chacune étant détaillée dans le **tableau II.10.**

bmRequestType	bRequest	wValue	wIndex	wLength	Données
1000 0000b	GET_STATUS (0x00)	Zéro	Zéro	Deux	Etat de l'appareil
0000 0000b	CLEAR_FEATURE(0x01)	Sélecteur defonction	Zéro	Zéro	Aucune
0000 0000b	SET_FEATURE (0x03)	Sélecteur defonction	Zéro	Zéro	Aucune
0000 0000b	SET_ADDRESS (0x05)	Adresse de l'appareil	Zéro	Zéro	Aucune
1000 0000b	GET_DESCRIPTOR(0x06)	Type de Descripteur & Index	Zéro ou ID de Langues	Longueur du Descripteur	Descripteur
0000 0000b	SET_DESCRIPTOR (0x07)	Type de Descripteur & Index	Zéro ou ID de Langues	Longueur du Descripteur	Descripteur
1000 0000b	GET_CONFIGURATION (0x08)	Zéro	Zéro	1	Valeur de Configuration
0000 0000b	SET_CONFIGURATION (0x09)	Valeur de Configuration	Zéro	Zéro	Aucune

Tableau II. 10: Requêtes d'appareils standards

La requête **GetStatus** dirigée vers l'appareil retournera 2 octets pendant l'étage de donnéessuivant le format :

Si D0 est à 1, ceci indique que l'appareil est auto alimenté. S'il est à 0, l'appareil est alimenté par le bus. Si D1 est à 1, l'activation à distance de l'appareil est validée et ce dernier peut réveiller l'hôte pendant le mode veille. Le bit d'activation à distance peut être positionné par les requêtes **SetFeature** et **ClearFeature.**

Set Address est utilisé pendant l'énumération pour attribuer une adresse unique à l'appareil USB. L'adresse est précisée dans **wValue** et peut valoir au maximum 127. Cette requête est unique dans le sens ou l'appareil ne positionnera pas son adresse tant que la phase d'état ne sera pas achevée. Toutes les autres requêtes doivent être terminées avant la phase d'état.

Set Descriptor/Get Descriptor est utilisé pour renvoyer le descripteur indiqué dans **wValue**. Une requête pour le descripteur de configuration retournera le descripteur d'appareil et, tous les descripteurs d'interfaces et de terminaisons dans la même requête.

- Les descripteurs de terminaisons ne peuvent pas être accessibles directement par une requête de **GetDescriptor / SetDescriptor**.
- Les descripteurs d'interfaces ne peuvent pas être accessibles directement par une requête de **GetDescriptor / SetDescriptor**.
- Les descripteurs de chaînes incluent une Identification de langues dans **wIndex** pour autoriser le support de plusieurs langues.

GetConfiguration/SetConfiguration est utilisé pour demander ou positionner la configuration de l'appareil actuel. Dans le cas d'une requête **GetConfiguration**, un octet sera renvoyé pendant la phase de donnée indiquant l'état de l'appareil. Une valeur zéro signifie que l'appareil n'est pas configuré et une valeur différente de zéro indique que l'appareil est configuré. **SetConfiguration** est utilisé pour valider un appareil. Il doit contenir la valeur de **bConfigurationValue** du descripteur de configuration voulu dans l'octet de poids faible de **wValue** pour sélectionner quelle configuration valider.

XVIII.2 Les requêtes d'interface standard:

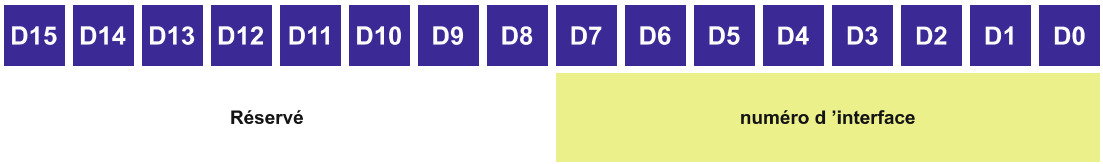
La spécification actuelle définit 5 requêtes d'interface standard qui sont détaillées dans **tableau II.11.** Il est intéressant de noter que, seules 2 requêtes donnent quelque chose de compréhensible.

bmRequestType	bRequest	wValue	wIndex	wLength	Données
1000 0000b	GET_STATUS (0x00)	Zéro	Interface	Deux	Etat de l'appareil
0000 0001b	CLEAR_FEATURE(0x01)	Sélecteur defonction	Interface	Zéro	Aucune
0000 0001b	SET_FEATURE (0x03)	Sélecteur defonction	Interface	Zéro	Aucune

1000 0001b	GET_INTERFACE (0x0A)	Zéro	Interface	Un	Interface de remplacement
0000 0001b	SET_INTERFACE (0x11)	Positionnement alternatif	Interface	Zéro	Aucune

Tableau II. 11: Requêtes d'interfaces standards

Le champ**wIndex** est normalement utilisé pour préciser l'interface de référence pour des requêtes liées à l'interface. Voir son format dans le schéma ci-dessous.



GetStatus est utilisé pour retourner l'état de l'interface. Une telle requête à l'interface doit renvoyer 2 octets de valeur 0x00, 0x00. (Les 2 octets sont réservés pour une utilisation future).

Les requêtes **ClearFeature** et **SetFeature** peuvent être utilisées pour positionner des fonctions booléennes. Quand le destinataire désigné est l'interface, la spécification USB actuelle révision 2 précise qu'il n'y a pas de fonctions d'interface.

GetInterface et **SetInterface** règle le positionnement de l'interface de remplacement.

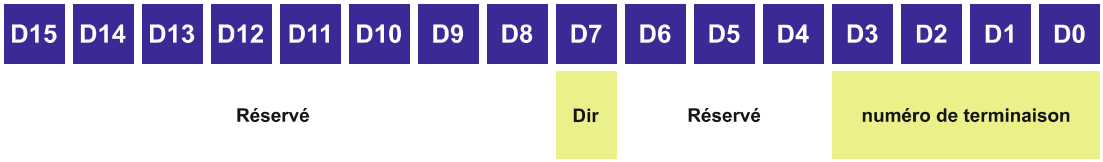
XVIII.3 Les requêtes de terminaison standard:

Les requêtes de terminaisons standard sont au nombre de 4, listées dans le **tableau II.12**.

bmRequestType	bRequest	wValue	wIndex	wLength	Données
1000 0010b	GET_STATUS (0x00)	Zéro	Terminaison	Deux	Etat de l'appareil
0000 0010b	CLEAR_FEATURE(0x01)	Sélecteur defonction	Terminaison	Zéro	Aucune
0000 0010b	SET_FEATURE (0x03)	Sélecteur defonction	Terminaison	Zéro	Aucune
1000 0010b	SYNCH_FRAME (0x12)	Zéro	Terminaison	Deux	Numéro de Trame

Tableau II. 12: Requêtes de terminaisons standards

Le champ **wIndex** est normalement utilisé pour préciser la terminaison de référence et la direction pour les requêtes liées à la terminaison. Voir son format dans le schéma ci-dessous.



GetStatus renvoie 2 octets indiquant l'état (Arrêté/Bloqué) d'une terminaison. Le format des 2 octets renvoyés est illustré ci-dessous.

ClearFeature et **SetFeature** sont utilisés pour positionner les fonctions de la terminaison. Le standard défini actuellement un sélecteur de fonction de terminaison : ENDPOINT_HALT (0x00) qui permet à l'hôte de bloquer et d'effacer une terminaison. Seules les terminaisons autres que la terminaison par défaut sont conseillées pour avoir cette fonctionnalité.

Une requête **SynchFrame** est utilisée pour rapporter une trame de synchronisation determinaison.

XIX L'ENUMERATION:

L'énumération est la manière de déterminer l'appareil qui vient juste d'être branché au bus et les paramètres dont il a besoin, comme la consommation électrique, le nombre et le type de terminaison, la classe du produit, etc.… L'hôte attribuera donc à l'appareil une adresse et validera une configuration lui permettant de transférer des données sur le bus. Toutefois lorsque l'on écrit un microprogramme USB pour la première fois, il est plus pratique de connaître la manière dont l'hôte répond pendant l'énumération, plutôt que le processus d'énumération général détaillé dans la spécification.

Une énumération sous Windows ordinaire implique les étapes suivantes :

- L'hôte ou Hub détecte la connexion d'un nouvel appareil via les résistances de rappel de l'appareil reliées sur les 2 fils de données. L'hôte attend au moins 100ms, le temps que la prise soit insérée complètement et que l'alimentation de l'appareil soit stabilisée.
- L'hôte émet un " reset " mettant l'appareil dans l'état par défaut. L'appareil peut maintenant répondre à l'adresse zéro par défaut.
- L'hôte (sous MS Windows) demande les 64 premiers octets du descripteur d'appareil.
- Après avoir reçu les 8 premiers octets du descripteur d'appareil, l'hôte émet immédiatement un autre reset sur le bus.
- L'hôte émet maintenant une commande **SetAdress**, mettant l'appareil dans l'état adressable.
- L'hôte demande la totalité des 18 octets du descripteur d'appareil.
- Puis il demande les 9 octets du descripteur de configuration pour déterminer la taille totale.
- L'hôte demande les 255 octets du descripteur de configuration.
- L'hôte demande l'un des descripteurs de chaînes s'ils étaient indiqués.

A la fin de l'étape 9, " Windows " demandera un driver (pilote logiciel) pour votre appareil. Il est alors courant de le voir redemander tous les descripteurs avant d'émettre une requête**SetConfiguration.**

L'étape 4 embarrasse souvent les gens qui écrivent des microprogrammes pour la première fois. L'hôte demande les 64 premiers octets du descripteur d'appareil, aussi lorsque l'hôte met à zéro votre appareil après avoir reçu les 8 premiers octets, il est tout à fait naturel de penser qu'il y a un problème soit au niveau du descripteur d'appareil soit dans la façon dont votre microprogramme manipule la requête. Cependant, comme vous le diront beaucoup de gens, si vous insistez sur la mise en œuvre de la commande **SetAdress**, elle sera récompensée par la demande suivante de 18 octets pleins du descripteur d'appareil.

Généralement quand il y a un problème avec le descripteur ou sur la façon dont il est envoyé, l'hôte tentera de le lire 3 fois avec de longues pauses entre les requêtes. Après la troisième tentative, l'hôte abandonne signalant une erreur au niveau de l'appareil.

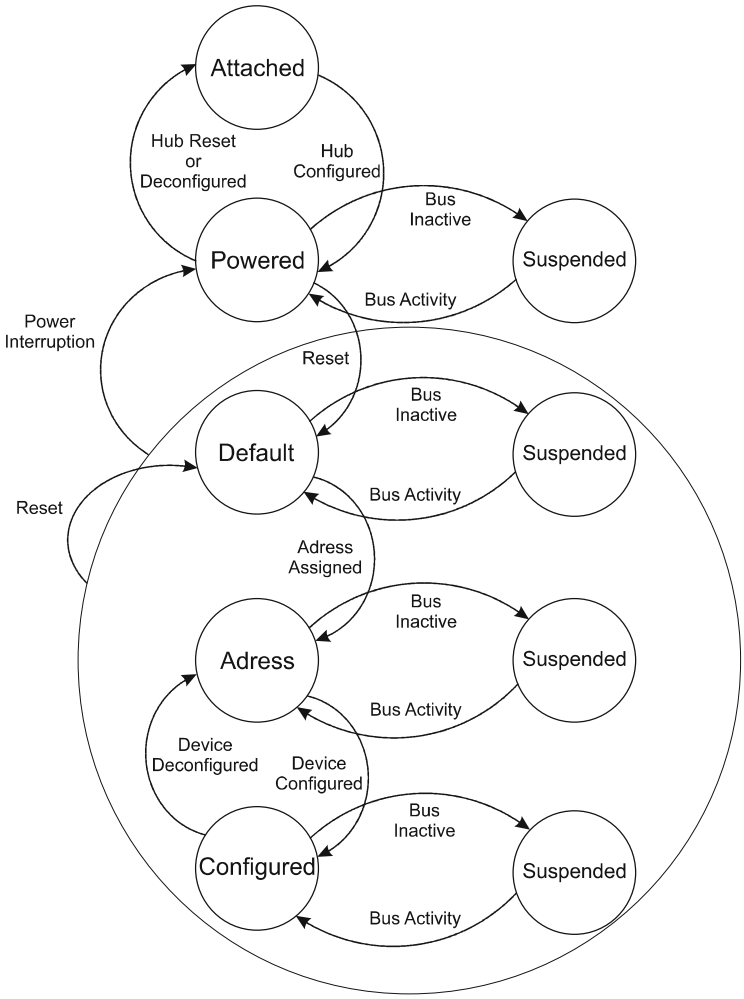


Figure II.23: Diagramme d'état d'un appareil USB

Chapitre 3 :MICROCONTROLEUR MICROCHIP

I Etude pratique:

Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un *ordinateur* : processeur, mémoires (mémoire morte pour le programme, mémoire vive pour les données), unités périphériques et interfaces d'*entrées-sorties*.

Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique (quelques milliwatts en fonctionnement, quelques nanowatts en veille), une vitesse de fonctionnement plus faible (quelques mégahertz à quelques centaines de mégahertz) et un coût réduit par rapport aux *microprocesseurs* polyvalents utilisés dans les *ordinateurs personnels*.

Par rapport à des systèmes électroniques à base de *microprocesseurs* et autres composants séparés, les microcontrôleurs permettent de diminuer la taille, la consommation électrique et le coût des produits. Ils ont ainsi permis de démocratiser l'utilisation de l'informatique dans un grand nombre de produits et de procédés.

II Du microprocesseur au microcontrôleur :

Le processeur est l'élément central d'un système informatique : il interprète les instructions et traite les données d'un programme. Il a besoin de certains éléments externes pour fonctionner :

- une horloge pour le cadencer (en général à quartz ou Boucle à verrouillage de phase (PLL : Phase-Locked Loop) ;
- de la mémoire pour stocker les variables durant l'exécution du programme (mémoire vive RAM) et le programme d'une mise sous tension à l'autre (mémoire morte ROM).
- des périphériques (pour interagir avec le monde

extérieur). Ces éléments sont reliés par 3 bus :

- le *bus d'adresse* qui permet au microprocesseur de sélectionner la case mémoire ou le périphérique auquel il veut accéder pour lire ou écrire une information (instruction ou donnée).
- le *bus de données* qui permet le transfert des informations entre les différents éléments, ces informations seront soit des instructions, soit des données en Provenance ou à destination de la mémoire ou des périphériques.
- le *bus de contrôle* qui indique si l'opération en cours est une lecture ou une écriture, si un périphérique demande une *interruption* pour faire remonter une information au

processeur, etc.

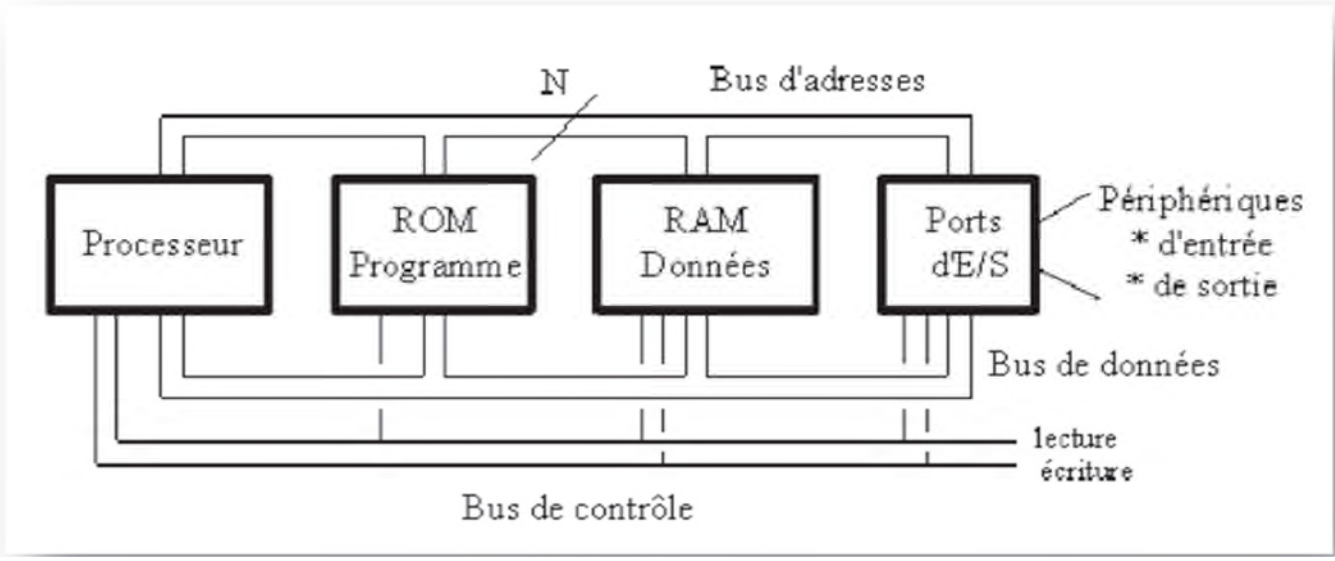


Figure III. 1:Structure d'un système à microprocesseur

Les microcontrôleurs améliorent l'intégration et le coût (lié à la conception et à la réalisation) d'un système à base de *microprocesseur* en rassemblant ces éléments essentiels dans un seul *circuit intégré*.

Il s'agit de mémoire morte, de mémoire vive, de portes d'entrée – sortie parallèles, de ports pour la communication par ligne série, de logique de gestion du temps et des événements et d'interfaces spécialisée, par exemple des convertisseurs A /N.

Un microcontrôleur est donc un composant autonome, capable d'exécuter le programme contenu dans sa mémoire morte dès qu'il est mis sous tension. Selon les modèles et les conditions de fonctionnement, les microcontrôleurs peuvent avoir besoin de quelques composants externes (quartz, quelques condensateurs, parfois une ROM), mais ceci reste très limité.

II.1 Architecture d'un microcontrôleur :

Les microcontrôleurs, quel que soit leurs constructeurs, ont des architecture très similaires et sont constitués de modules fondamentaux assurant les mêmes fonctions : UAL, Ports d'E/S, interfaces de communications série, Interfaces d'E/S analogiques, Timers et horloge temps réels …On peut dire que seul le langage de programmation (Assembleurs) constitue la différence majeure en deux microcontrôleur (similaires) venant de deux constructeurs différents.

Un microcontrôleur peut comporter :

- un *processeur (CPU)*, avec une largeur du chemin de données allant de 4 bits pour les modèles les plus basiques à 32 ou 64 bits pour les modèles les plus évolués.
- de la *mémoire vive (RAM)* pour stocker les données et variables.
- de la *mémoire morte (ROM)* pour stocker le programme. Différentes technologies peuvent être employées : EPROM, EEPROM, mémoire flash (la plus récente).
- souvent un *oscillateur* pour le cadencement. Il peut être réalisé avec un quartz, un circuit RC ou encore une PLL .
- des périphériques, capables d'effectuer des tâches spécifiques. On peut mentionner entre autres :
    - les *convertisseurs* analogiques-numériques (CAN) (donnent un nombre binaire à partir d'une tension électrique).
    - les *convertisseurs* numériques-analogiques (CNA) (effectuent l'opération inverse).
    - les générateurs de signaux à modulation de largeur d'impulsion (MLI, ou en anglais, PWM pour Pulse Width Modulation),
    - les timers/compteurs (compteurs d'impulsions d'horloge interne ou d'événements externes).164
    - les chiens de garde (watchdog).
    - les comparateurs (comparent deux tensions électriques).
    - les contrôleurs de bus de communication (USART, I2C,SSP, CAN, FlexRay, USB,Ethernet, etc.).

Figure III. 2 : Microcontrôleur pic18F2550

Le fonctionnement des périphériques peut être paramétré et commandé par le programme et/ou les entrées-sorties. Les périphériques peuvent générer une interruption qui, contraint le processeur à quitter le programme en cours pour effectuer une routine de traitement de l'interruption, lorsque l'événement qui la déclenche survient.

II.1.1 L'unité centrale ou CPU :

Le processeur, ou CPU (de l'anglais Central Processing Unit, « Unité centrale de traitement »), est le composant de l'ordinateur qui exécute les programmes informatiques , donc il est responsable d'exécuter les instructions qui sont dans la mémoire de programme.

Un processeur construit en un seul circuit intégré est un microprocesseur.

II.1.2 Composition d'un processeur :

Les parties essentielles d'un processeur sont :

- l'Unité Arithmétique et Logique (UAL, en anglais Arithmetic and Logical Unit - ALU), qui prend en charge les calculs arithmétiques élémentaires et les tests et les instructions logiques.
- l'unité de contrôle ou séquenceur, qui permet de synchroniser les différents éléments du processeur. En particulier, il initialise les registres lors du démarrage de la machine et il gère les interruptions.
- Les registres, qui sont des mémoires de petite taille (quelques octets), suffisamment rapides pour que l'UAL puisse manipuler leur contenu à chaque cycle de l'horloge.

Un certain nombre de registres sont communs à la plupart des processeurs :

- Compteur ordinal : ce registre contient l'adresse mémoire de l'instruction en cours d'exécution ;
- accumulateur : ce registre est utilisé pour stocker les données en cours de traitement par l'UAL ;
- registre d'adresses : il contient toujours l'adresse de la prochaine information à lire par l'UAL, soit la suite de l'instruction en cours, soit la prochaine instruction ;
- registre d'instructions : il contient l'instruction en cours de traitement ;
- registre d'état : il sert à stocker le contexte du processeur, ce qui veut dire que les différents bits de ce registre sont des drapeaux (flags) servant à stocker des informations concernant le résultat de la dernière instruction exécutée ;
- pointeurs de pile : ce type de registre, dont le nombre varie en fonction du type de processeur, contient l'adresse du sommet de la pile (ou des piles) ;
- registres généraux : ces registres sont disponibles pour les calculs ;

- l'horloge qui synchronise toutes les actions de l'unité centrale. Elle est présente dans les processeurs synchrones, et absente des processeurs asynchrones et des processeurs auto synchrones ;
- l'unité d'entrée-sortie, qui prend en charge la communication avec la mémoire de l'ordinateur ou la transmission des ordres destinés à piloter ses processeurs spécialisés, permettant au processeur d'accéder aux périphériques de l'ordinateur.

II.2 Modèle de type (CISC) :

L'architecture CISC (Complexe Instruction Set Computer, soit « Ordinateur à jeu d'instruction complexe ») consiste à câbler dans le processeur des instructions complexes, difficiles à créer à partir des instructions de base.

Les instructions sont de longueurs variables et peuvent parfois nécessiter plus d'un cycle d'horloge.

Or, un processeur basé sur l'architecture CISC ne peut traiter qu'une instruction à la fois, d'où un temps d'exécution conséquent.

II.3 Principe de fonctionnement de l'architecture HAWARD (RISC) :

Un seul cycle d'horloge par instruction :

L'opérande est intégrée à l'instruction.

Exemple MOVLW 10 ; Charger la constante 10 dans le registre de travail W. [To Move : déplacer, L (littéral) : constante, W : work (registre de travail)] On trouvera en mémoire :

Cette instruction se traduira par le code suivant en mémoire 1100xx 00001010

MOVLW 10

Pour un PIC16 (architecture RISC), cette instruction est codée sur 14 bits (instruction + opérande).Les mémoires programmes et données et les bus correspondants sont séparés :Ce microcontrôleur est basé sur une architecture de type Harvard, c'est-à-dire qu'il y a séparation des bus d'instruction et de données ainsi que de l'espace d'adressage.

Ceci permet au même instant :

> D'exécuter l'instruction correspondant à l'adresse courante,

> D'extraire l'instruction suivante.

III PIC18F2550 :

Un **PIC** n'est rien d'autre qu'un microcontrôleur, c'est-à-dire une unité de traitement de l'information de type microprocesseur à laquelle on a ajouté des périphériques internes permettant de réaliser des montages sans nécessiter l'ajoute de composants externes.

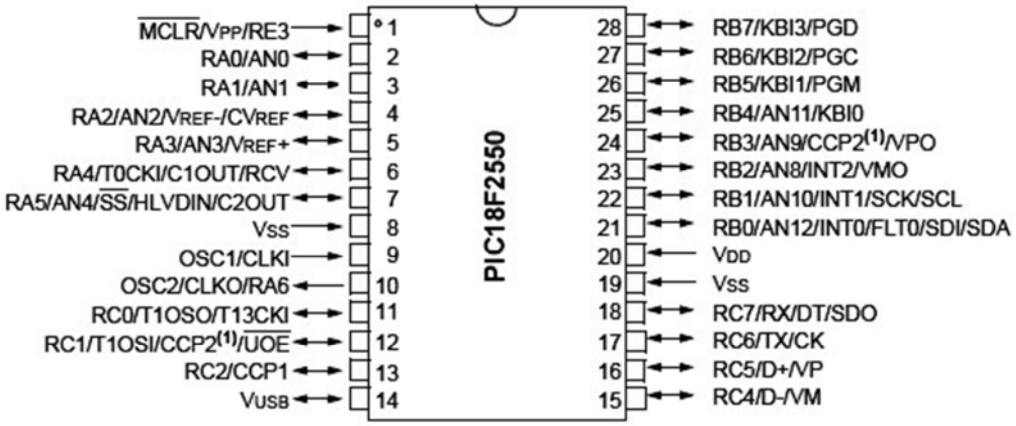


Figure III. 3: Brochage du PIC18F2550

III.1 Modèle de type Harward (RISC) :

Un processeur utilisant la technologie RISC (Reduced Instruction Set Chip, soit « Ordinateur à jeu d'instructions réduit ») n'a pas de fonctions évoluées câblées.

Les programmes doivent ainsi être traduits en instructions simples, ce qui entraîne un développement plus difficile et/ou un compilateur plus puissant. Une telle architecture possède un coût de fabrication réduit par rapport aux processeurs CISC. De plus, les instructions, simples par nature, sont exécutées en un seul cycle d'horloge, ce qui rend l'exécution des programmes plus rapide qu'avec des processeurs basés sur une architecture

CISC. Enfin, de tels processeurs sont capables de traiter plusieurs instructions simultanément en les traitants en parallèle.

IV Caractéristiques du PIC 18F2550

Le 18F2550 est un microcontrôleur en boîtier DIL 28 pattes possédant jusqu'à 24 E/S sur 3 ports (A, B, C, et éventuellement RE3).

Alimentable de 2 à 6 V continu, il est compatible avec le mode de programmation ICSP (In Chip Serial Programming), et dispose d'un oscillateur interne pouvant monter jusqu'à 8 Mhz. Sur oscillateur externe, le 18F2550 peut monter jusqu'à 48 MHz.

Au niveau mémoire, ce PIC dispose de 16 KO de mémoire flash pour le fireware, de 2 KO de RAM et de 256 octets d'E²PROM.

Côté fonctionnalité, outre les E/S numériques classiques, ce PIC dispose de 10 CAN, de 2 comparateurs, de 2 PWM 10 bits, d'1 module de communication série synchrone et asynchrone, d'un module de communication USB, de 4 timers, ...

Il faut savoir qu'il existe le 18F4550. Ce PIC, pouvant être considéré comme une extension du 18F2550, est en boitier DIL 40 pattes, et possède jusqu'à 35 E/S. Possédant également un module de communication USB, leur principale différence, outre le nombre de pattes, et donc d'E/S se situe au niveau des CAN. En effet, le 18F4550 en possède 13, soit 3 de plus que le 18F2550.

IV.1 Les entrées/sorties

Dans cette partie, nous allons voir les différentes possibilités de chaque patte, en les désignant par leur numéro. A noter, qu'une seule fonction est disponible à la fois, par patte.

PATTE 1 : E numérique RE3, reset.

PATTE 2 : E/S numérique RA0, CAN0.

PATTE 3 : E/S numérique RA1, CAN1.

PATTE 4 : E/S numérique RA2, CAN2, tension de référence du comparateur, référence basse CAN.

PATTE 5 : E/S numérique RA3, CAN3, référence haute CAN.

PATTE 6 : E/S numérique RA4, entrée comptage timer 0, sortie comparateur 1, E module USB externe.

PATTE 7 : E/S numérique RA5, CAN4, sélection de périphérique synchrone, sortie de comparateur 2.

PATTE 8 : Masse.

PATTE 9 : Entrée oscillateur, entrée horloge.

PATTE 10 : Entrée oscillateur, sortie horloge, S numérique RA6.

PATTE 11 : E/S numérique RC0, sortie horloge timer 1.

PATTE 12 : E/S numérique RC1, entrée horloge timer 1, sortie PWM 2, S module USB externe.

PATTE 13 : E/S numérique RC2, sortie PWM 1.

PATTE 14 : Référence de tension USB 3,3V (condensateur de 470 nF).

PATTE 15 : E/S numérique RC4, E/S – USB, E module USB externe VM PATTE 16: E/S numérique RC5, E/S + USB, E module USB externe VP.

PATTE 17: E/S numérique RC6, S RS232, S horologe synchronic.

PATTE 18 : E/S numérique RC7, E RS232, E donnée synchrone, sortie donnée SPI.

PATTE 19 : Masse.

PATTE 20 : Alimentation positive +5V.

PATTE 21 : E/S numérique RB0, CAN12, interruption externe 0, entrée d'erreur PWM, entrée SPI, E/S I²C.

PATTE 22 : E/S numérique RB1, CAN10, interruption externe 1, horloge SPI, horloge I²C.

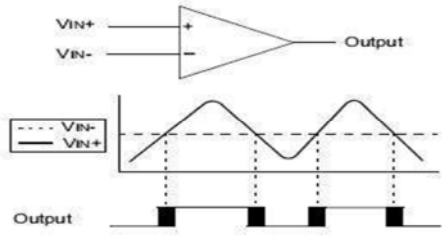
PATTE 23 : E/S numérique RB2, CAN8, interruption externe 2, S module USB externe VM.

PATTE 24: E/S numérique RB3, CAN9, S PWM 2, S module USB extern VP.

PATTE 25 : E/S numérique RB4, CAN11, interruption de changement 0.

PATTE 26 : E/S numérique RB5, interruption de changement 1.

PATTE 27 : E/S numérique RB6, interruption de changement 2.



PATTE 28 : E/S numérique RB7, interruption de changement 3.

Les E/S se configurent via TRISA, TRISB, TRISC, TRISE.

IV.2 La tension de référence

La tension de référence est configurable via CVRCON. Toutefois cette source est dédiée au comparateur (d'où le C de CVREF).

CVRCON

CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0

CVREN : active (1) ou non la référence de tension

CVROE : la tension de référence est connectée sur RA2 (1) ou non CVRR : sélection de la formule de calcul de tension

CVRSS : sélectionne la tension de référence : VDD-VSS (0) ou bien (Vref+) - (Vref -)

CVR3-0 : si CVRR=1, alors CVREF= (CVR3-0/24)*VDD, si CVRR=0, alors CVREF=(VR3-0/32)*VDD+VDD/4

Attention toutefois, car la tension maximale que peut atteindre cette référence de tension est d'environ 3,6V.

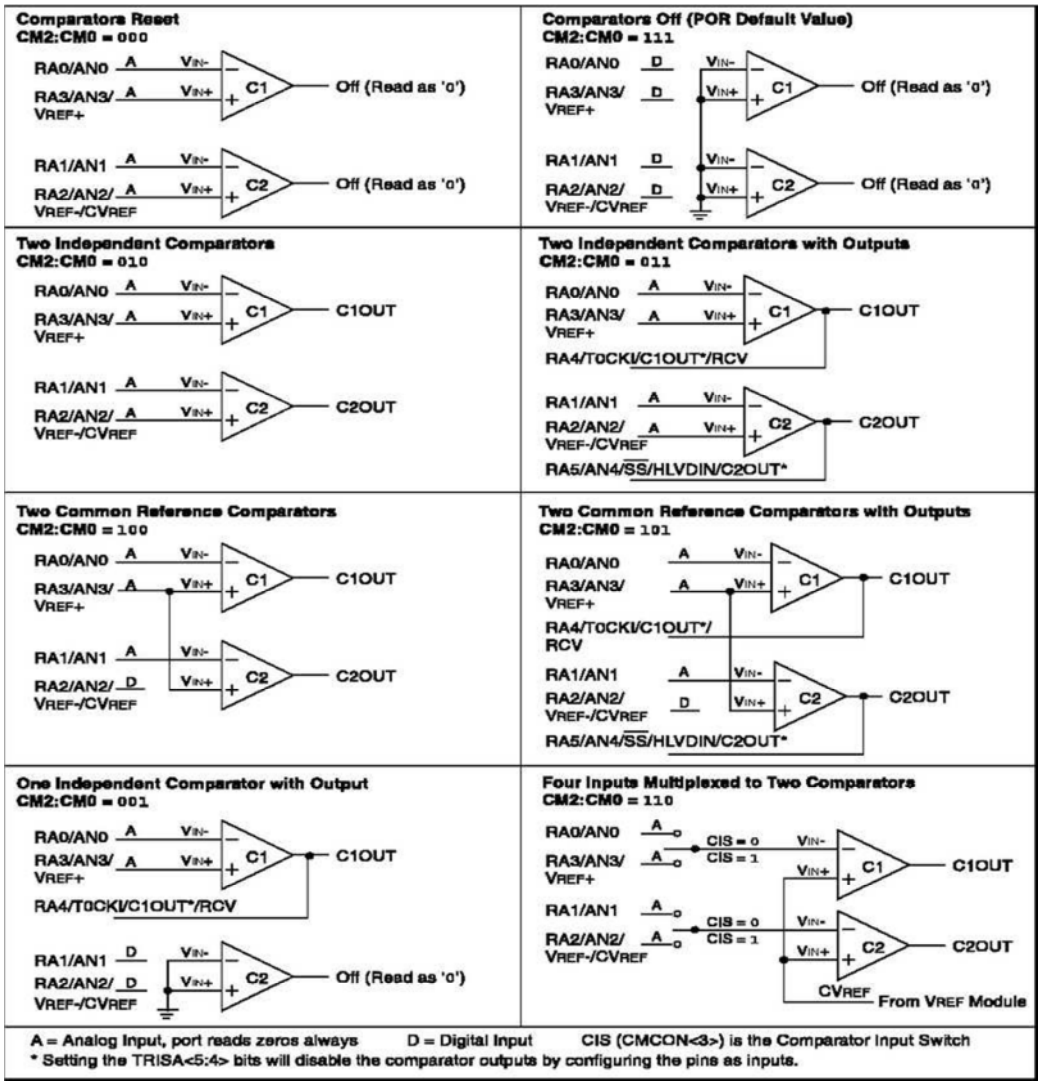
IV.3 Les comparateurs

La présence de comparateurs analogiques est fort pratique. En effet, cette fonctionnalité permet de réaliser la comparaison et de connaître au niveau logiciel & hardware le résultat de façon simple.

Le principe d'un comparateur, fort simple, est le suivant :

Figure III. 4: Principe d'un comparateur

Si V+>V- alors la sortie =1, 0 sinon. Ce comparateur permet de comparer diverses choses en fonction du mode de configuration.



Afin de configurer le comparateur dans le mode désiré, il faut configurer le registre CMCON. Le CVREF est une référence de tension.

CMCOM :

Figure III. 5: Les modes de configuration

C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0

C2OUT : en lecture seulement, permet de connaître logiciellement l'état de la comparaison (sortie du comparateur 2)

C1OUT : en lecture seulement, permet de connaître logiciellement l'état de la comparaison (sortie du comparateur 1) C2INV-C1INV : inverse l'état de la sortie (1) ou non

CIS : sélection d'entrée dans certains modes, voir schémas précédents

CM2-0 : configuration du comparateur, voir schémas précédents.

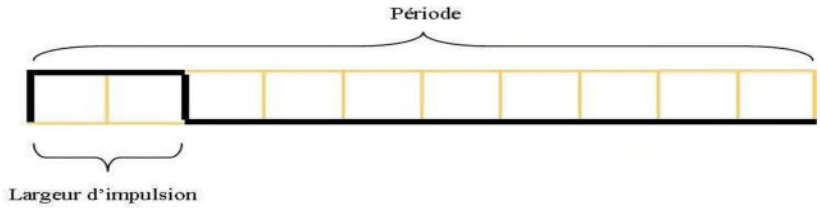
IV.4 La PWM

La PWM fait partie d'un bloc interne du PIC : le CCP ; ou Capture/Compare/PWM. Les deux premiers modes assez flous et d'une approche assez difficile ne seront pas vus ici, car très peu utilisés. Nous nous concentrerons sur la PWM, plus répandue.

La PWM pour Pulse Width Modulation, ou en français MLI pour Modulation en Largeur d'Impulsion, permet de gérer l'énergie transmise à l'extérieur. En effet, si un signal continu correspond à 100% d'énergie, un signal carré dont la durée d'état haut égale celle d'état bas correspond à 50 % d'énergie. Le pourcentage d'énergie transmis se calcule en faisant le rapport de la durée d'état haut sur la durée de la période.

La PWM peut permettre de gérer la luminosité d'une lampe, le pilotage d'un servomoteur de modélisme, contrôler la vitesse d'un moteur continu, ...

Voici une période PWM :



Deux formules permettent de calculer la durée de la période, et la largeur de l'impulsion.

PERIODE= (PR2+1)*4*Tosc*(prédiviseur timer 2)

LARGEUR= (10 bits)*Tosc*(prédiviseur timer 2)

Par Tosc, nous comprenons la période de l'oscillateur servant d'horloge au PIC. Les 10 bits correspondent à l'octet de CCPR1L et des 2 bits de DC1Bx. Cependant cette résolution de 10 bits est la valeur maximale. En effet, selon la fréquence désirée, cette résolution peut changer. Pour connaître la résolution maximale, on utilise la formule suivante :

RESOLUTION= (log (Fosc/Fpwm))/log (2) bits.

Par exemple, avec le pré diviseur timer 2 à 16, et un PR2 à 255 (0XFF), nous avons une résolution maximale de 10 bits, et une fréquence de PWM de 1,22 Khz.

Autre exemple, avec un pré diviseur à 1, et un PR2 à 31, la résolution maximale est alors de 7 bits, et une fréquence de PWM de 156,3 KHz.

Ici, les registres utiles sont le CCP1CON, CCP2CON, PR2, CCPR1L, CCPR2L. De plus, comme précisé précédemment, c'est le timer 2 qui est utilisé ici. De fait, il faudra également utiliser T2CON, afin d'activer le timer 2. Cependant, la plupart des compilateurs rendent toutes ces actions transparentes.

CCP1CON

		DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0

DC1B1-0 : contient les 2 bits de poids faibles de la période (PWM 10 bits) CCP1M3-0: Pour la PWM, égal à 11xx

Les 8 autres bits doivent être chargés dans le registre CCPR1L.

CCP2CON

		DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0

DC2B1-0 : contient les 2 bits de poids faibles de la période (PWM 10 bits) CCP2M3-0: Pour la PWM, égal à 11xx

Les 8 autres bits doivent être chargés dans le registre CCPR2L.

ADR9	ADR8	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2
ADR1	ADR0						

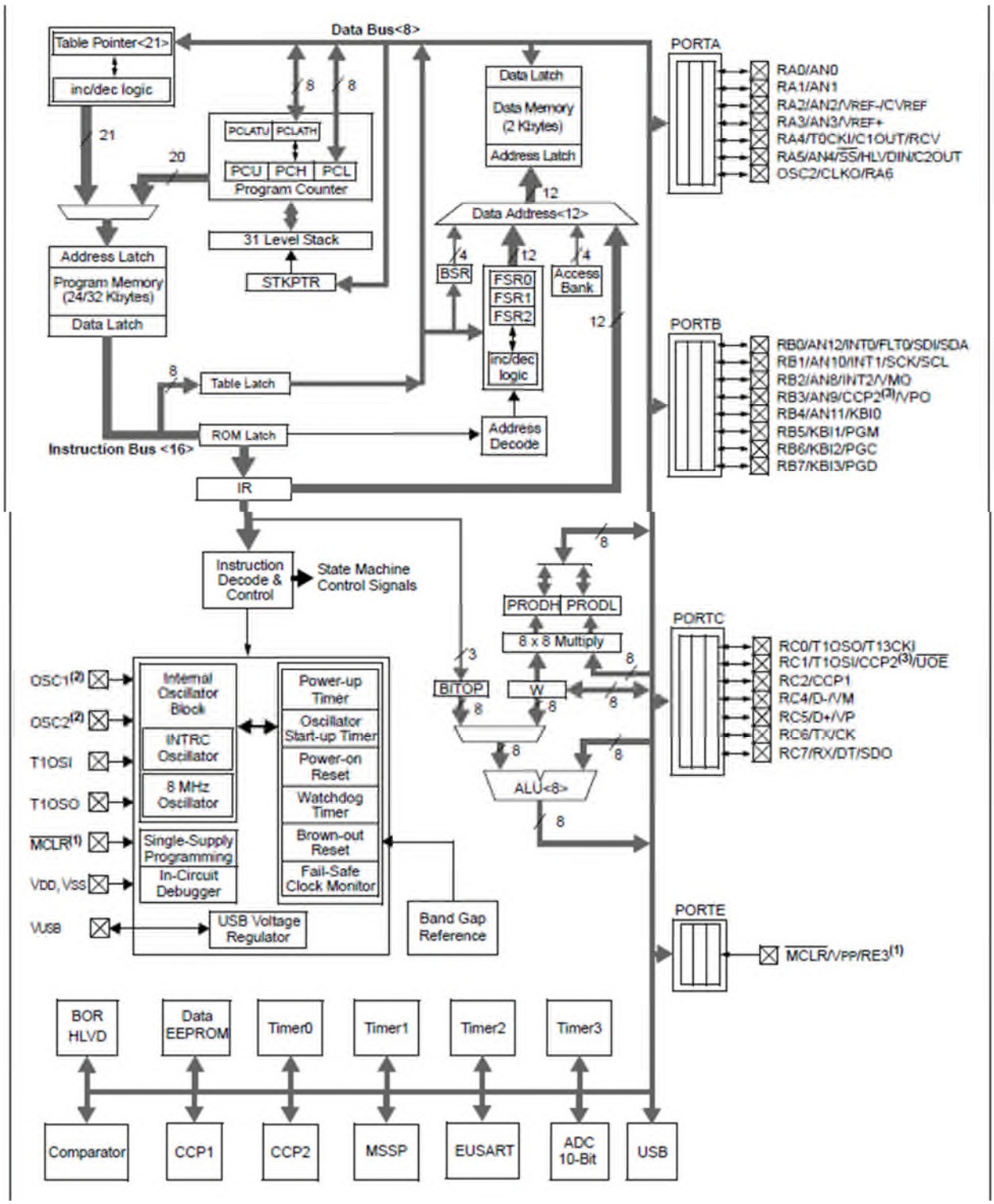


Figure III. 6: PIC18F2550 block diagramme

Chapitre IV LES CAPTEURS

I Etude théorique des capteurs, amplificateurs et filtres analogiques

Etude théorique des capteurs :

Généralités :

Le capteur est l'élément d'une chaîne ou d'un instrument de mesure auquel est directement appliquée la grandeur à mesurer. Son rôle est de transformer les valeurs de cette grandeur appelée mesurande, en signaux électriques exploitables par les autres éléments de la chaîne ou de l'instrument.

La variété des capteurs est telle qu'il est impossible en un seul chapitre de couvrir la totalité de leurs principes. Nous avons donc limité nos propos à quatre classes de mesurandes : électriques, mécaniques, optiques et manométriques. Cependant, même après cette restriction, il faut encore faire un choix et nous ne présenterons que les lois qui sont les plus couramment mises en oeuvre dans ce type de capteurs physiques.

Le capteur est le premier élément rencontré dans une chaîne de mesure.[1] C'est un dispositif qui transforme une grandeur physique en une grandeur exploitable, souvent de nature électrique. Une information est une grandeur abstraite (grandeur physique à mésurer **M**) qui précise un événement particulier parmi un ensemble d'événements possibles.pour pouvoir etre traitée, cette information sera portée par un support physique (énergie) on parlera alors de signal. Les signaux sont généralement de nature électrique (signal électrique de mesure **S**)[2].

Le capteur est l'interface entre le monde physique et le monde électrique. Il va délivrer un signal électrique image du phénomène physique que l'on souhaite numériser. Il est toujours associé à un circuit de mise en forme.

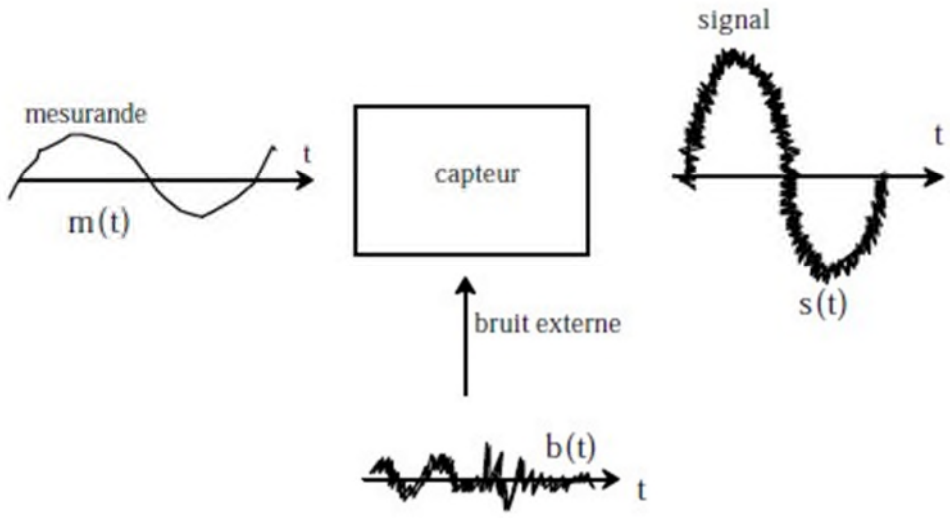


Figure IV. 1: schéma d'un capteur

Soit m la mesurande variable au cour de temps. Le but assigné au capteur est de convertir **m** en une grandeur électrique que l'on appellera **s** comme le montre la figure II.1

La mesure **s** peut être une impédance, une charge électrique, un courant ou une différence de potentiel. La relation qui lie **s** à **m** est : s(t) = m(t)+b(t) [1].

Avec s(t) : le signal de sortie, m(t) : la variation de mesurande et b(t) le bruit externe (grandeur d'influence). On appelle grandeur d'influence, toutes les grandeurs physiques autres que la grandeur à mesurer, susceptibles de perturber la mesure. Généralement les capteurs industriels sont compensés par un dispositif interne au capteur limitant l'influence des grandeurs perturbatrices. La température est la grandeur d'influence qui est le plus souvent rencontrée.

Principales caractéristiques des capteurs :

- *L'étendue de la mesure* : c'est la différence entre le plus petit signal détecté et le plus grand perceptible sans risque de destruction pour le capteur, [2]. La Figure II.2 présente la caractéristique typique d'un capteur linéaire.

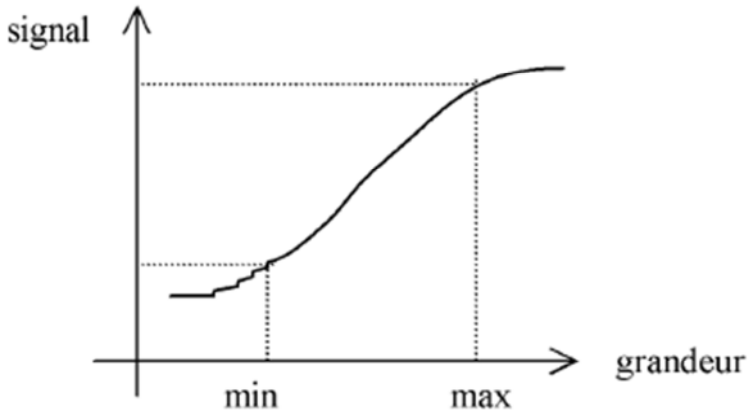


Figure IV. 2:) Étendue de mesure d'un capteur

- *La sensibilité* : c'est la plus petite variation d'une grandeur physique que peut détecter un capteur,
- *La rapidité* : c'est le temps de réaction d'un capteur entre la variation de la grandeur physique qu'il mesure et l'instant où l'information sera prise en compte par la partie commande [2],
- *La précision* : C'est la qualité qui caractérise l'aptitude d'un capteur à donner des indications proches de la valeur vraie de la grandeur mesurée,[3] Cela permet d'évaluer l'influence du capteur sur la mesure. On la définit non seulement vis à vis du capteur mais aussi vis à vis de l'environnement d'utilisation du capteur [4],
- *La linéarité :* zone dans laquelle la sensibilité du capteur est indépendante de la valeur de la mesurande, cette zone peut être définie à partir de la définition d'une droite obtenue comme approchant au mieux la caractéristique réelle du capteur, par exemple par la méthode des moindres carrés, on définit à partir de cette droite l'écart de linéarité qui exprime en
  % l'écart maximal entre la courbe réelle et la droite approchant la courbe.

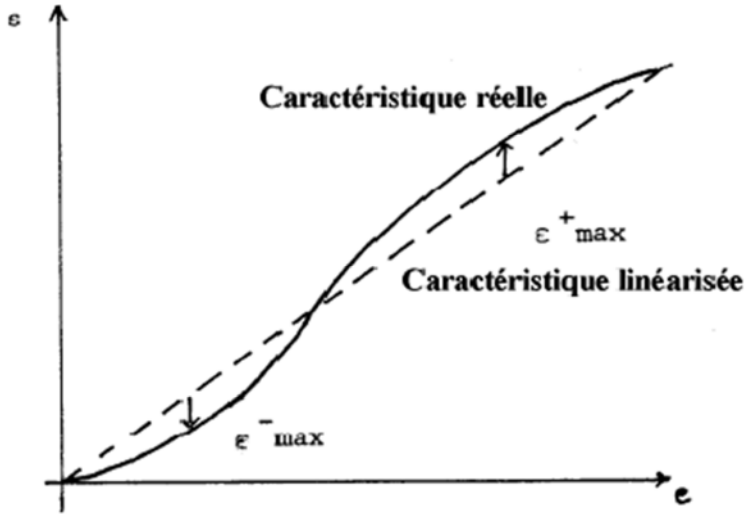


Figure IV. 3: Exemple de linéarisation de caractéristiques

Modes de fonctionnement des capteurs :

On classifie les capteurs en deux grandes familles en fonction de la caractéristique électrique de la grandeur de sortie. Cette classification influe sur le conditionneur qui lui est associé [4].

A. Les capteurs Actifs :

Un capteur actif est un capteur pour lequel la grandeur de sortie est une tension ou un courant électrique, cette grandeur est directement utilisable par l'instrumentation de mesure, donc il fonctionnant en générateur, dont une partie de l'énergie physique prélevée sur le mesurande est transformée directement en énergie électrique qui constitue le signal de sortie. Ce signal est un courant, une tension ou une quantité d'électricité.

Généralement, un capteur actif est un système de mesure qui nécessite une source d'énergie embarquée, la plupart du temps assurée par une batterie, et ce pour la réalisation de la phase de traitement au cours de laquelle le signal est filtré (nettoyé), amplifié et converti dans un format compatible et exploitable. Dans ce cas, le capteur doit non seulement mesurer des, propriétés physiques mais doit également effectuer des tâches additionnelles au travers de circuits de traitement et de communication intégrés. Ce type de capteur est surtout utilisé pour assurer des mesures continues en temps réel

[5]. Dans ce cas, la sortie du capteur est équivalente à un générateur. C'est un dipôle actif qui peut être du type courant, tension ou charge, figure (II.4 a) [4].

Figure IV. 4: les types de capture

(a) Capteur actif (b) Capteur passif

B. Les capteurs passifs :

Un capteur passif est un capteur qui ne fournit pas d'énergie. Sa grandeur électrique de sortie est généralement une impédance qui varie avec la grandeur à mesurer. Afin de pouvoir obtenir un signal utilisable, ce type de dispositif nécessite la mise en œuvre d'un conditionneur qui transforme ces variations en une différence de potentiel ou un courant. Ce type de capteur est utilisé dans des applications spécifiques (surveillance environnementale**,** des instruments de suivis spatial et aéronautique**,** des applications liées à la santé) qui nécessitent des unités de mesure miniatures, passives, de grande précision et fiables [5].

Les capteurs dont le signal électrique délivré est une variation d'impédance sont dits passifs car ils nécessitent une source d'énergie électrique pour que l'on puisse lire la sortie, figure (II.4b) [1]. Dans ce cas le capteur se comporte en sortie comme un dipôle passif qui peut être résistif, capacitif ou inductif [4].

Choix d'un capteur :

Tous les capteurs dont les fonctionnements ont été décrits précédemment présentent deux parties distinctes. Une première partie qui a pour rôle de détecter un événement et une deuxième partie qui a pour rôle de traduire l'événement en un signal compréhensible d'une manière ou d'une autre par une partie affichage (analogique ou numérique). Pour choisir correctement un capteur, il faudra définir tout d'abord :

- le type d'événement à détecter,

- la nature de l'événement,
- La grandeur de l'événement,
- l'environnement de l'événement.

En fonction de ces paramètres on pourra effectuer un ou plusieurs choix pour un type de détection. D'autres éléments peuvent permettre de cibler précisément le capteur à utiliser :

- ses performances,
- son encombrement,
- sa fiabilité,
- son prix... [2]

I-1.4. Conditionneur associé :

Le conditionnement de la mesure consiste à rendre exploitable la mesure issue du capteur, l'association capteur-conditionneur détermine le signal électrique et ses caractéristiques, on effectue une adaptation de la source du signal à la chaîne de mesure complète.

Par exemple, le capteur source de courant, le capteur peut se modéliser par une source de courant avec une impédance en parallèle, comme le montre la **figure (II.5)**.

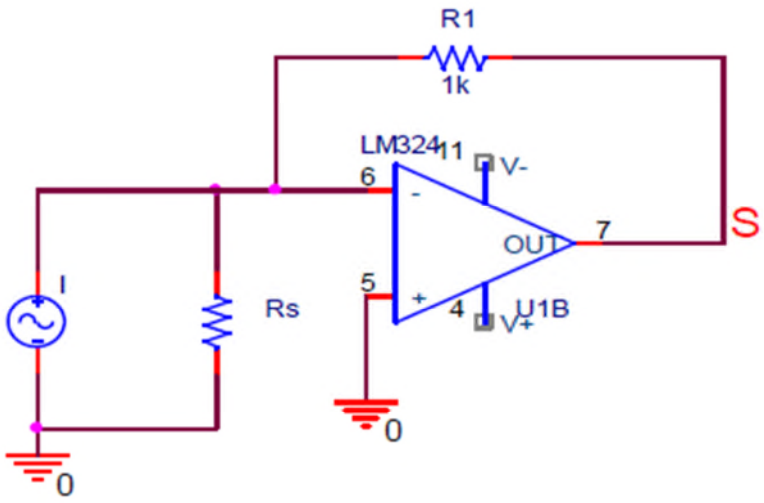


Figure IV. 5: Conditionneur convertisseur courant-tension

R_s : est court-circuitée Tension

sortie : **Vs= -R1*i**

On fait appel dans ce cas à un convertisseur courant-tension de manière à obtenir une tension proportionnelle au courant de sortie du capteur (utilisation d'un montage adaptateur d'impédance – transducteur I-V) [4]. Pour les capteurs passifs on associe la variation d'impédance du capteur à une source de tension ou une source de courant et la grandeur exploitée est la tension de sortie [3].

Applications médicales :

Dans le domaine de la médecine, les capteurs peuvent être utilisés pour assurer une surveillance permanente des organes vitaux de l'être humain. Ceux-ci pouvant être implantés sur la peau pour faciliter le diagnostic de maladies par la mesure continue de paramètres tel que : la tension artérielle, les battements du cœur, la fonction respiratoire, etc. [7]

Les capteurs biomédicaux :

Les capteurs biomédicaux sont des capteurs destinés à mesurer les signaux physiologiques générés par des organismes vivants et en particulier par les êtres humains. Le rôle joué par les capteurs biomédicaux est l'un des aspects techniques que l'on rencontre au cours du dialogue patient-machine, en effet toute transformation biologique doit, pour être exploitée, se présenter sans la forme de signaux compréhensibles, enregistrables et mesurable en valeurs normalisées. Ces informations sont nombreuses :

On peut classer les grandeurs physiques mis en jeu dans le cadre de ce projet en quatre familles.

- Electrique : capteur ECG (électrodes)
- Mécanique : capteur de vélocité pariétale (ultrasons)
- Optique : capteur de saturation d'O_2 du sang (photo transistor)
- Pression : capteur de respiration (inductif)

Le capteur est certainement l'élément le plus important dans une chaine de mesure et d'enregistrement. De ses qualités dépend la valeur de l'information médicale dans sa finalité et technique dans sa réalisation [8].

Puisque ces capteurs sont généralement appliqués au corps du patient, ils doivent donc obéir aux différentes contraintes suivantes :

- très haute sensibilité,
- supportant la stérilisation,
- possibilité d'usage unique,
- non invasivité,
- bio- compatibilité,

- résistance aux agressions du corps humain,
- provocation d'un minimum de perturbation du signal physiologique mesuré,
- fiabilité et stabilité de leurs caractéristiques,
- haute sécurité.

On distingue généralement deux types de capteurs biomédicaux : les capteurs invasifs et les capteurs non invasifs, Les capteurs invasifs sont implantés en partie ou en totalité à l'intérieur du corps humain, Ces capteurs délivrent des mesures de bonne qualité (très peu bruitées) parce qu'ils sont placés très près de la source des signaux qu'ils mesurent. A l'inverse, les capteurs non invasifs sont placés à l'extérieur du corps humain. Cependant, les mesures qu'ils génèrent sont plus bruitées que celles des capteurs invasifs parce qu'ils sont plus éloignés que ces derniers de la source des signaux qu'ils mesurent.

I-2.1.1. Les électrodes :

Les êtres vivants sont communément le siège de phénomènes électriques intimement liés aux activités vitales, dont ils sont un des aspects les plus révélateurs. Ces phénomènes électrophysiologiques sont mis en évidence à l'aide de capteurs (électrode) appliqués en surface ou introduit dans la profondeur des tissus [6].

C'est un convertisseur de courant ionique en courant électrique. Son fonctionnement s'explique par les principes d'électrochimie qui énoncent que tout conducteur métallique **M** plongé dans une solution du même métal est le siège d'une réaction chimique qui s'équilibre pour un potentiel **E** entre l'électrode et la solution.

La chaine de mesure peut engendrer un courant qui traversera l'électrode et changera donc le potentiel de mesure. Ce courant dit de « polarisation » doit être minimisé afin d'assurer une mesure précise et la sécurité du patient. L'électrode Ag/Agcl qui répond à cette exigence est la plus utilisée. Les électrodes de surface sont de simples plaques métalliques reliées à la peau du patient par une pâte électrolytique (gèl conducteur) **Figure (II.6)**. Pour réduire les artefacts du mouvement, une isolation mécanique a été collée à sa périphérie [8].

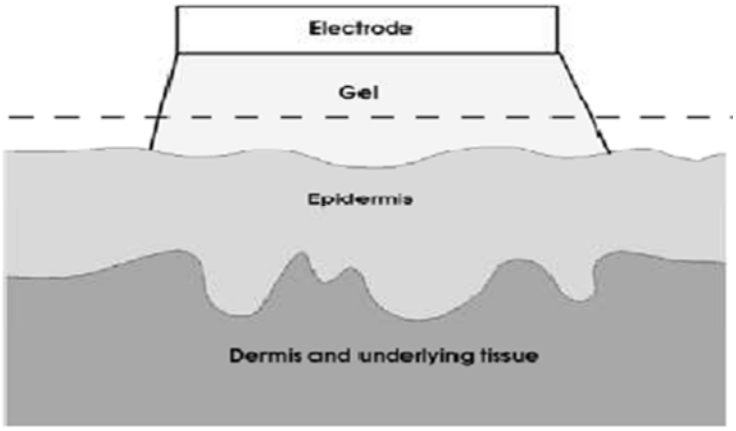


Figure IV. 6: Electrodes de surface

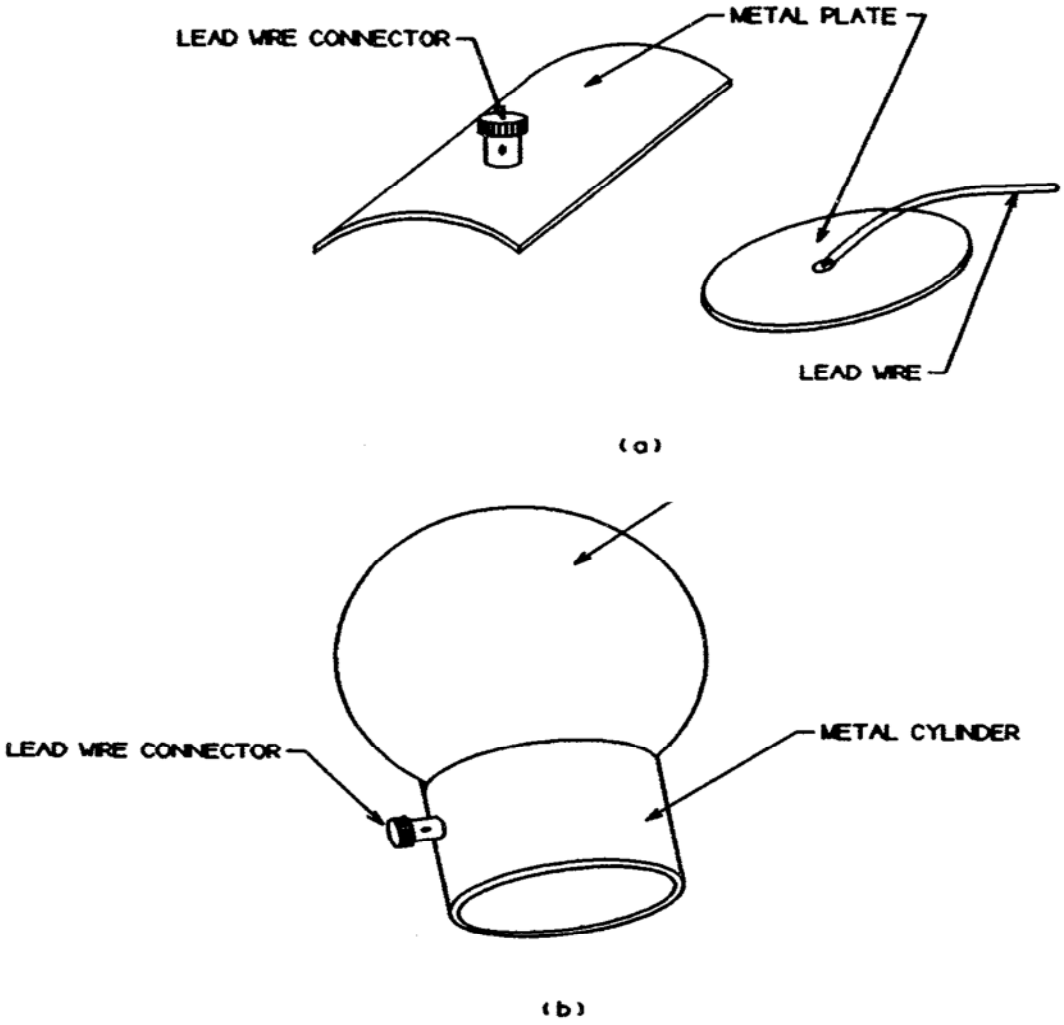


Figure IV. 7: Exemple de différentes électrodes d'ECG

I-2.1.2. Le capteur ultrasonique US :

Son principe de fonctionnement repose comme son nom l'indique sur l'utilisation des ultrasons. Ce sont des vibrations mécaniques de la matière qui se propagent dans tout support matériel (solide, liquide ou gaz) présentant une certaine élasticité. Ils correspondent à des fréquences oscillatoires supérieures à 20 kHz (seuil de perception de l'oreille humaine)… [11], [17].

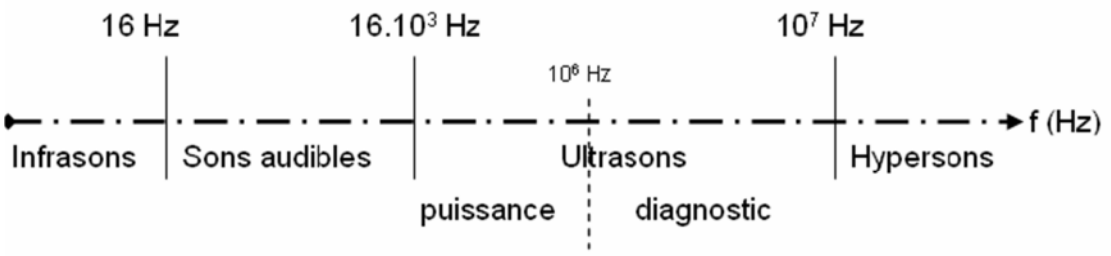


Figure IV. 8: Domaines de fréquence des sons

Le succès des ultrasons dans le domaine médical doit beaucoup au faible coût des appareils ultrasonographiques, au caractère non invasif de ce rayonnement à faible intensité et également à la simplicité de cette technique. Elle consiste à interroger un milieu à l'aide d'une impulsion ultrasonore et à enregistrer les échos générés par des inhomogénéités. En médecine celles-ci peuvent être des organes, des échantillons biologiques en général. Le signal rétrodiffusé est ensuite soumis à un traitement simple dans lequel son enveloppe par exemple est extraite.

L'extraction de ces informations latentes est un vaste domaine de recherche qui comprend par exemple: la détection d'écoulement, la rhéographie pariétale et se révèle particulièrement efficace dans le diagnostic des maladies cardio-vasculaires [13].

Principes du Doppler (1803 - 1853):

L'application médicale de l'effet doppler, découvert en 1843 par **Johann Christian Doppler**, n'est apparue que dans les années 1960. Elle n'est réalisée que vers les années 1960, avec l'utilisation d'ondes ultrasonores.

L'effet Doppler repose sur l'émission d'ultrasons qui sont des vibrations mécaniques comparables aux sons audibles mais de fréquence plus élevée [14]. Émis par une source rencontre une cible fixe, la fréquence réfléchie par cette cible est identique à la fréquence émise. Quand la cible se déplace, la fréquence réfléchie est différente de la

fréquence émise. Cette différence (ΔF) entre la fréquence d'émission (Fe) et la fréquence de réception (Fr) s'appelle la fréquence doppler, Figure (II.9). ΔF est exprimée en hertz (Hz). Elle est située dans un spectre audible.

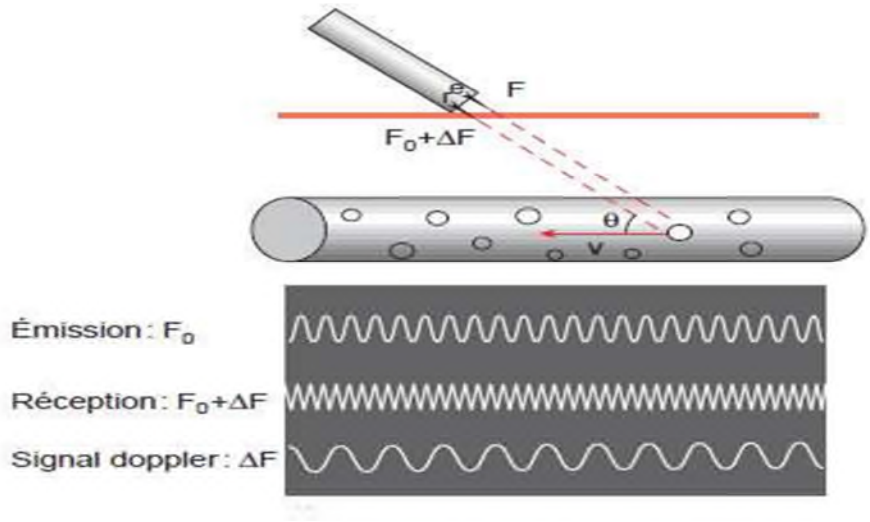


Figure IV. 9: Schéma d'un doppler continu

Ainsi, l'effet doppler permet de déceler le mouvement des hématies dans un vaisseau sanguin. La fréquence doppler s'exprime par la relation suivante :

– Fe : fréquence d'émission de la sonde ;

– Fr : fréquence de réception de la sonde ;

– V : vitesse des éléments figurés dans le vaisseau ;

– θ: angle entre l'axe du vaisseau et l'axe du faisceau ultrasonore ;

– C : vitesse moyenne des ultrasons dans le corps humain (1 540 m/s) ;

Fe : est en général comprise entre 2 et 10 MHz. Le choix de la fréquence d'émission résulte d'un compromis entre l'atténuation de l'onde ultrasonore (fonction de la fréquence et de la profondeur de l'examen) et le pouvoir de rétrodiffusion des organes qui croît avec la fréquence.

L'angle θ est un paramètre capital. En effet, si Cosinus (cos) θ= 0, ΔF est nulle. Pour

un angle de 90° entre le vaisseau et le faisceau ultrasonore, on n'obtient aucun signal

doppler. Pour un angle de 0°, faisceau ultrasonore parfaitement dans l'axe du vaisseau, l'effet doppler est maximal.

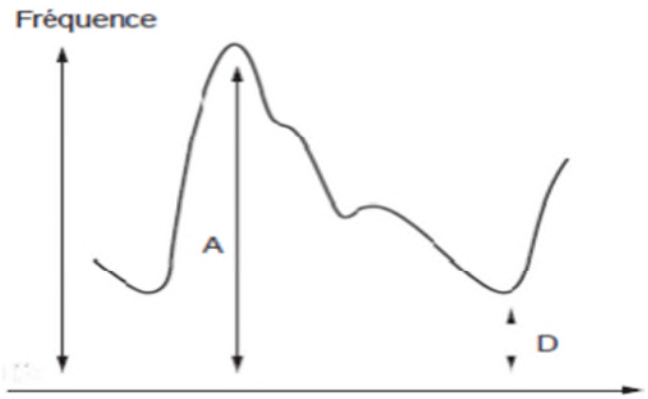


Figure IV. 10: Tracé spectral d'une artère à basse résistance. A : vitesse systolique D : vitesse diastolique

Pour un fluide connu (C connu) et un dispositif émetteur-récepteur connu (θ connu) on obtient directement la vitesse V en mesurant le glissement fréquentiel ΔF :

La fréquence Doppler est proportionnelle à la vitesse du flux sanguin. L'effet Doppler permet donc de mesurer la vitesse d'écoulement si l'on connait l'angle de tir Doppler

(C et Fe connues). Aux fréquences Doppler élevées correspond des vitesses rapides, et inversement, les fréquences basses traduisent l'existence de flux lente.

L'importance des variations systolo-diastoliques des vitesses d'écoulement reflétant le degré des résistances vasculaires, Figure (II.10).différentes méthodes de représentation du signal Doppler ont été proposée afin de fournir un support simple à l'interprétation des tracés Doppler et pour permettre une analyse quantitative des principaux paramètres vélocimétriques, (**voire la chapitre III**) [17].

Différents modes doppler :

A. Doppler continu :

Il utilise une émission continue d'ultrasons avec une sonde à deux cristaux, l'un émetteur, l'autre récepteur, **Figure (II.9)**. Il a un inconvénient majeur : il n'y a pas de repérage en profondeur possible, c'est-à-dire l'absence de localisation spatiale du

signal. La variation de fréquence du faisceau incident résulte de la sommation de tous les flux rencontrés sur le trajet du faisceau ultrasonore.

Ainsi, deux structures voisines peuvent être échantillonnées ensemble et ne peuvent donc être distinguées. Il y a donc ambiguïté en profondeur. En revanche, il donne l'avantage de ne pas limiter la vitesse mesurable. Il n'y a pas d'ambiguïté en vitesse. De plus, le doppler continu est très sensible pour détecter les flux lents, [15] l'excellente qualité des spectres fournis par cette technique est liée à la réception continue du signal [17].

B. Doppler pulsé :

Il est constitué d'une sonde qui, alternativement, émet un faisceau ultrasonore et reçoit le faisceau réfléchi, **figure(II.11).**

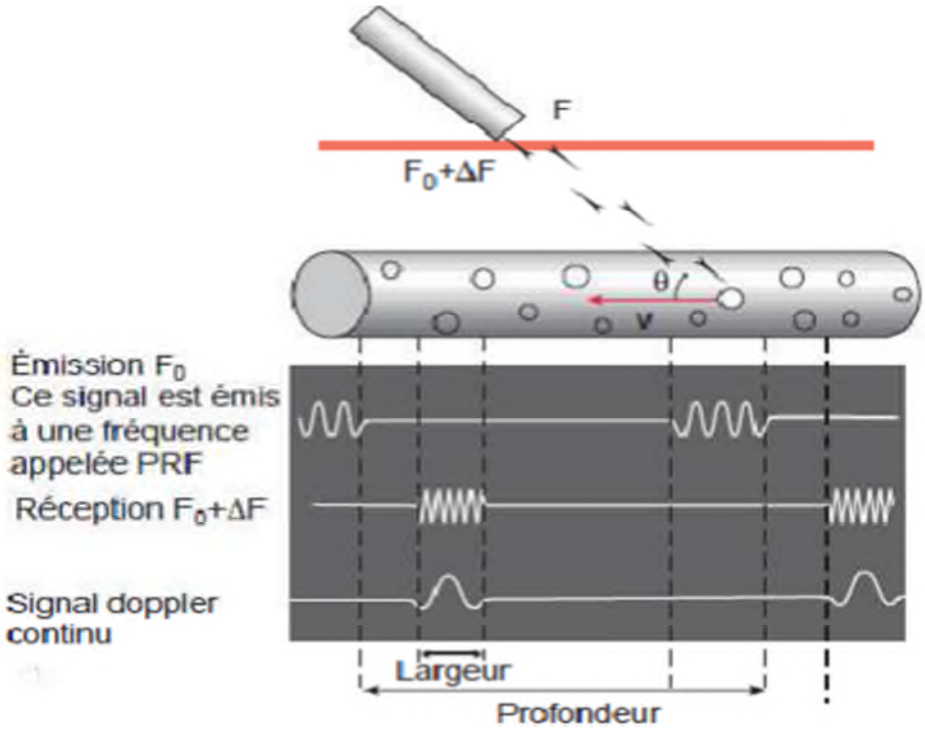


Figure IV. 11: Schéma d'un doppler pulsé. L'émission spectrale du signal F0 est discontinue et inversement proportionnelle à la profondeur. Le nombre de signaux émis par seconde est la PRF

Le délai entre deux impulsions détermine la fréquence de répétition (PRF [pulse repetition frequency]

C : vitesse des ultrasons ; d : profondeur du vaisseau, (2d pour l'aller retour des US).

La PRF détermine la profondeur du champ d'exploration, car il faut attendre le retour de tous les échos avant d'émettre une nouvelle impulsion. Les échos venant des zones les plus profondes fixent ainsi l'intervalle de temps à respecter avant un nouveau tir.

La PRF détermine également la sensibilité aux flux. Une PRF basse est nécessaire pour explorer en profondeur et détecter des flux lents.

Une PRF élevée est nécessaire pour bien analyser les flux rapides (évitant l'aliasing sur lequel nous reviendrons). On peut augmenter la PRF également si on analyse les régions superficielles.

Entre deux impulsions, le signal réfléchi est analysé pendant une durée très courte appelée « fenêtre d'écoute ». Le délai entre la fin de l'impulsion et le début de la fenêtre d'écoute (P) permet de déterminer la profondeur du volume d'échantillonnage.

L'intérêt du doppler pulsé par rapport au doppler continu est de pouvoir bénéficier de

la résolution spatiale et de focaliser l'examen sur un vaisseau à analyser [15].

Suivant le théorème de shannon, il est nécessaire de prélever au mois deux échantillons par période du phénomène périodique observé. C'est une des limites de Doppler pulsé [17].

Principe technique (Transduction électromécanique) :

Pour émettre et recevoir l'onde ultrasonore, on utilise des "transducteurs" (appelés aussi "traducteurs" ou "palpeurs"). A l'émission, le transducteur génère une onde ultrasonore par effet piézoélectrique. A la réception de l'onde, le transducteur convertit l'énergie mécanique perturbée en signal électrique [11].

La production d'ultrasons se fait principalement par la transformation d'oscillations électriques en vibrations mécaniques. Différentes méthodes de transformation de l'énergie électrique en énergie mécanique peuvent être utilisées, telle la magnétostriction ou la magnéto-induction. Cependant, le phénomène physique le plus utilisé est la piézoélectricité, qui est un phénomène propre à certains matériaux anisotropes [18].

La conversion *piézo-électrique*, basée sur la déformation de certains matériaux lorsqu'ils sont soumis à un champ électrique, et réciproquement le changement de leur polarisation sous l'action de contraintes mécaniques [19].

Application à la vélocimétrie pariétale :

Dans notre cas nous utilisons l'effet Doppler en vue d'une tentative de caractérisation des propriétés viscoélastiques de la paroi vasculaire au moyen de l'enregistrement de la vitesse de déplacement de celle-ci transmis à la surface de la peau sus-jacente de manière localisée (rhéographie ultrasonore localisée). Pour cela nous utilisons une fréquence ultrasonore de 40KHz relative à la propagation des ultrasons dans l'air.

I-2.1.3. Capteur à pression différentielle:

La pression est une notion physique fondamentale et une grandeur de mesure importante, on peut la voir comme une force rapportée à la surface sur laquelle elle s'applique [22].

Elle se définit comme suit :

$$P = P/S$$

avec P : pression en N/m² (1 Pa = 1 N/m²)

F : force en Newton

S : surface en m²

I-2.1.3.1. Pression différentielle **:** C'est la différence de deux pressions ou la différence de grandeur entre une valeur de pression donnée et une pression de référence donnée.

I-2.1.3.2. Pression hydrodynamique : elle résulte de la vitesse du fluide ou gaz en mouvement. Un fluide qui se déplace crée une pression supplémentaire.

Le principe de conversion du débit d'air en tension électrique, est d'utiliser un Fluxmètre à pression différentielle. Il s'agit de créer au sein de la canalisation une restriction localisée de la section (ou constriction) qui engendrera une différence de pression statique dont la mesure nous permettra d'en déduire le débit.

Fluxmètre à pression différentielle (les pneumotachymètre PTM):

Une différence de pression entre deux points entraine l'existence d'un flux convectif. A traverse la relation entre la différence de pression et le flux de volume à traverse un système on peut estimer le flux de volume. Les pneumotachymètres PTM, mesurer le flux instantané du volume de gaz respiré. Malgré que ce système mesurent

directement le flux du volume de gaz, il peuvent être utilisés pour déterminer les changements de volume absolu des poumons (spiromètre) en intégrant électroniquement le signal flux [23]. Le pneumotachymètre du type à écran fin **figure(II.11)** consiste en un écran très fin. Cet écran est introduit dans un couvercle métallique ou plastique. Cette configuration de transducteur permet de maintenir un flux laminaire sur une large gamme de débit de flux. Dans le cas d'un flux laminaire, la chute de pression à traverse l'écran est directement proportionnel au flux d'air le traversant.

Dans notre cas nous utilisons la pneumotachographie de Fleish qui consiste à mesurer la différence de pression entre deux points d'un écoulement laminaire et qui utilise la loi de Poiseuille selon laquelle le débit est proportionnel à la différence entre les deux points de l'écoulement

$$D=k.\Delta P$$

La méthode consiste à utiliser deux tubes soumis à la pression dans deux endroits différents à l'intérieur la tête de Fleish. Une membrane située à l'intérieur d'un circuit magnétique à reluctance variable se déplace au rythme de la variation ΔP c'est-à-dire en fonction du débit d'écoulement d'air D généré par la respiration (figure II.12). Un circuit de conditionnement permettra de transformer la différence de pression en grandeur électrique.

La mesure de débit, réduite à la mesure des différences de pression ΔP du gaz qui apparaît entre deux points situés en amont et en aval de l'étranglement de la conduite est donc réalisée par un capteur de pression différentielle.

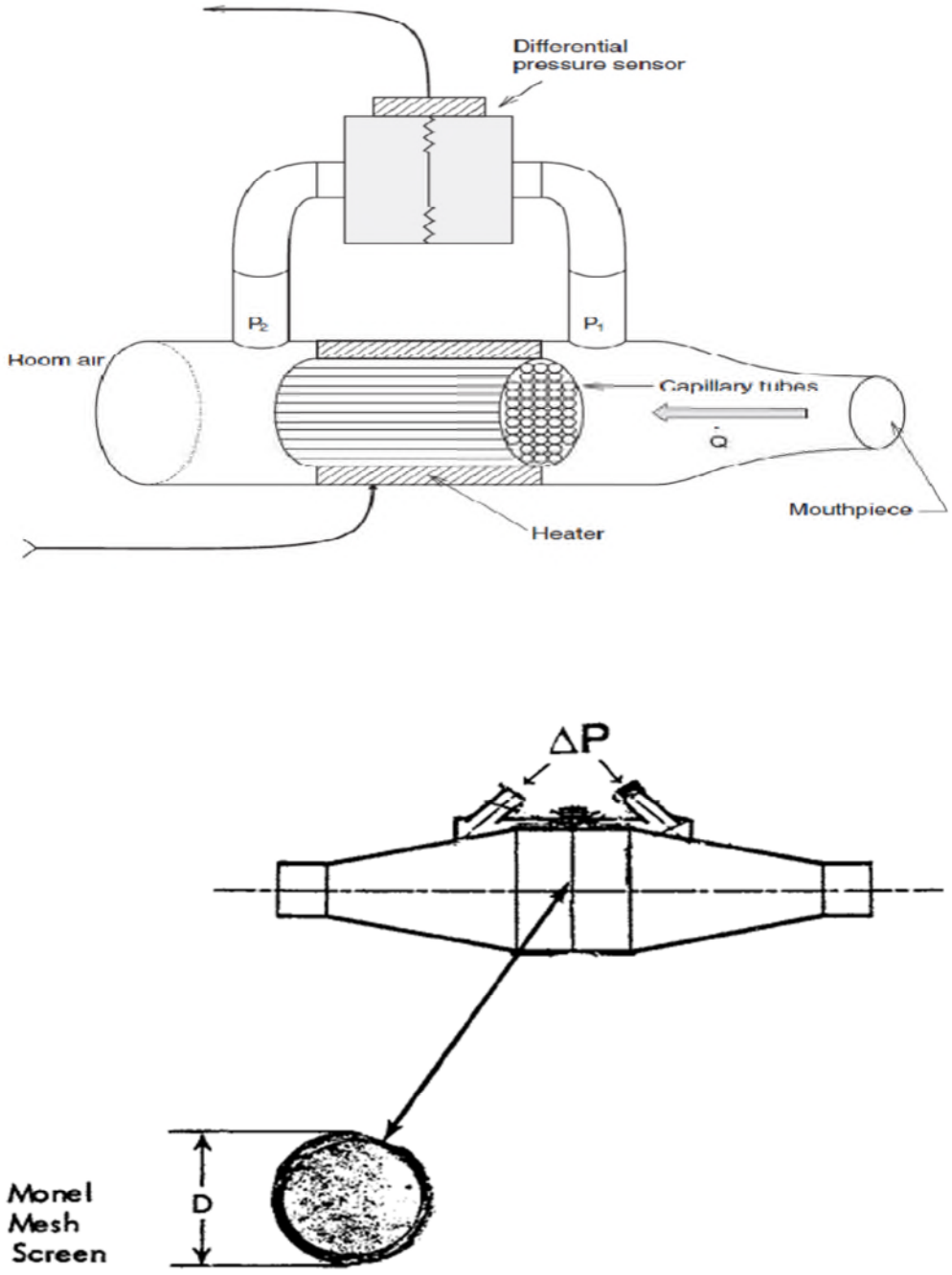


Figure IV. 12: Différente type de capteur à pression différentielle [24], [35]

Capteurs inductifs :

La variation du flux d'induction magnétique dans un circuit électrique induit une tension électrique. Les capteurs inductifs produisent à l'extrémité de leur tête de

détection un champ magnétique oscillant. Ce champ est généré par un matériau ferromagnétique, lorsqu'il est soumis à l'excitation électrique. Lorsqu'un objet métallique pénètre dans ce champ, il y a perturbation de ce champ puis atténuation du champ oscillant. Cette variation est exploitée par un amplificateur qui délivre un signal de sortie, de nature électrique.

I-2.1.4. Capteur optique :

Une classe importante de capteurs concerne la détection des faisceaux électromagnétiques. Un capteur optique est un dispositif capable de détecter l'intensité ou la longueur d'onde des photons. Il se compose d'un émetteur de lumière associé à un récepteur. La détection d'un objet se fait par coupure ou variation d'un faisceau lumineux. [2] La mesure (grandeur de sortie) des capteurs optiques est généralement un courant. C'est à partir des variations de ce courant en fonction de différents paramètres que l'on définit les performances du capteur [1]. Le signal est amplifié pour être exploité par la partie commande [2].

L'effet photoélectrique :

La libération de charges électriques dans la matière sous l'influence d'un rayonnement lumineux ou plus généralement d'une onde électromagnétique dont la longueur d'onde est inférieure à un seuil caractéristique du matériau.

Albert EINSTEIN a montré en 1905 que l'impact d'un photon de fréquence **V** sur un métal suffisait à en extraire un électron si l'énergie du photon **h.V** dépassait l'énergie d'extraction **W** nécessaire pour dégager l'électron du métal. C'est le phénomène photoélectrique mis en œuvre dans la plupart des capteurs. Réciproquement, si un électron libre **e** percute un électron **e'**. Si l'énergie apportée est suffisante, ce dernière se déplace sur une orbite d'atome de plus grand diamètre. En retrouvant sa position d'origine, il restitue l'énergie reçue sous forme de photons. C'est le principe mis en œuvre dans les émetteurs photoélectriques (diodes électroluminescentes).

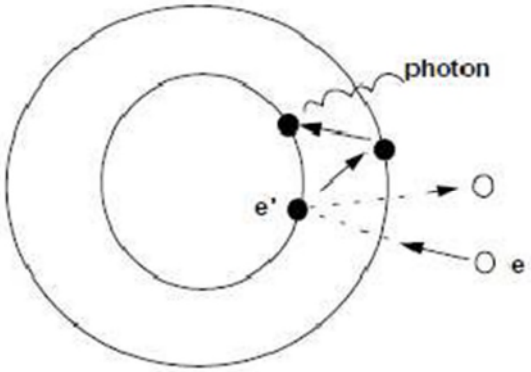


Figure IV. 13: L'effet photoélectrique

Les photodiodes :

Les photodiodes sont des diodes au silicium qui exploitent l'effet photoélectrique. Sous éclairement, les photons libèrent des paires électron-trous. Sa polarisation en inverse produit un courant (**IR**) qui augmente proportionnellement à l'intensité lumineuse. Les photodiodes sont beaucoup utilisées en raison de leur grande stabilité, de leur faible encombrement, de leur coût réduit.

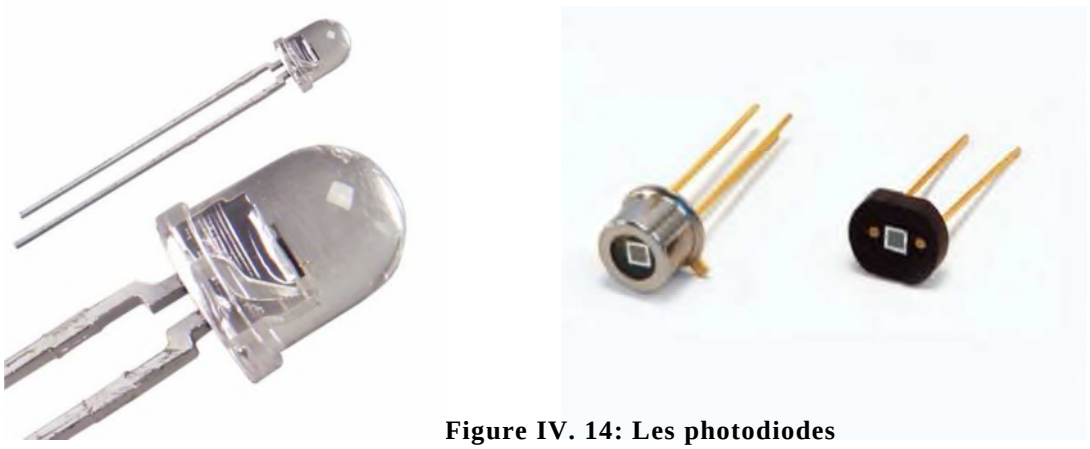


Figure IV. 14: Les photodiodes

La photodiode est un composant basé sur la jonction d'un semi conducteur de type **P** et d'un semi conducteur de type **N** :

- Chaque photon absorbé par le semi-conducteur peut créer une paire électron-

trous,

- sous l'action du champ interne, l'électron se diffuse vers la zone **N** et le trou vers la zone **P**,
- on a une diffusion des trous et des électrons dans des sens opposés,
- ces porteurs donnent naissance à un photocourant de génération.

Les phototransistors :

Le faible courant électrique transmis par les photodiodes a poussé les constructeurs de semi-conducteurs à rajouter à ce composant un transistor donnant ainsi naissance au phototransistor, Les caractéristiques sont sensiblement les mêmes photodiodes. Mais on remarque un courant beaucoup plus important.

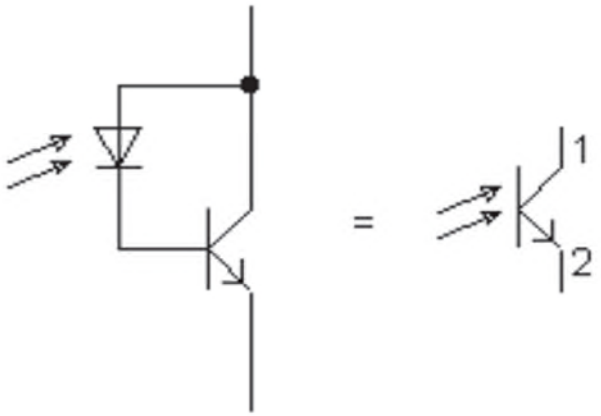


Figure IV. 15: Photodiode et phototransistor

I-2.1.4.4. Principe de fonctionnement du capteur SpO2 :

Les appareils de mesure fonctionnent selon deux principes :

- La **pléthysmographie**, qui utilise l'absorption des ondes lumineuses pour reproduire les ondes émises par le sang pulsatif.
- La **spectrophotométrie**, qui mesure l'absorption de la lumière à travers les substances à certaines longueurs d'onde.

Nous venons de voir que la couleur du sang variait selon son oxygénation autrement dit, sa saturation. C'est donc grâce à cette propriété que les capteurs des appareils vont pouvoir déterminer la SpO2. L'appareil va d'une part repérer chaque onde pulsatile artérielle et ensuite déterminer la saturation selon la couleur du sang qui sera déduite selon l'absorption des lumières émises. Il va donc fournir une SpO2, mais également une fréquence cardiaque [25]. Le principe de base de l'oxymétrie de pouls relève de la spectrophotométrie percutanée et de la loi de Beer-Lambert, c'est-à-dire que l'on utilise les propriétés de réflexion de la lumière de molécules pour mesurer la concentration d'entités chimiques dans un environnement liquide ou gazeux.

La loi de Beer :

Ce paragraphe décrit le contexte théorique pour la mesure de l'absorbance de lumière dans le tissu biologique en tant qu'élément de base pour la détermination de saturation en oxygène et la courbe pléthysmographique.

II Réalisation pratique des différents dispositifs

Nous mettons en œuvre dans ce chapitre la réalisation pratique des différents dispositifs électroniques constituant le plateau technique dédié à l'exploration cardiovasculaire multiparamétrique

III Introduction :

Le principe le plus classique d'une mesure physiologique consiste à capter, amplifier mettre en forme et visualiser les variations de grandeurs physiques issus des différents du corps humain. Dans ce chapitre nous présentons les différents circuits que nous avons réalisés.

La mise en forme d'un signal :

La mise en forme du signal est une chaine de mesure qui comporte un certain nombre de composants électroniques, permettant le traitement analogique de signal (amplification, filtrage, adaptation d'impédance, calibration, étalonnage, linéarisation…). Après la mise en forme nous ramenons le signal de sortie des différents circuits à un niveau compatible pour être connecté à un microcontrôleur .

Réalisation pratique de la chaîne d'amplification de l'Electrocardiogramme (ECG):

Le schéma bloc suivant représente la réalisation pratique d'un électrocardiogramme ECG. A travers ce schéma bloc deux grandes parties se distinguent, partie des électrodes et la partie d'amplification différentielle.

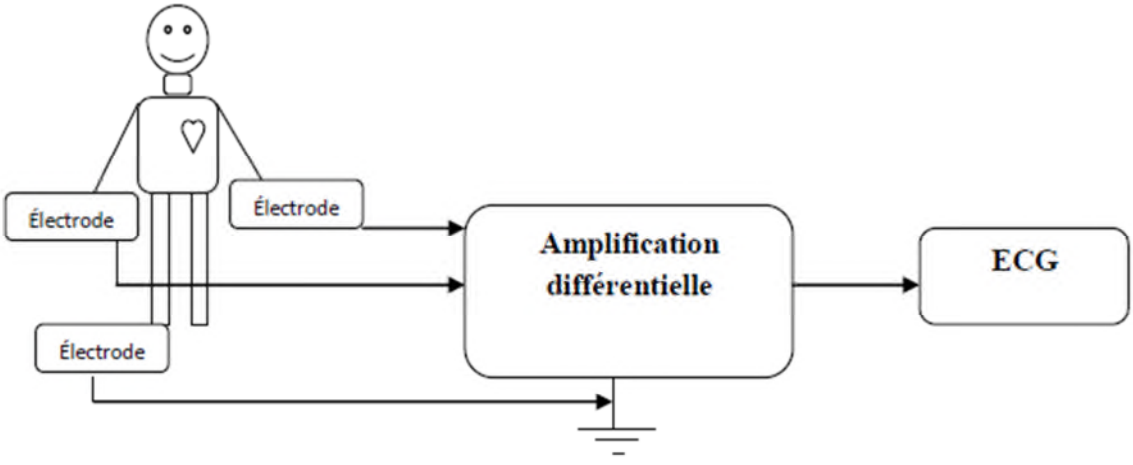


Figure IV. 16: Schéma bloc du système réalisé ECG

Les signaux captés étant particulièrement faibles, des amplificateurs de hautes performances (gain, linéarité, différentialité, minimum de bruit de fond) sont souvent nécessaires.

L'indisponibilité des composants spécialisés en instrumentation biomédicale nous à posée certaines difficultés. Aussi nous nous sommes rabattus sur le simple amplificateur opérationnel LM324 disponible. Le premier étage de la chaîne de traitement est le plus important .C'est celui où on a recours a l'Amplificateur d'instrumentation *(Instrumentation Amplifier).*Cet élément prend en charge l'amplification, l'adaptation, l'élimination de la tension de mode commun et même, dans certains cas, l'isolation du patient des courants de fuites.

Après le recueil de l'information, un traitement de celle-ci est nécessaire .Ce dernier inclut dans notre cas : l'amplification en améliorant le gain et le filtrage pour éliminer le bruit.

Un A.I. est un amplificateur différentiel (**Voir le chapitre II**) adapté au traitement des signaux en présence d'une tension de mode commun relativement importante et dont les caractéristiques sont les suivantes :

- Gain différentiel réglable (de 1 a 10000)
- Impédance d'entrée très élevée (10 KΩ en parallèle avec quelques µF)
- Impédance de sortie très faible (0.1Ω).
- Courant de polarisation des entrées très faible (de quelques pA a quelques nA).
- Grande stabilité thermique des performances (0.0015 %/°c pour le gain différentiel).
- Taux de réjection en mode commun très élevé (>100dB).

Les signaux que l'on visualise sont généralement mesurés par rapport à une électrode dite de référence (CONN-SIL3, **FL**), (**figure III.2**). Celle-ci est placée à un endroit précis du corps (pied gauche). Il s'agit donc de mesurer le potentiel de chaque électrode (CONN-SIL1**LA** et CONN-SIL2**RA**) par rapport à celui de l'électrode de référence (celle-ci ne constituant pas une masse). Une mesure différentielle s'impose donc et l'utilisation d'un amplificateur différentiel est nécessaire.

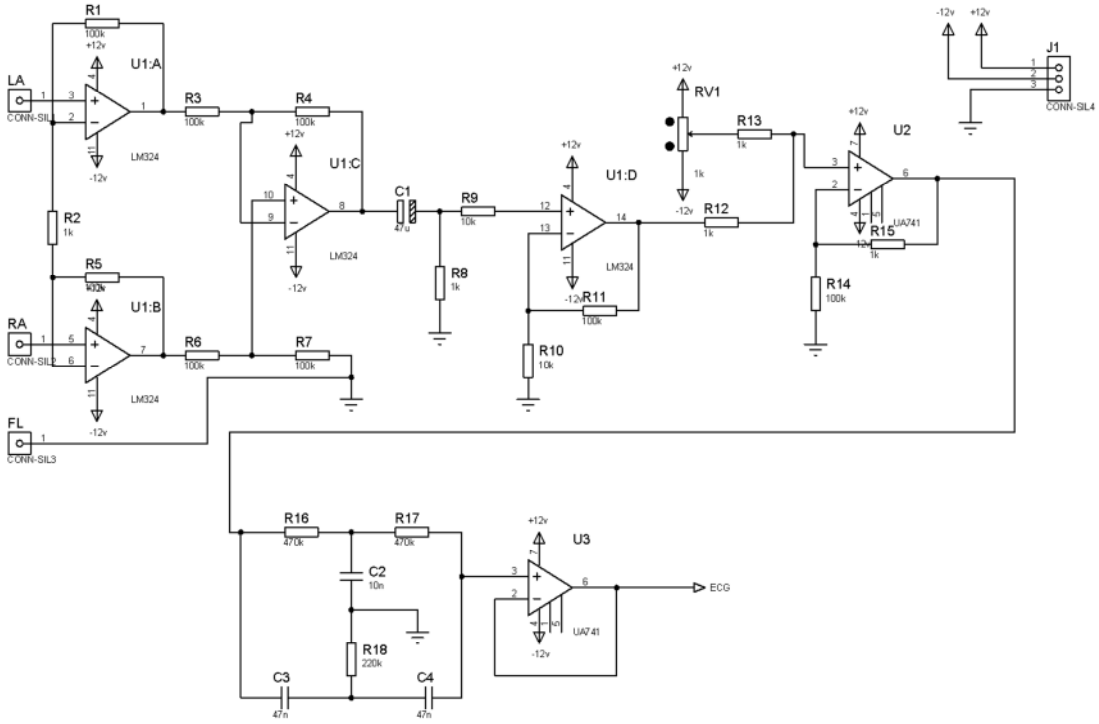


Figure IV. 17:Schéma électrique d'ECG

Le signal obtenu en employant l'amplificateur LM324 (figure III.2) est un signal extrêmement faible et bruité, influencé principalement par le 50Hz, donc pour bien l'amplifier et éviter ces perturbations, nous avons employé un amplificateur

et un filtre de type Twin T passif pour éliminer le 50HZ. L'amplificateur U2 (de type µA741) est utilisé, avec un potentiomètre pour régler la composante continue. Un deuxième amplificateur opérationnel U3 (de type µA741), à la fin de ce schéma monté en suiveur, permet de réaliser l'adaptation d'impédance. Après avoir été filtré et amplifié, le signal ECG est transmis à la carte d'acquisition et peut alors être visualisé sur un terminal local moyennant l'implémentation d'un ETTD micro contrôlé (équipement terminal de traitement des données) (voir le chapitre IV et V).

Les figures suivantes représentent le circuit imprimé et la visualisation 3D du circuit électrique précédant.

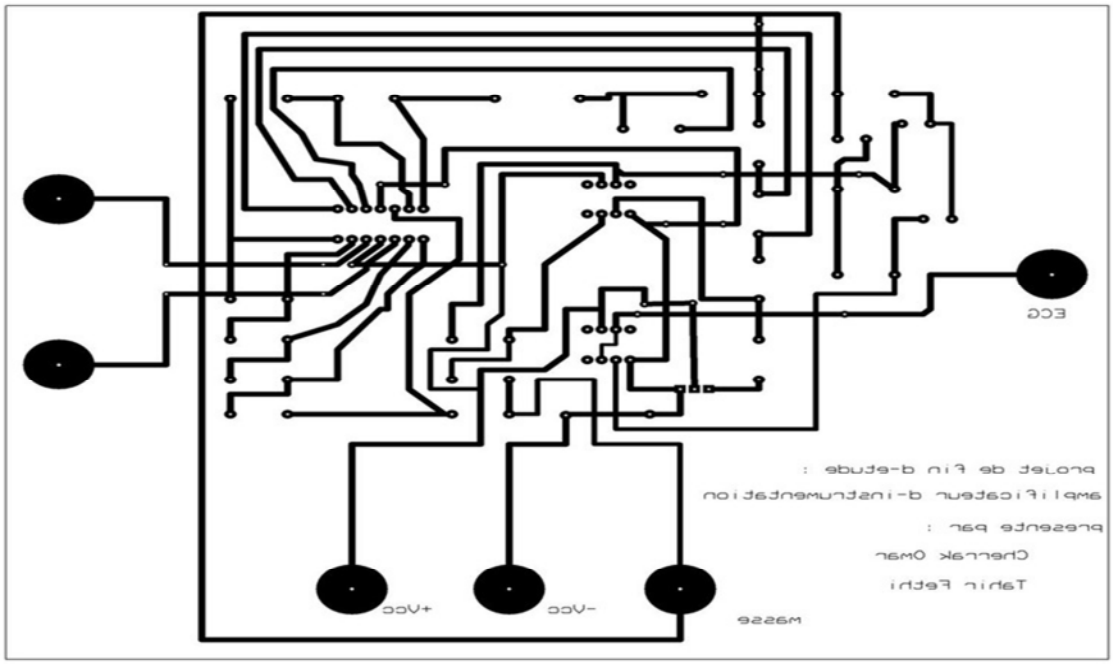


Figure IV. 18: Circuit imprimé d'ECG

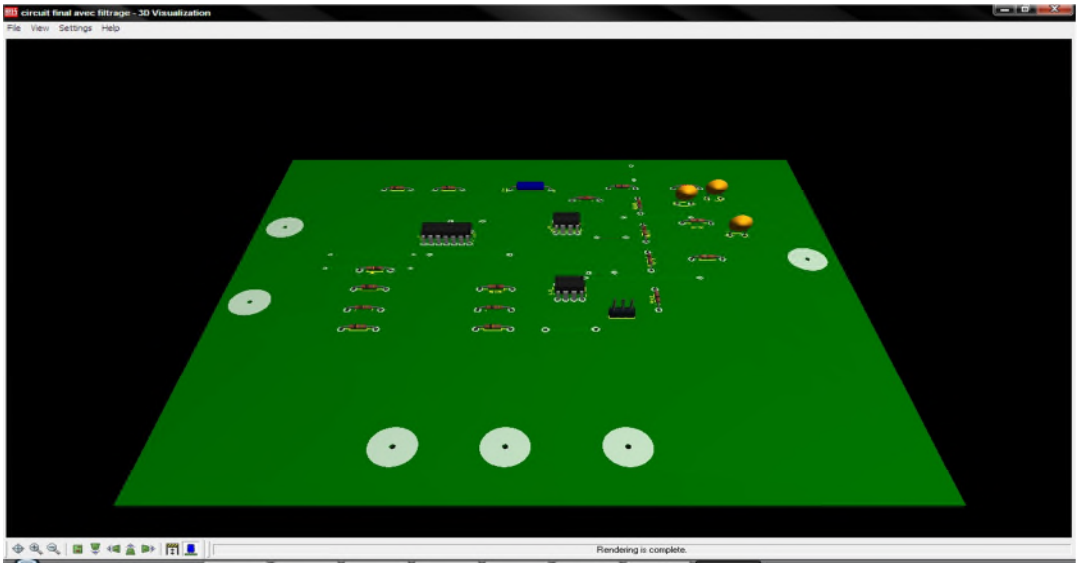


Figure IV. 19: Circuit réalisé 3D d'ECG

Réalisation pratique de la chaîne de mesure d'une vélocimétrie pariétale :

L'objectif de ce travail pratique est d'élaborer une méthode ultrasonore capable de visualisé le profil de vitesse de mouvance localisé des artères carotidiennes. Le schéma

synoptique suivant, représente notre réalisation pratique de la vélocimétrie ultrasonore pariétale.

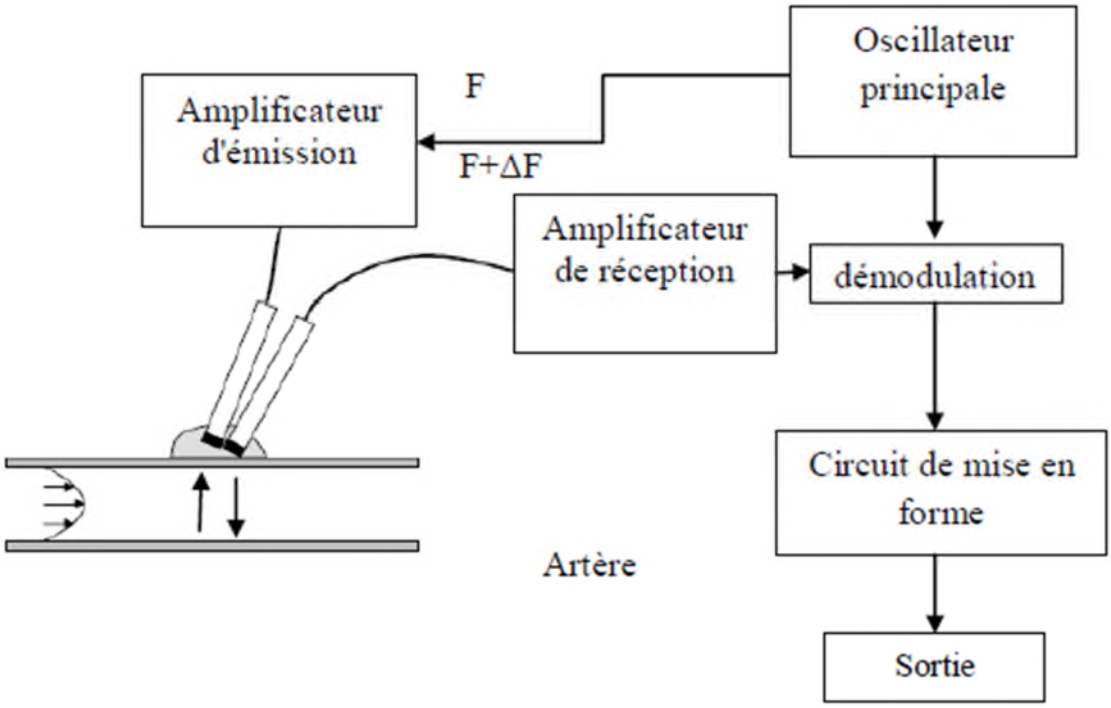


Figure IV. 20: Schéma synoptique de la vélocimétrie Doppler pariétale

Le signal recueilli par la sonde, de fréquence F +ΔF, est traité par le démodulateur afin d'extraire le signal Doppler, de fréquence ΔF.aprés a été filtré et amplifié. La figure III.6 représente le circuit électrique de la vélocimétrie ultrasonore.

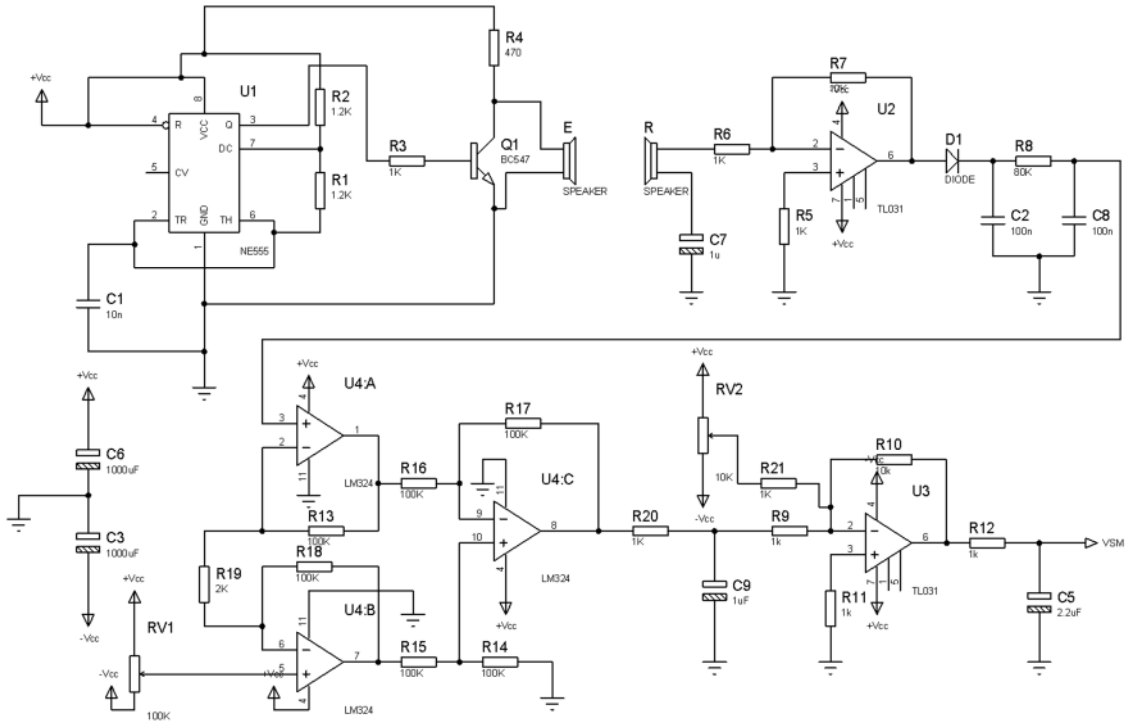


Figure IV. 21: Schéma électrique réalisé de la vélocimétrie Doppler pariétale

Le premier étage représente un oscillateur conçu avec le circuit intégré NE555 alimenté en 12V, qui permet dans une configuration donnée (voir le schéma Figure III.6) de fournir un signal sinusoïdale de fréquence 40KHz, ce dernier va être transformé en une onde mécanique (ultrasonore US) à travers un émetteur ultrasonore **E (figure III.7**) de type piézoélectrique, fonctionnant aux alentours de la fréquence de résonance mécanique de 40kHz. La vitesse et l'atténuation de cette onde US sont reliées aux paramètres physiques du milieu qu'elle traverse (l'air dans notre cas). Le transistor Q1(BC547) est utilisé comme un amplificateur de puissance.

Le signal d'écho reçu au transducteur peut être vu comme étant une onde porteuse modulée en fréquence et en amplitude par les échos provenant des mouvements de la paroi artérielle transmise localement à la peau. Le récepteur ultrasonore US **R (Figure III.7**) va capter les ondes US avec une petite modification de fréquence ΔF et d'amplitude ΔA.

Nous optons pour une démodulation d'amplitude de l'onde porteuse réfléchie par la peau. Pour cela, on utilise un détecteur linéaire d'enveloppe composé de

l'amplificateur **U2** de type **TL031** monté en amplificateur inverseur avec gain $A_{U2}=$

11, de la diode **D1, de** la résistance **R8** et des condensateurs **C2,**

C8 représentant la détecteur d'enveloppe pour extraire notre information ΔA. Un deuxième étage d'amplification est nécessaire pour bien visualiser l'information ΔA en utilisant un LM324 ; RV1 est utilisé pour régler la tension de référence par apport à la tension d'entrée ΔA. Un dernier étage d'amplification et de filtrage, avec **U3** de

type **TL031** du gain $A_{U3} = 11$ à été utilisé.

En outre, pour éliminer tous signaux parasites, on a décidé de filtrer le signal amplifié avec un simple pont R-C passe bas, dont la fréquence de coupure, $FC = 72{,}5Hz$

A la sortie de ce montage nous avons récupéré le signal de la vélocité pariétale, d'une amplitude de l'ordre de **4v.**

La compensation externe consiste à générer une tension externe avec la même valeur mais de signe opposé à la tension d'offset globale du montage. Cette tension générée sera appliquée à l'une des entrées de l'A.OP. La tension variable est obtenu grâce au potentiomètre connecté entre +Vcc et –Vcc et l'une des entrées de l'A.OP. Donc les potentiomètres sont utilisés pour la compensation de tension.

La figure IV.22 représente les émetteurs récepteurs utilisés.

Figure IV. 22: Transducteur ultrasonore

Le temporisateur NE555 (Timer 555) :

Le temporisateur NE555 est un circuit intégré de 8 broches spécialisé dans la production de signaux **Figure III.8** (pour la configuration interne de ce circuit intégré voir l'**annexe2**). Il peut

être facilement configuré (avec des composants externes) pour fonctionner en mode monostable ou astable dont la fréquence d'oscillation est facilement ajustable, par des composants externes. Afin d'être très exactement à la fréquence de résonance de la paire des transducteurs.

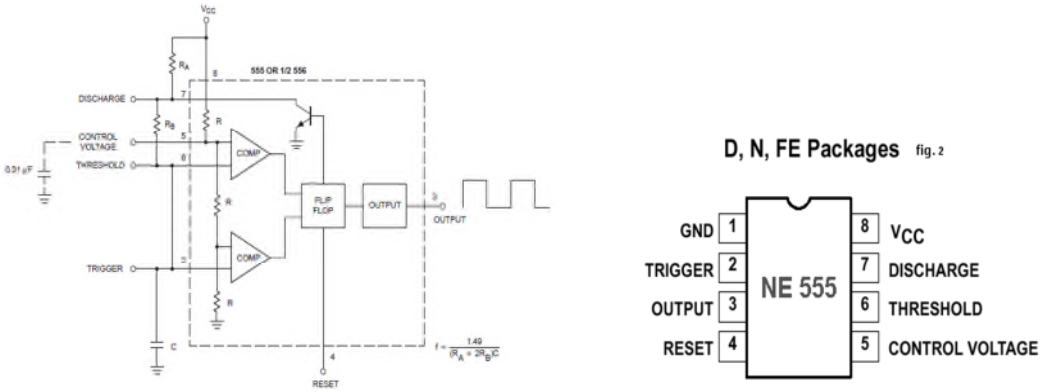


Figure IV. 23: Schéma de câblage du NE555 de Philips Semiconductors

Les figure III.9 et III.10 représentants respectivement le circuit imprimé et la visualisation 3D de ce montage.

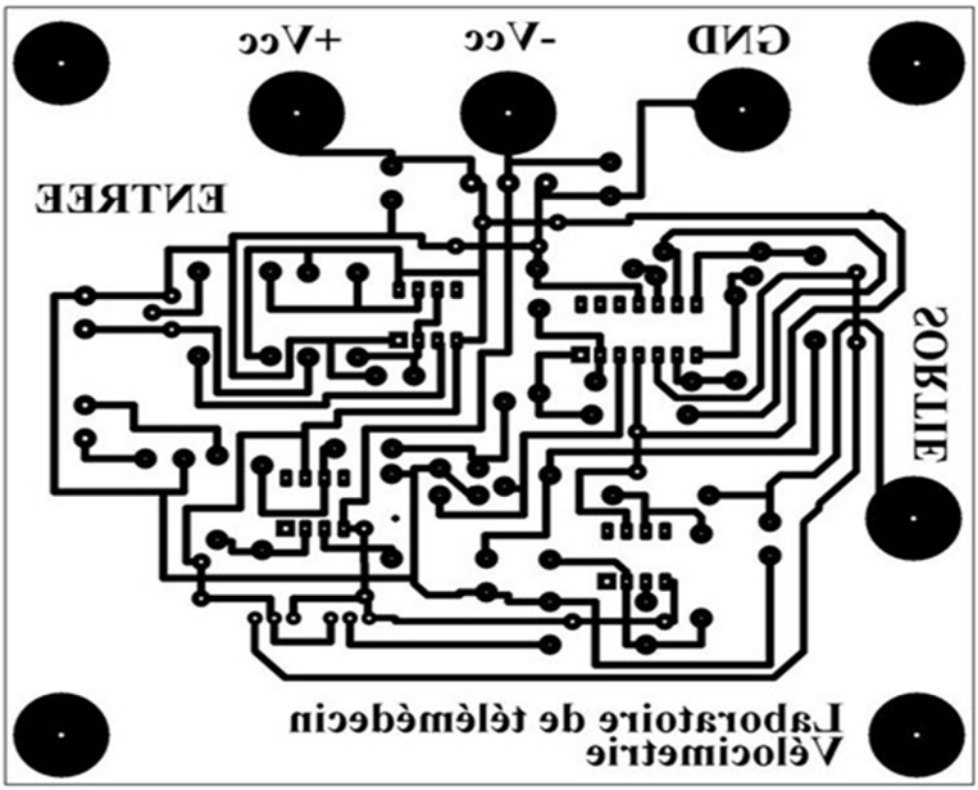


Figure IV. 24: Circuit imprimé réalisé de la vélocimétrie Doppler

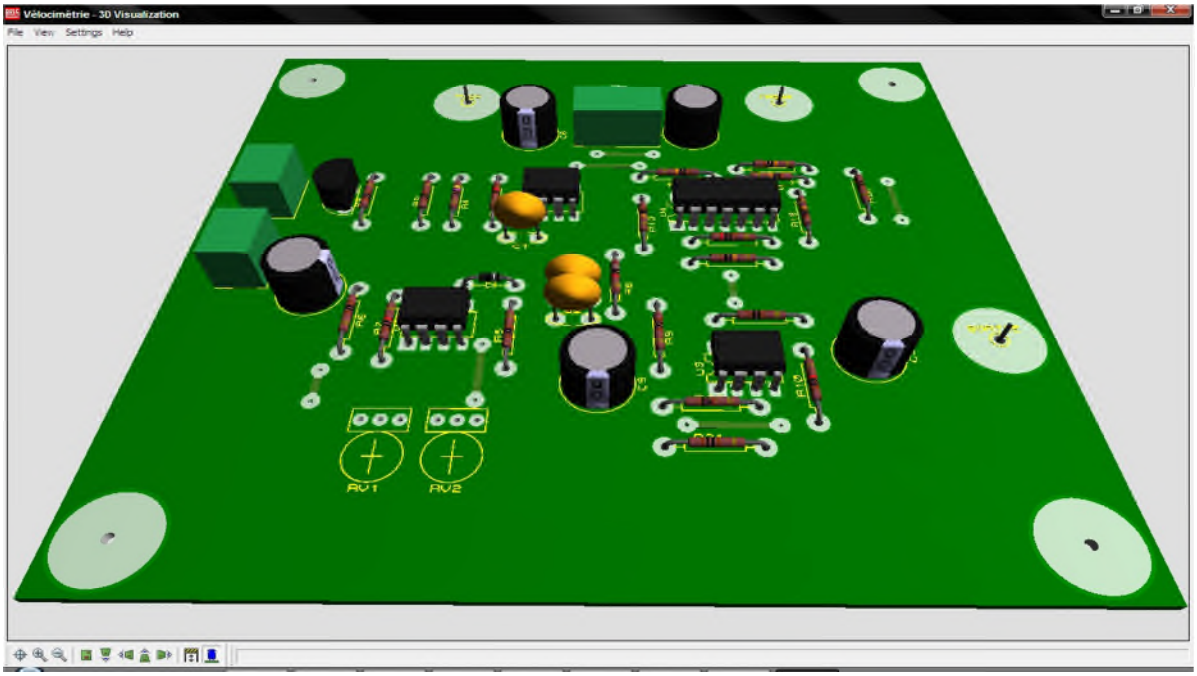
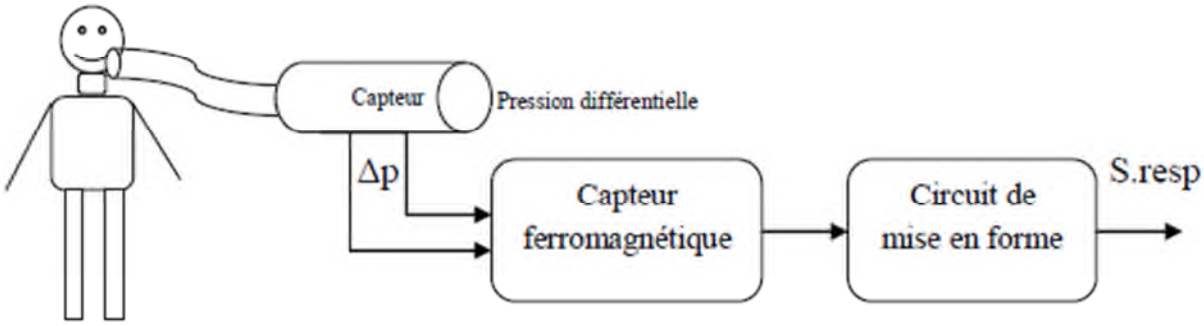


Figure IV. 25: 10 image 3D de la mise en forme d'une vélocimétrie Doppler

Réalisation pratique de la chaîne de mesure du signal respiratoire :

La figure IV.26 représente le schéma de principe d'un capteur de pression différentielle à reluctance variable.



La figure IV.27 représente le circuit électrique de mise en forme du signal respirographique.

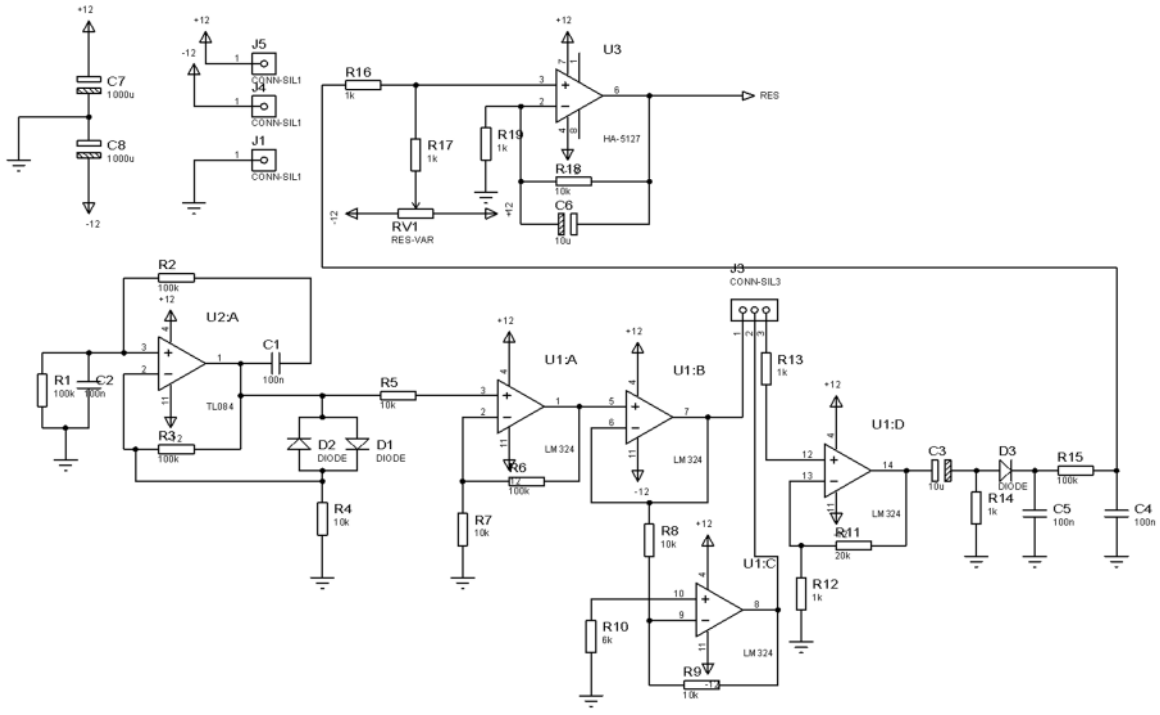


Figure IV. 26: Circuit électrique d'un respirateur à pression différentielle

Le premier étage de ce circuit conçu avec l'**U2:A** (TL084) (**Annexe3**), représente un oscillateur pour générer les deux signaux d'excitation. Le type d'oscillateur utilisé dans ce schéma est un oscillateur à pont de Wien, Cet oscillateur, va nous permettre d'appliquer une méthode d'approche générale pour les oscillateurs de ce type. Nous allons tout d'abord faire apparaître la structure générale d'un oscillateur quasi- sinusoïdal en identifiant l'amplificateur et le filtre sélectif. Ceci étant fait, nous verrons la condition à vérifier pour que les oscillations apparaissent. Nous pourrons alors calculer les principales grandeurs attendues (fréquence et amplitude des oscillations notamment).

Structure et identification des différents éléments :

Le circuit d'oscillateur à pont de Wien se présente sous la forme suivante :

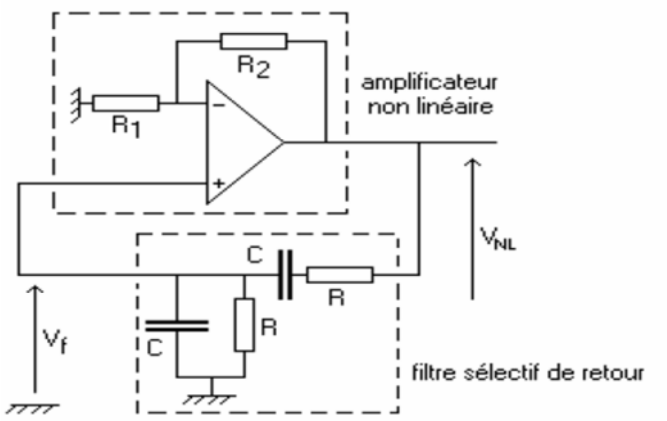


Figure IV. 27: Schéma électrique d'oscillateur à pont de Wien

Dans sa zone de fonctionnement linéaire, l'amplificateur a un gain $A=1+R2/R1$ (pour l'étude du démarrage, ce gain sera suffisant), pourvu que l'oscillation se fasse dans la

bande passante de l'amplificateur. Cependant la tension de sortie de l'amplificateur est limitée à la plage [-Vcc ,+Vcc]. Sa caractéristique entrée-sortie, si on suppose l'amplificateur opérationnel parfait (excepté vis à vis de la saturation) est donc la suivante:

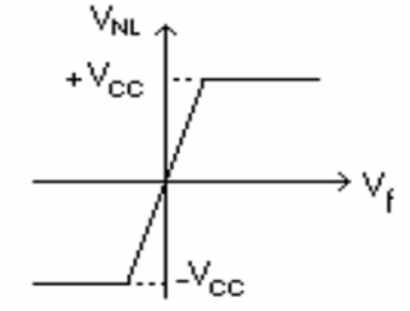


Figure IV. 28: 14 Caractéristique entrée-sortie d'amplificateur

Condition de démarrage des oscillations :

Un système bouclé du type décrit dans notre exemple est instable lorsque l'un des pôles de sa fonction de transfert en boucle fermée a une partie réelle positive. Ces pôles sont les solutions de l'équation

$$A\,B(p) = -1$$

Dans notre travaille pratique nous avons vu à la sortie de l'oscillateur un signal sinusal de fréquence **F=1KHz**, et d'amplitude **A=1V** crêt à crêt. Nous désirons amplifier ce signal entre - *12V* et *+ 12V*, pour pouvoir exciter les matériaux ferromagnétique. Pour ce faire, nous utilisons un amplificateur LM324 **U1.A** du gain

Donc nous avons vu à la sortie du cette ampli un signal sinusoïdale de fréquence **F= 1KHz** et **d'amplitude A= 12V**.

Pour limiter l'amplitude, on peut introduire une non-linéarité au moyen de deux diodes **D1**et **D2** tête-bêche branchées en parallèle avec R3, Les diodes réduisent le gain dés que la valeur instantanée du signal aux bornes de R3 dépasse 0,6 V.

U1.B et **U1.C** sont réalisés en suiveur et en inverseur pour rendre les deux signaux opposé en phase, au niveau de deux sorties 1et 2 de **J3.**

Les deux sorties 1et 2 sont reliées avec les deux têtes de capteur ferromagnétique. Nous avons ramené le signal à travers l'entrée 3 de **J3** avec une faible modification d'amplitude **ΔA**, représentant le signal respiratoire. L'amplificateur **U1.D** est réalisé

avec un gain $A_{U1.D}=21$, pour amplifier le signal recueilli par le

capteur et portant l'information **ΔA**. La résistance R15 et les condensateurs C4 etC5 représentent le détecteur d'enveloppe pour extraire **ΔA.** Le dernier étage représente un

filtre actif du premier ordre de gain $A_{U3}=11$, utilisé pour amplifier et bien filtrer l'information **ΔA.** Le potentiomètre RV1 est utilisé pour régler la composante continue. Après le traitement analogique nous avons visualisé le signal représentatif de la fonction respiratoire **ΔA** de l'ordre de grandeur de $1V$ en sortie.

Les figure IV.30 et IV.31 représentants respectivement le circuit imprimé et la visualisation 3D de ce montage.

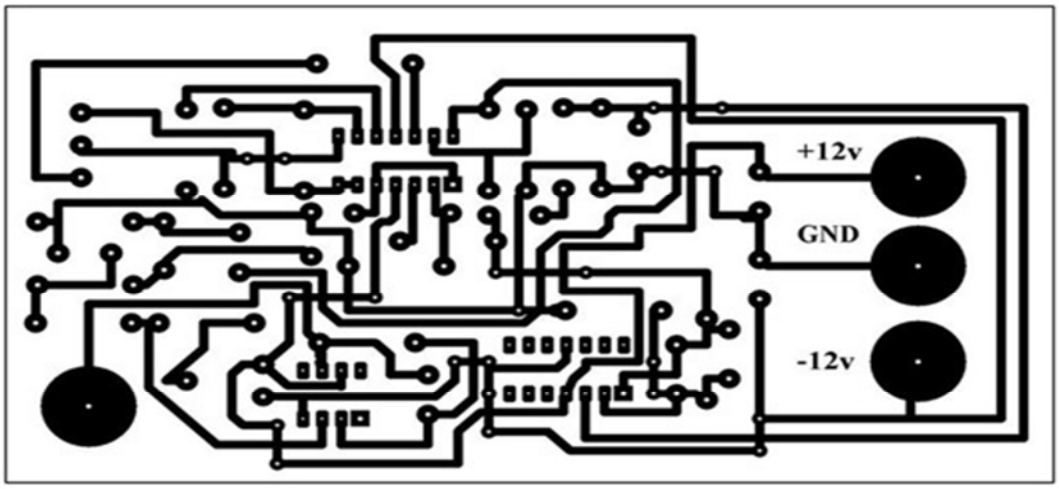


Figure IV. 29: Circuit imprimé de la mise en forme d'un respirateur

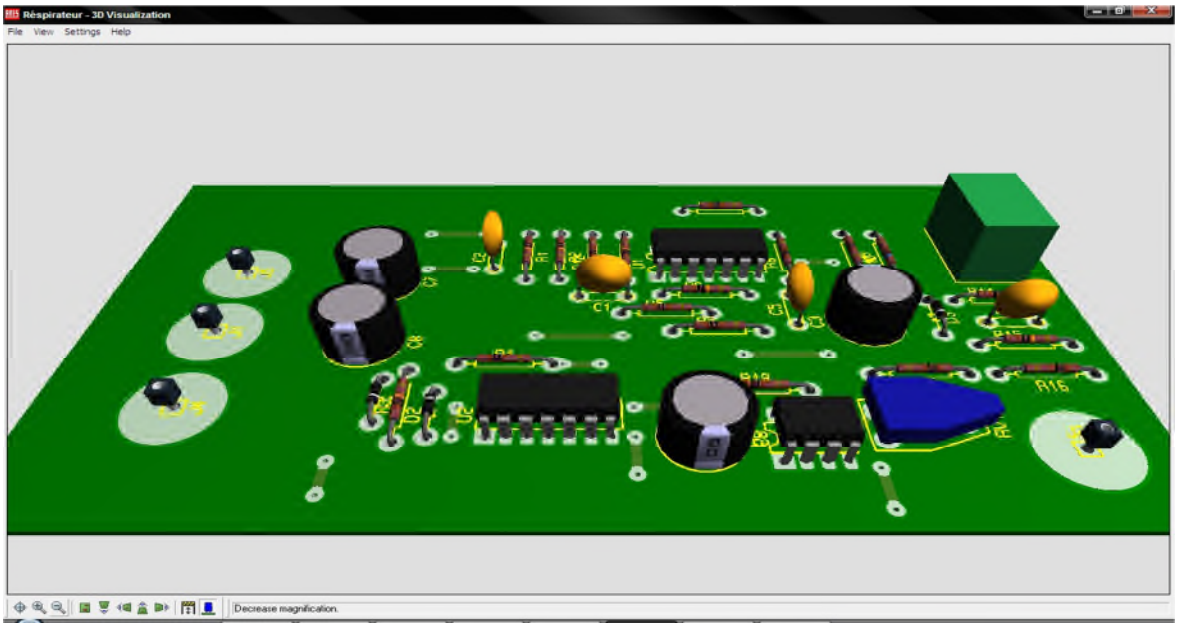


Figure IV. 30: Image 3D de la mise en forme d'un respirateur

Réalisation pratique de la chaîne de mesure photopléthysmographiques:

Aujourd'hui, les photopléthysmographes de façon générale existent dans chaque unité de soins intensifs, bloc chirurgical, et dans beaucoup de salles de secours. Ils sont conçus globalement en utilisant le schéma bloc ci-dessous.

Il est constitué des parties suivantes:

- La partie sonde regroupe la source lumineuse et le photodétecteur.
- Conditionnement du signal.
- Circuit de mise enforme.

La conception se présente alors selon le schéma bloc de la figure IV.32 :

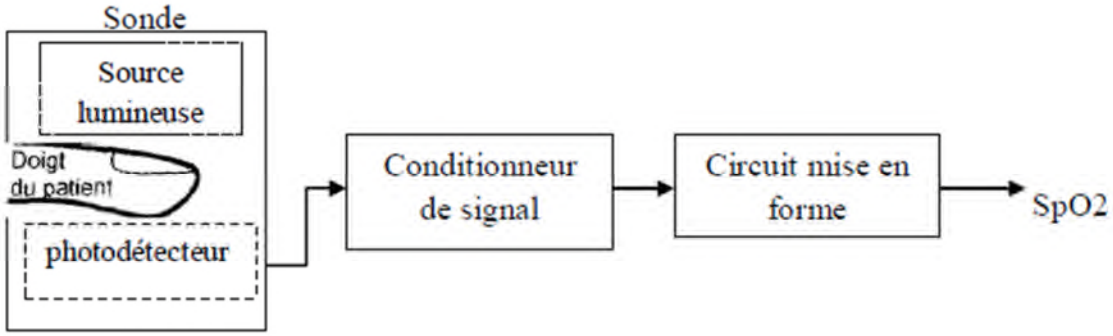


Figure IV. 31: Schéma bloc général d'un photopléthysmographe

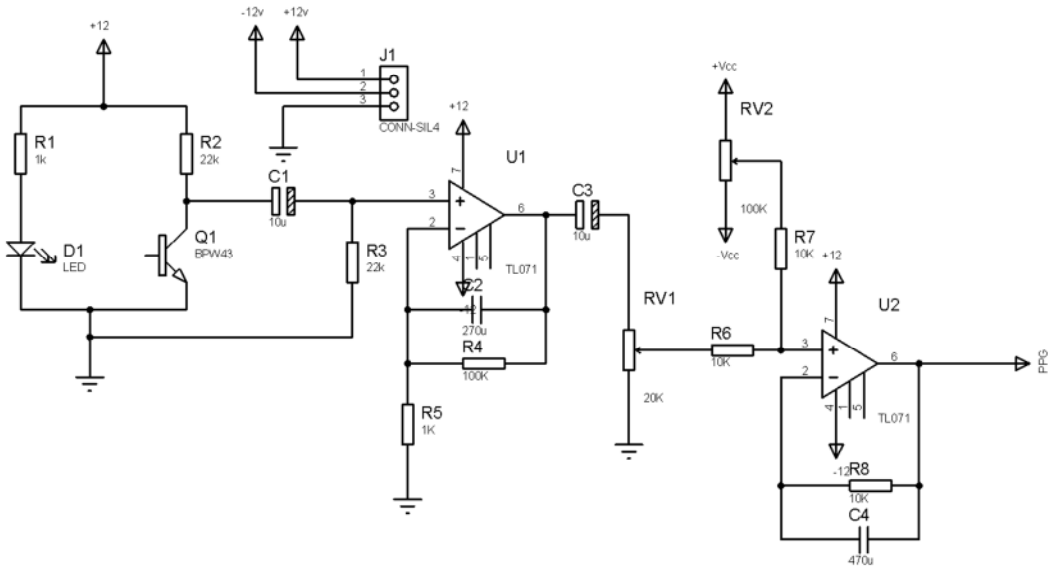


Figure IV. 32: Schéma électrique réalisé du photopléthysmographe

La figure III.17 représente le schéma électrique réalisé d'un photopléthysmographe PPG. Une LED est une semi-conductrice optoélectronique qui produit la lumière par électroluminescence. Les LED sont caractérisées par une grande efficacité luminescente comparée à d'autres méthodes d'émission légère telles que la cathode, la température, et la photoluminescence. Le LED que nous avons utilisé est une source de lumière infrarouge IR. Un capteur phototransistor **Q1** détecte les variations d'intensité lumineuse. Un système électronique permet d'amplifier et filtrer ces variations.

Le gain du premier ampli est égal à 100. Cela permet d'amplifier les signaux de faible amplitude de l'ordre de quelques millivolts issus de phototransistor. Le signal sera appliqué à un deuxième ampli. Op lié à un potentiomètre RV1 permettant de contrôler son gain.

Le second potentiomètre RV2 est destiné à régler la composante continue et de l'éliminer s'il est nécessaire. Le signal de sortie est de l'ordre de **1V.**

Les figures III.18 et III.19 représentants respectivement le circuit imprimé et la visualisation 3D, de la réalisation pratique du notre carte PPG.

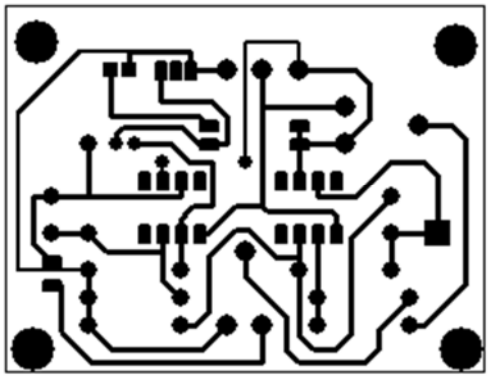


Figure IV. 33: Circuit imprimé de circuit de mise en forme PPG

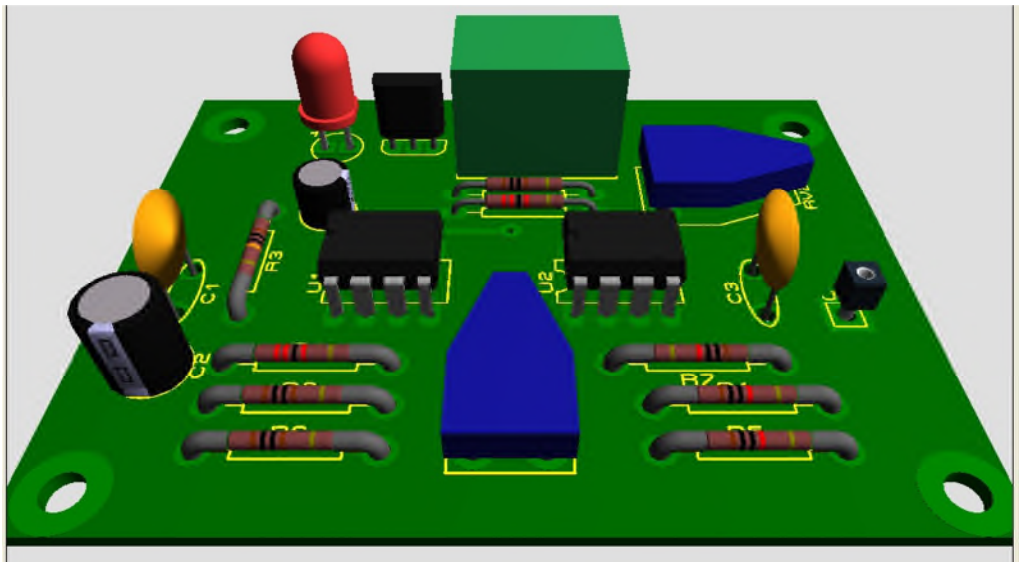


Figure IV. 34: Image 3D de circuit de mise en forme PPG

La figure III.20 représente le schéma bloc global de notre réalisation pratique. L'implantation électronique de ce schéma est représentée par la (figure III.21), les quatre sorties de ce circuit sont reliées avec la carte d'acquisition qui représente l'interface homme machine utilisant le microcontrôleur de Microchip faisant l'objet du chapitre suivant.

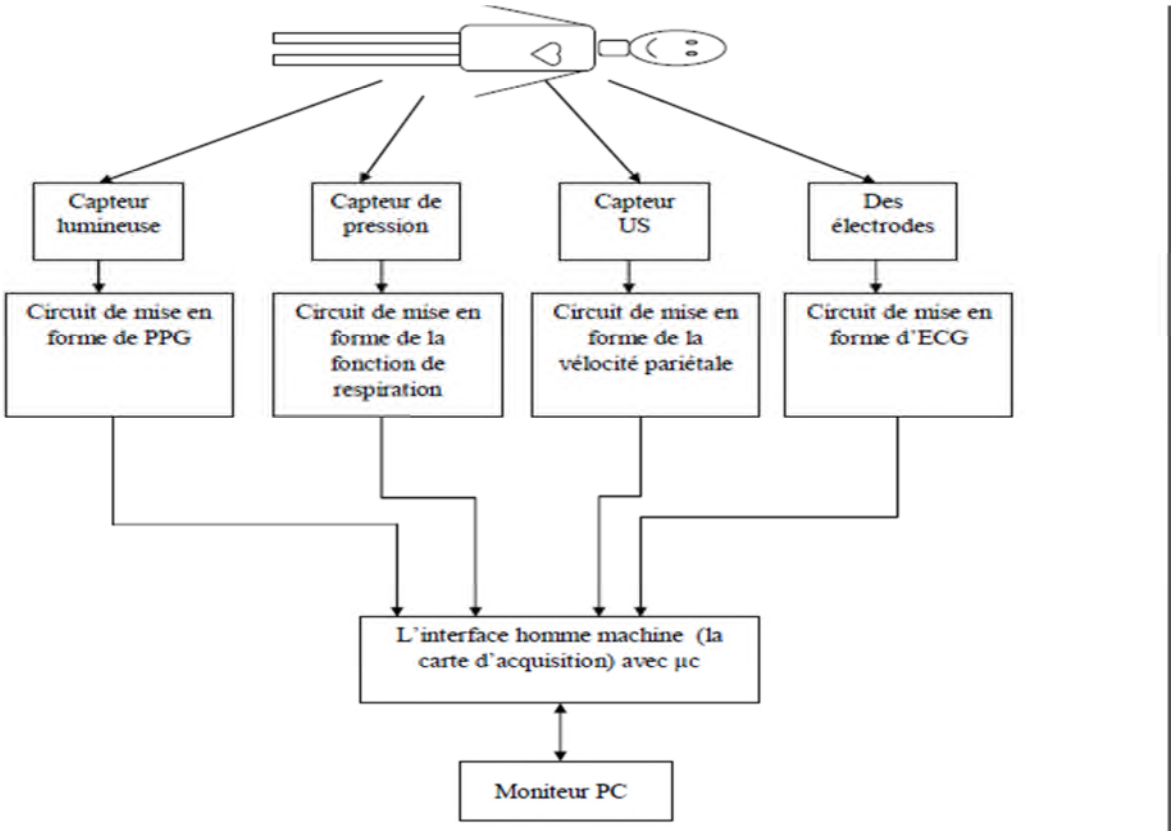


Figure IV. 35: Schéma bloc globale de la réalisation pratique

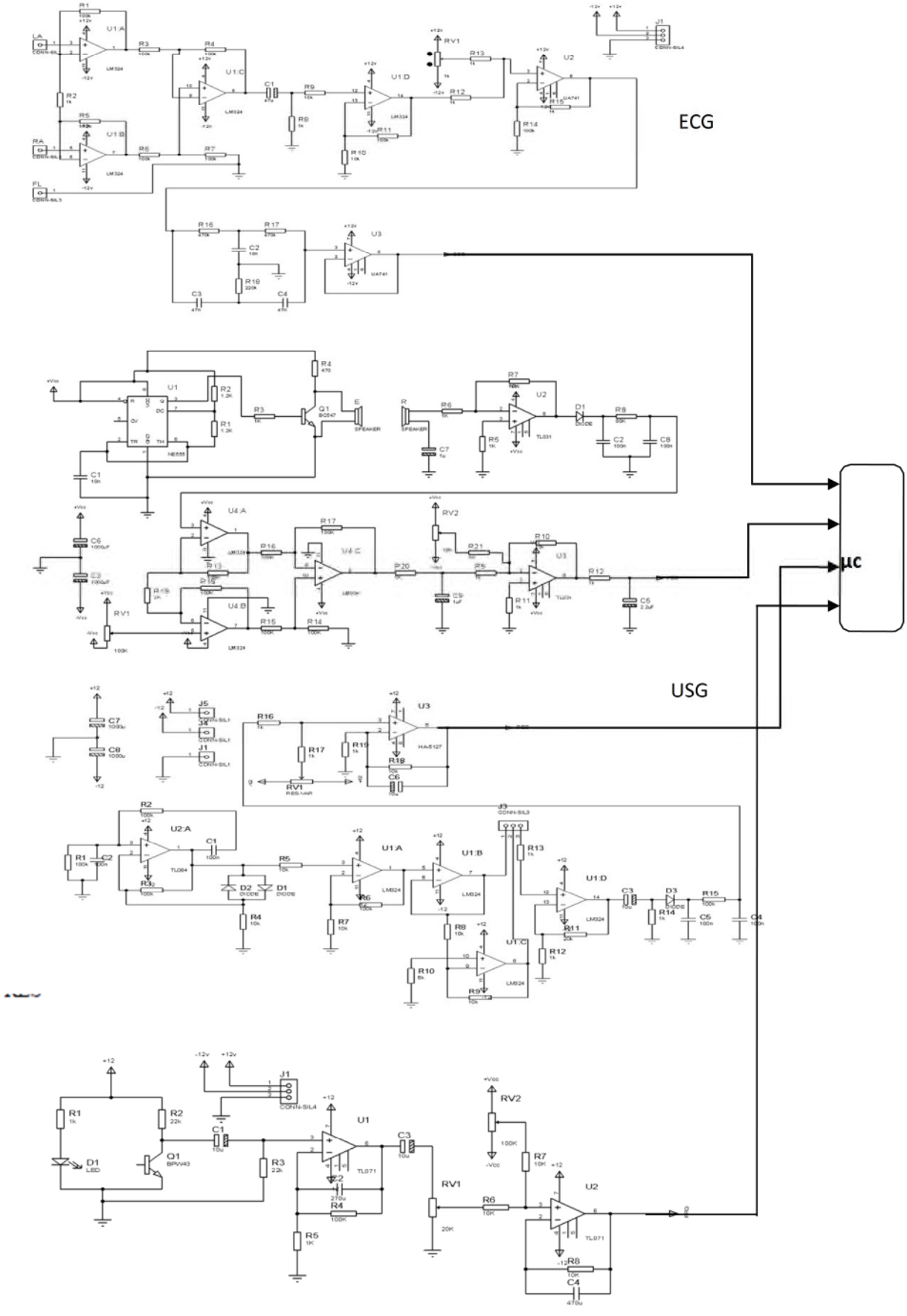


Figure IV. 36: Schéma électrique complet du système réalisé

PPG

Conclusion :

Ce chapitre a été consacré essentiellement à la réalisation pratique des différents circuits permettant le recueil des signaux dédiés à l'exploration cardiovasculaire c'est- à-dire du plateau technique faisant l'objet de notre mémoire. Ces différents signaux devant parvenir à un terminal local après numérisation pour être visualisés, traités, analysés, archivés, nous consacrons le chapitre suivant à l'implantation de l'interface Hardware et Software dédiée.

Chapitre 5 : IMPLEMENTATION DE LA PLATEFORME TELE MEDICALE SOUS ARCHITECTURE DE COMMUNICATION LOCALE USB-HID

Résumé— La pratique télé médicale consiste à prélever sur le patient des informations multidimensionnelles et multi médias représentatives de son état physiopathologique, de les faire parvenir à un terminal informatique local dans un premier temps puis à un terminal informatique distant dans un deuxième temps.

Nous nous proposons dans le cadre de cet article de mettre en œuvre :

La réalisation d'une plateforme télé médicale non invasive et non intrusive dédiée à l'acquisition simultanée et en temps réel de jusqu'à 16 signaux physiologiques unidimensionnels.

La réalisation d'une interface hardware construite autour des C.A.N. séries AD0830 d'Analog Device assurant la numérisation des signaux et du microcontrôleur 18F2550 de Microchip assurant leurs transferts du patient vers la poste local sous protocole USB avec les spécifications HID.

La réalisation d'une interface graphique sous environnement Visual Basic mettant à profit le composant Winsock compatible avec le système d'exploitation Windows et l'architecture client-serveur permettant la connexion des différents terminaux informatiques pour le transfert des données à travers les réseaux télé médicaux sous protocole TCP-IP.

Mots clés— Télémédecine, USB, HID, mcHID.dll, TCP-IP, Microcontrôleur, firmware, VB6, socket.

I INTRODUCTION [1]-[11]-[14]-[15]-[18]

La pratique médico chirurgicale (Diagnostic, thérapeutique, surveillance) met en jeu l'utilisation d'une panoplie de plateaux techniques. La puissance actuelle des techniques de l'information et de la communication nous a permis de penser à l'intégration progressive de cette multitude de plateaux techniques sous forme d'une plateforme Télé médicale polyvalente, adaptative et évolutive qui transformerait un terminal informatique en une véritable station de pratique médico chirurgicale **locale** ou **distante** et qui aboutirait à terme à la matérialisation des notions de **Télé hôpital**, **d'Habitat Intelligent pour la santé (H.I.S)**, de **Domiciliation Médicalisée** des patients atteins de maladies chroniques ou de **Dossier Médical Personnalisé Partagé (DMP)**. Cet engouement est souvent représenté par le terme **e-s@nté** sur le plan structurel et **Télé médecine** sur le plan fonctionnel.

L'originalité du dispositif que nous avons développé réside dans faits suivants :

Le recours à des C.A.N. externes au microcontrôleur qui travaillent en même temps permet d'optimiser la vitesse de numérisation des différents signaux.

Le recueil simultané de plusieurs signaux représentatifs de fonctions physiologiques différentes permet d'établir leur inter corrélations moyennant l'implémentation d'algorithmes appropriés. Ce qui a pour conséquence de mieux poser les diagnostics et de mieux porter les indications thérapeutiques.

La calibration automatique de l'affichage en fonction du nombre de signaux présents à l'entrée du dispositif facilite leurs exploitations par les acteurs de la Télé médecine.

L'intégration de plusieurs examens complémentaires dans un tel système permet d'éviter le recours systématique à l'orientation du patient vers différents services spécialisés et de réaliser ainsi un gain de temps précieux pour le patient, pour le médecin et pour le coût de la prise en charge médicale de ce dernier..

Et enfin Le caractère embarqué de la plateforme télé médicale permet :

- La réalisation d'examens complémentaires quel que soit le site ou se trouve le patient et par conséquent une meilleure appréhension de la CAT urgente.

- La dispense des soins à tous: desserte beaucoup plus large dans les zones rurales et isolées.

Nous présentons dans le cadre de cet article :

1- l'implémentation hardware et software d'une plateforme micro contrôlée capable de prélever sur le patient des signaux unidimensionnelles, de les faire parvenir à un poste local sous l'architecture de communication USB-HID construite sous environnement MPLAB [2]-[4].

2- L'implémentation d'une application permettant l'affichage, l'archivage et le traitement numérique des différents signaux construite sous environnement Visual Basic (VB) et mettant à profit l'API (Application Programing Interface) **mcHID.dll** pour la norme USB-HID [5].

3- L'implémentation d'un réseau Télé médical pour le transfert des données sous protocole TCP-IP utilisant notamment le composant winsock de Visual Basic qui
permet la mise en œuvre de l'architecture Client-Serveur [6]-[8]

II STRUCTURE DE LA PLATEFORME TELE MEDICALE

Celle-ci comporte :

- Le patient source et destinataire de l'information médicale.

- Les ETTD ou DTE (Data Terminal Equipment) chargés de recueillir l'information sur le patient.

- Le CODEC (Codeur Décodeur) chargé de faire transiter l'information issus des DTE vers les terminaux informatiques locaux et inversement.

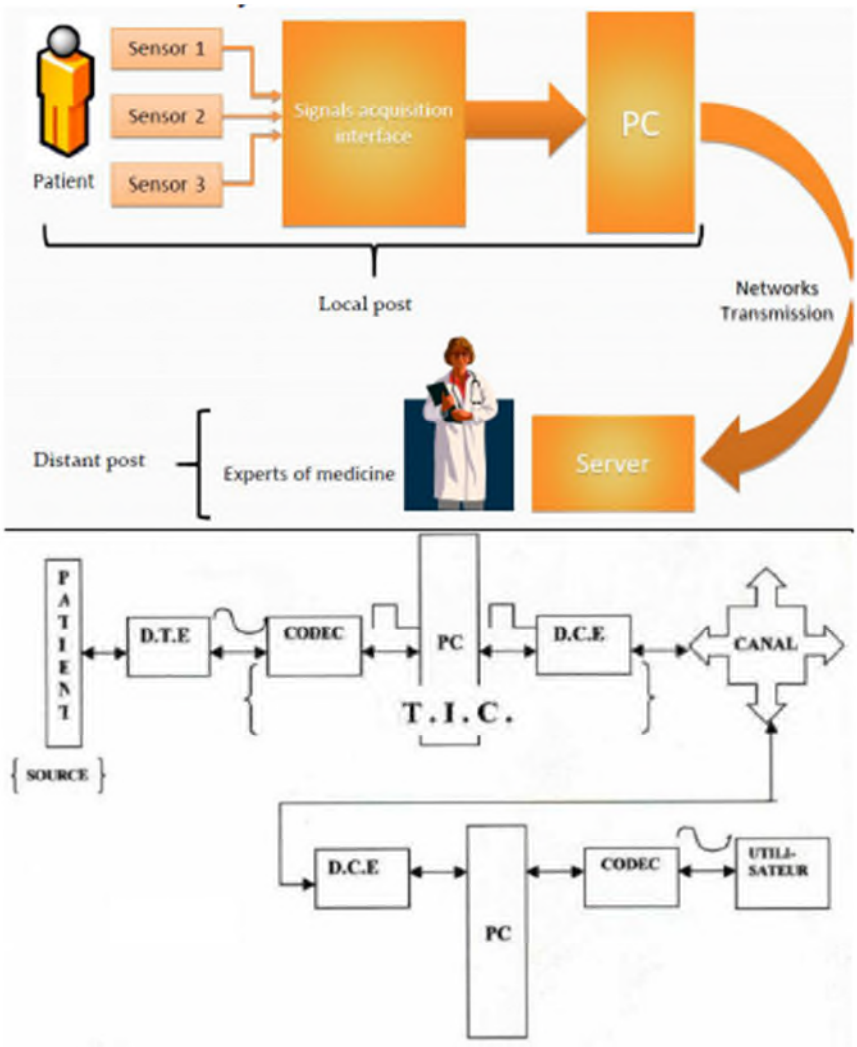


Figure V. 1: Structure de la plateforme Télé médicale

 Les PC (Terminaux informatiques) locaux ou distants chargés de présenter l'information médicale aux praticiens de la médecine sous forme exploitable et conviviale, de stocker ces informations et d'héberger les différentes applications et plateformes software de traitement numérique et de transfert de l'information médicale au moyen d'un environnement de programmation donné.

 Les ETCD ou DCE (Data Communication Equipment) chargés d'adapter le signal informationnel au canal de transmission, de transférer les données médicales vers des terminaux distants via les réseaux télé médicaux et de maximiser les débits au moyen des techniques du haut débit [9]-[10]-[13] .

III TECHNIQUES ET METHODES DEVELOPPEES SOUS PROTOCOLE USB.HID

La figure IV.2 montre le brochage des C.A.N avec le microcontrôleur. (Pour des raisons de clarté du schéma nous avons représenté uniquement 4 C.A.N.)

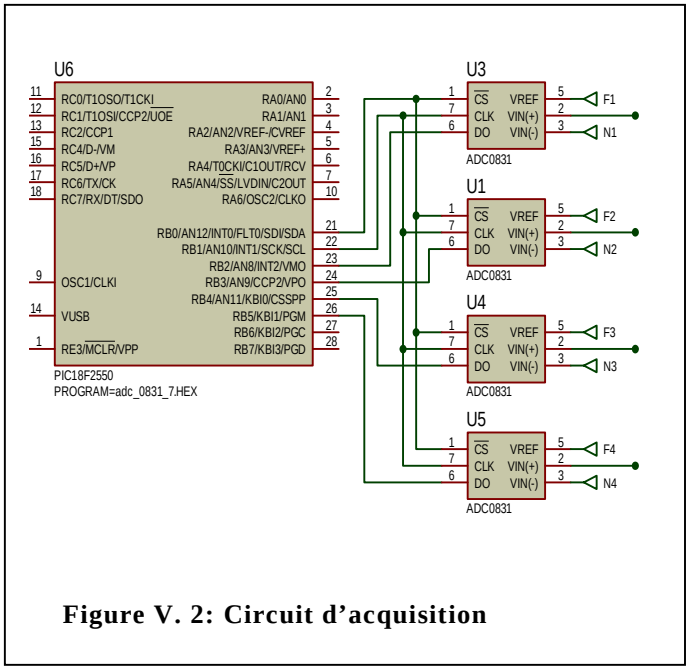


Figure V. 2: Circuit d'acquisition

Bien que le microcontrôleur possède un convertisseur A /D interne nous avons opté pour une solution à base de convertisseurs externes pour les raisons suivantes :

4 convertisseurs sont plus rapides qu'un seul : en effet si le choix s'est porté sur le convertisseur interne le temps de conversion global sera multiplié par 4 puisque le temps de ce convertisseur interne sera réparti entre les 4 entrées analogiques.

Avec ce procédé, même si on désire augmenter le nombre d'entrées analogiques le temps de conversion sera en fait celui d'un seul convertisseur puisque les convertisseurs travaillent en parallèle.

Durant la conversion le µC sera libre pour effectuer le travail de transfert des données vers et depuis l'hôte.

Comme le montre la figure2 le choix s'est porté sur les convertisseurs A /D de type ADC0831

Ce sont des convertisseurs à 8 bits avec sorties série. Le transfert des données avec le µC est de type synchrone via l'horloge CLK. Le bit le plus fort b7 MSB sera le premier à être transféré.

Le début de conversion commencera immédiatement après la sélection du composant et ce via le pin CS (chip select).

Le temps de conversion mentionné par le constructeur est de 32µs.

Par conséquent le temps total pour faire l'acquisition des 4 signaux analogiques est de 32µs.

III.1 VREF et VIN(-) :

Pour optimiser le processus de conversion il est nécessaire de connaitre dans quelle marge varie le signal à digitaliser. Pour ce faire le convertisseur ADC0831 dispose de 2 entrées noté VREF et VIN(-) pour permettre de mieux cadrer le signal.

A titre exemple supposons que notre signal fluctue entre les limites vmin = 1.5 v et vmax = 2.5 v voyons comment nous allons positionner les 2 tensions de références VREF et VIN(-) :

$$VIN(-) = vmin = 1.5\ v$$

$$VREF = vmax - vmin = 2.5 - 1.5 = 1\ v$$

La valeur n issue de la conversion du signal VIN(+) vaut

$$n = 255\frac{VIN(+)-VIN(-)}{VREF}$$

Nous proposons comme solution un simple diviseur de tension comme le montre la figure IV.3

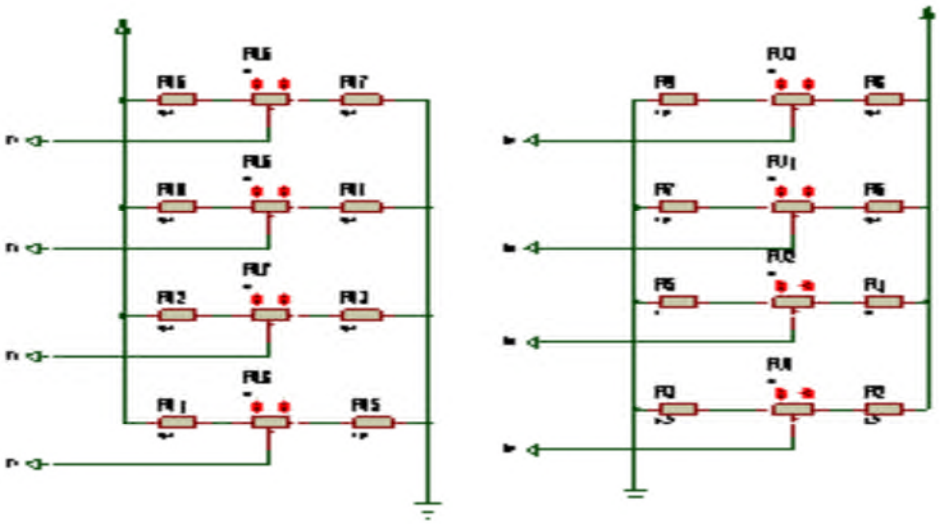


Figure V. 3: Circuit de cadre des entrées analogiques

La figure IV.4 montre la connexion de l'hôte aux périphériques (Capteurs biomédicaux)

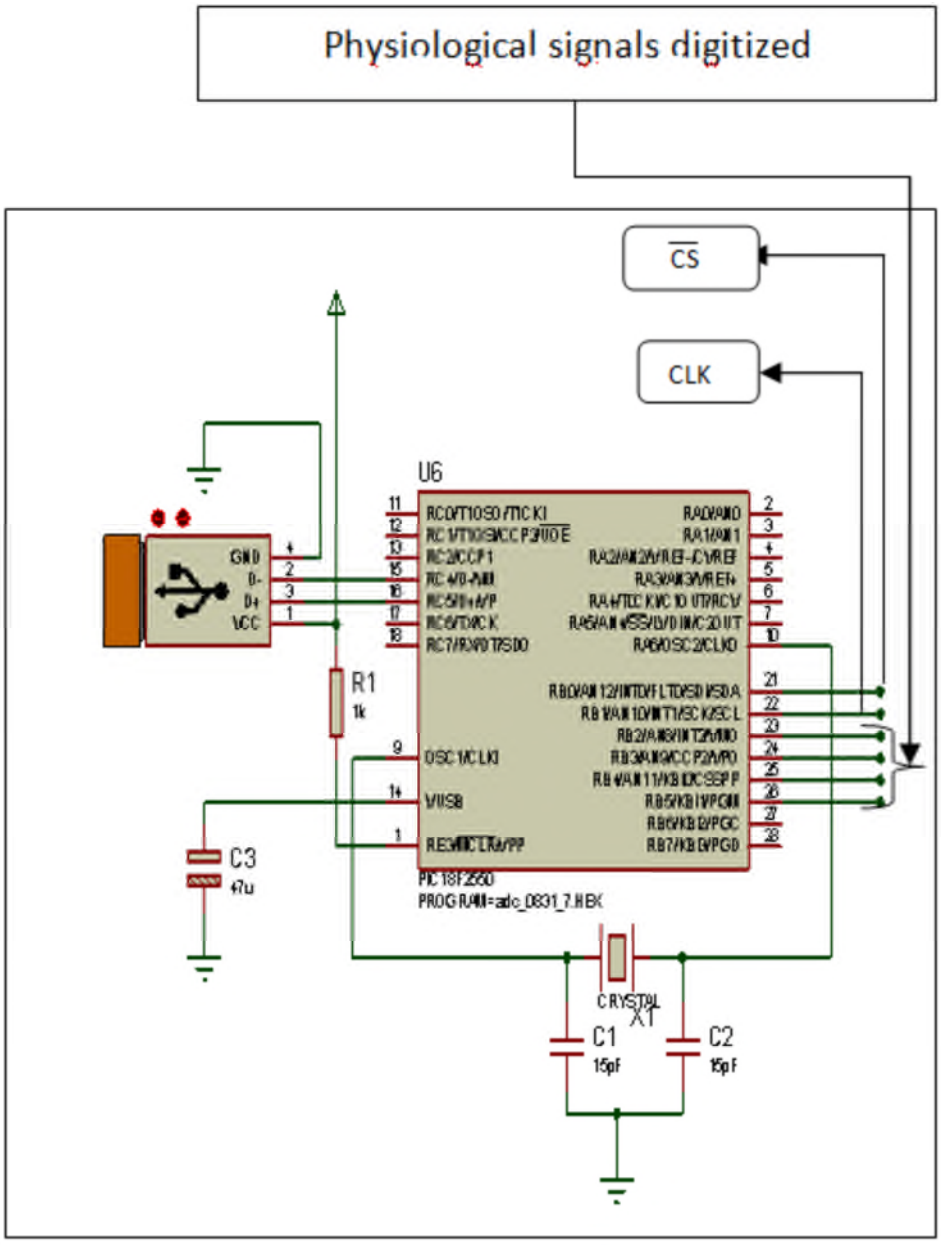


Figure V. 4: Connexion de l'hôte aux périphériques

III.2 DETAIL DES DESCRIPTEURS:

Descripteur de périphérique :

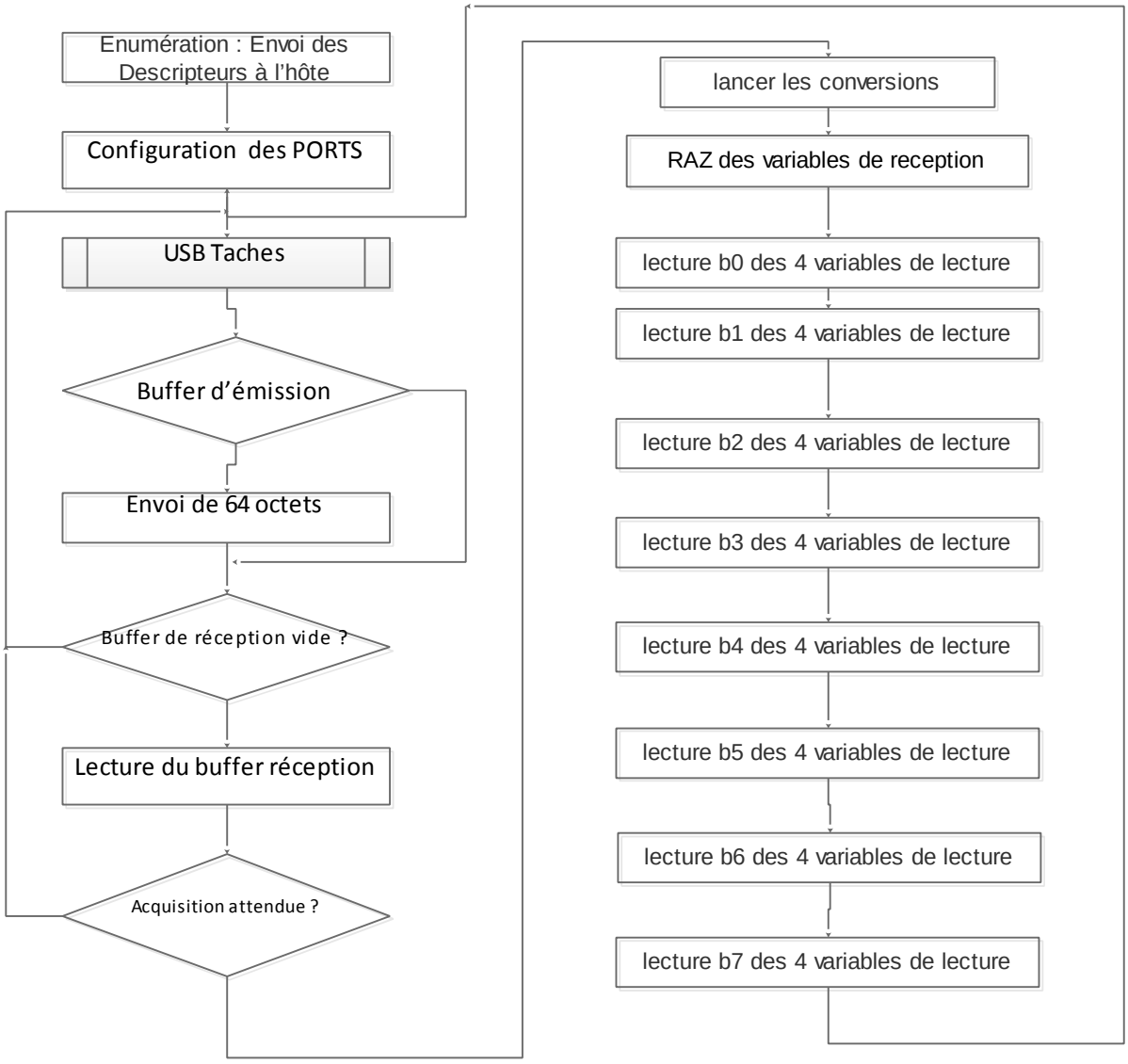
Nom	Nb octets	Valeur (hexa)
Longueur	1	12
descriptorType	1	01
cdUSB	2	02 00
DeviceClass	1	00
DeviceSubClass	1	00

DeviceProtocol	1	00
MaxPacketSize0	1	20
IdVendor	2	14D8
IdProduct	2	0015
bcdDevice	2	00 01
Manufacturer	1	01
Iproduct	1	02
ISerialNumber	1	00
NumConfigurations	1	01

Descripteur HID:

Nom	Nb octets	Valeur (hexa)
Longueur	1	09
descriptorType	1	21
HID	2	01 01
countryCode	1	00
NumDescriptors	1	01
DescriptorType	1	22
DescriptorLength	2	00 2F

III.3 ORGANIGRAMME GLOBAL DE TRANSFERT DES DONNEES :



III.4 PROGRAMMATION HOTE SUR PC

III.4.1 **L'interface sur le PC**

Notre projet a pour rôle de visualiser en temps réel et simultanément jusqu'à 16 signaux signaux physiologiques afin d'en faire une exploitation télé médicale pratique.

Pour cela nous implémentons une interface graphique sur PC qui permet le pilotage de la carte àµ C.

Le PC envoie à la carte les ordres concernant les voies à convertir et autorise l'envoi des résultats de la conversion (Mise à profit de la dll.Windows mcHID.dll qui permet les échanges de données entre le pc et le µc) .Nous réservons dans la trame envoyées au µc un octet de sélection des voies.

Apres la réception des données issues du µc celles-ci sont ordonnées et traduites sous formes de tracés sur l'écran du PC local. La figure IV.5 montre une vue d'ensemble de l'interface hébergé par le PC local.

Figure V. 5: Interface sur PC

III.4.1.1 Partie *paramètre*

Cette partie contient quatre cases de commande d'entrée de canal 1 à canal 16 plus une échelle qui donne le choix afin de calibrer le tracé.

III.4.1.2 Partie valeur

Dans celle-ci on peut visualiser le nombre d'octets transmit au PC et les valeurs numériques de la conversion A/N pour chaque échantillon correspondant au canal sélectionné. Ces valeurs ont stockées sur un fichier.

III.4.1.3 Partie *tracé*

C'est la partie la plus importante dans l'interface. Elle permet de visualiser les signaux des canaux sélectionnés dans le menu paramètre. Dans le cas où plusieurs signaux doivent être tracés simultanément, l'espace graphique est partagé en conséquence.

Le menu principal offre les possibilités suivantes :

- Entrées analogiques : choix jusqu'à 16 entrées analogiques différentes.

- Visualisation : on peut en outre choisir de ne visualiser que certaines voies parmi celles choisies précédemment.

Suivant le nombre de voies sélectionnées le programme fait le calcul automatiquement de l'échelle convenable. Si l'on désire de ne voir qu'un seul tracé toute la surface du plan lui sera réservé. Par contre si le choix est fait sur 4 voies, alors l'espace réservé au tracé de chaque signal analogique sera divisé par 4 et l'origine du repère de chaque graphe sera déterminé en conséquence.

Dans tous les cas de figures, les données reçues via l'USB sont stockées dans des tableaux en mémoire et seront sauvegardées dans des fichiers séparés .Ceci permettra de les réafficher ou d'en l'analyse spatio-spectro-temporelle plus tard.

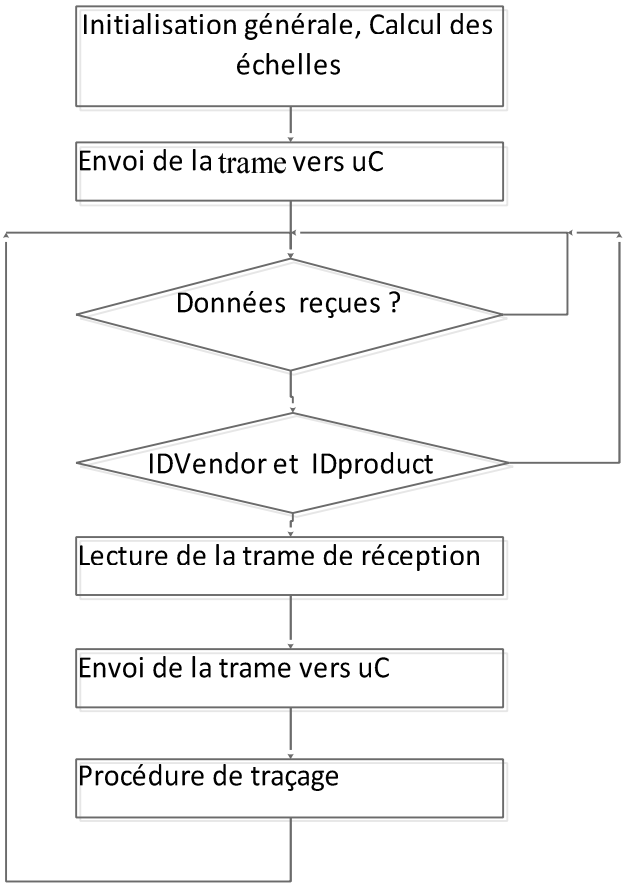
Vue de la trame de données reçue par le PC

Code	Octet1	Octet2	Octet3	Octet4	…	Octet7	Octet8

Vue de la trame de données envoyées par le PC

Code	Valeur1 : CAG	Valeur2 : canaux à traiter

III.4.2 **Organigramme sur PC**



IV RESULTATS SOUS PROTOCOLE USB-HID

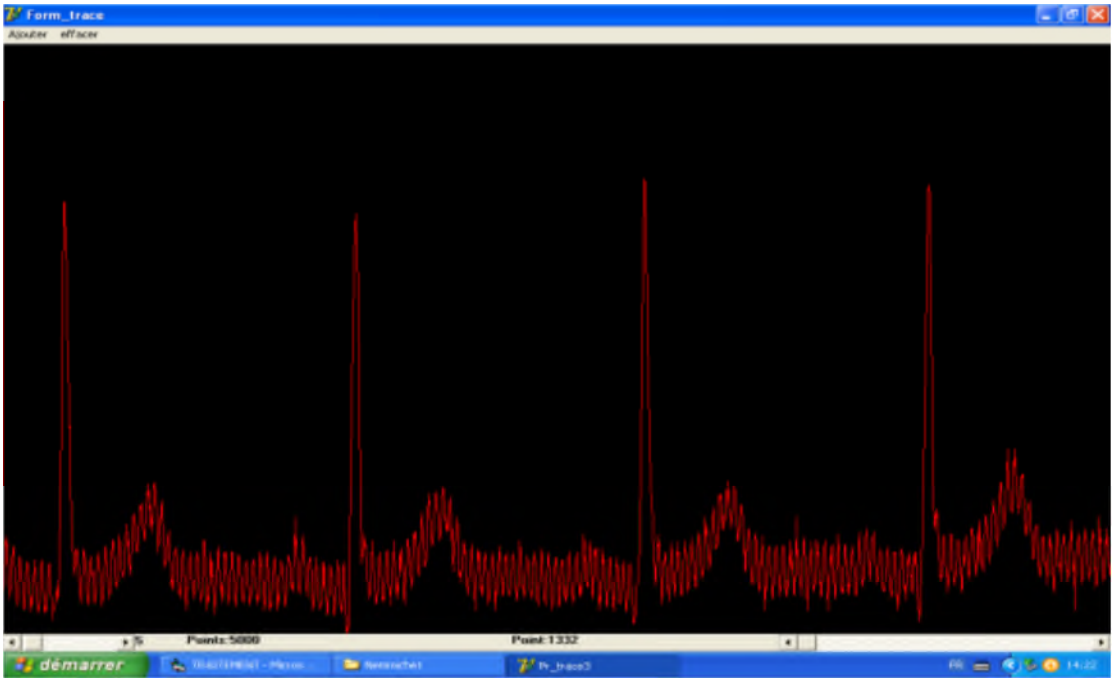


Figure V. 6: Affichage d'un seul signal

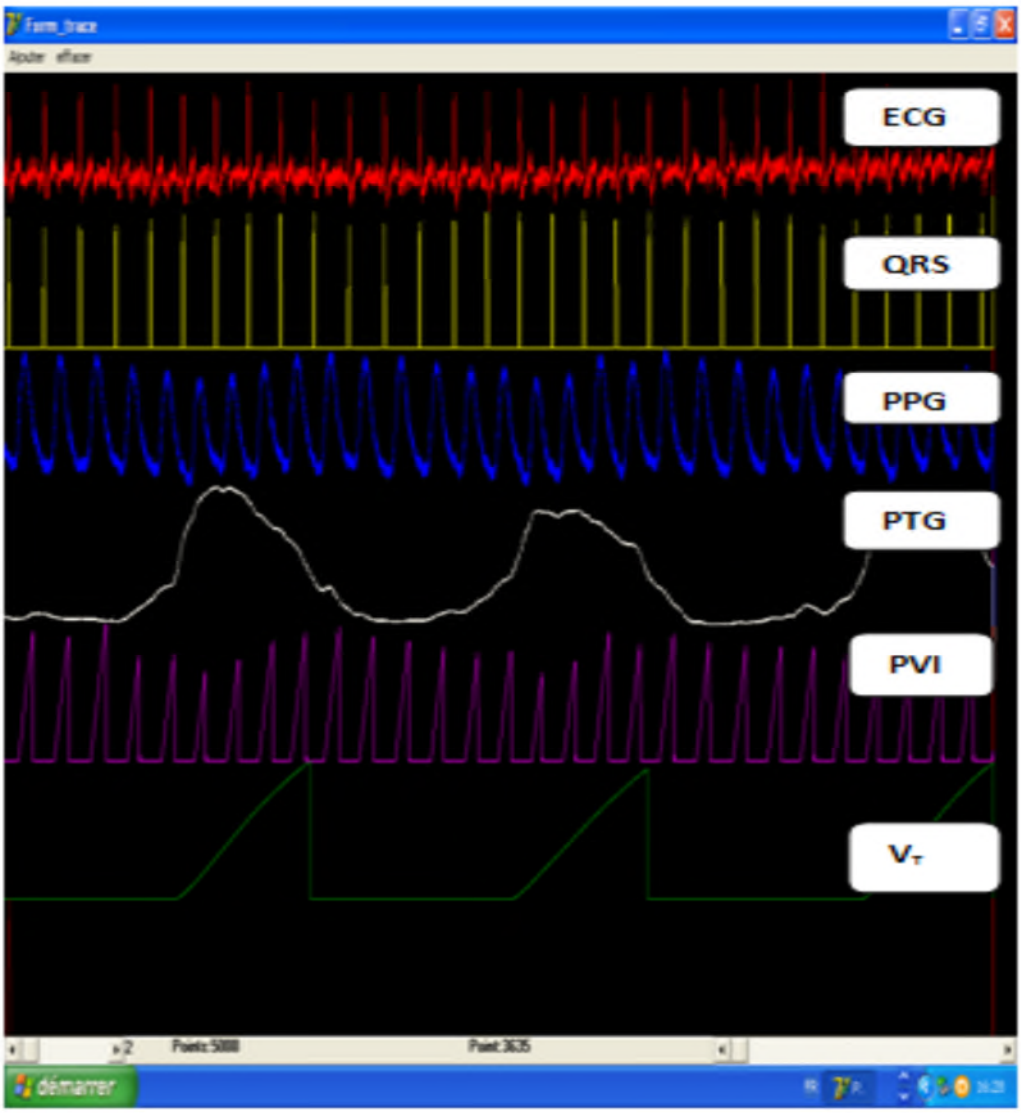


Figure V. 7: Affichage pour 6 signaux Physiologiques

l'électrocardiogramme **ECG** acquis par amplificateur d'instrumentation AD620 et le complexe **QRS** détecté.

Le photo pléthysmogramme **PPG** acquis par pectrophotométrie d'absorption moléculaire dans l'infrarouge représentatif de la concentration pulsée d'oxyhémoglobine **$HBO_{2[,}$** [17]

Le Pneumotachgramme **PTG** acquis par Pneumotachographie de Fleisch à reluctance variable et représentatif du débit ventilatoire [16],

L'indice de variation **Pleth PVI** obtenu par moyennage du PPG et représentatif de la variation respiratoire du PPG corrélée à la réponse au remplissage vasculaire **RRV**.

Le volume courant $\mathbf{V_t}$ (Tidal volume) obtenu par intégration du débit inspiratoire et représentatif de l'efficacité de la fonction musculaire ventilatoire qui pourrait être mis à profit au cours du télé sevrage de la ventilation artificielle.

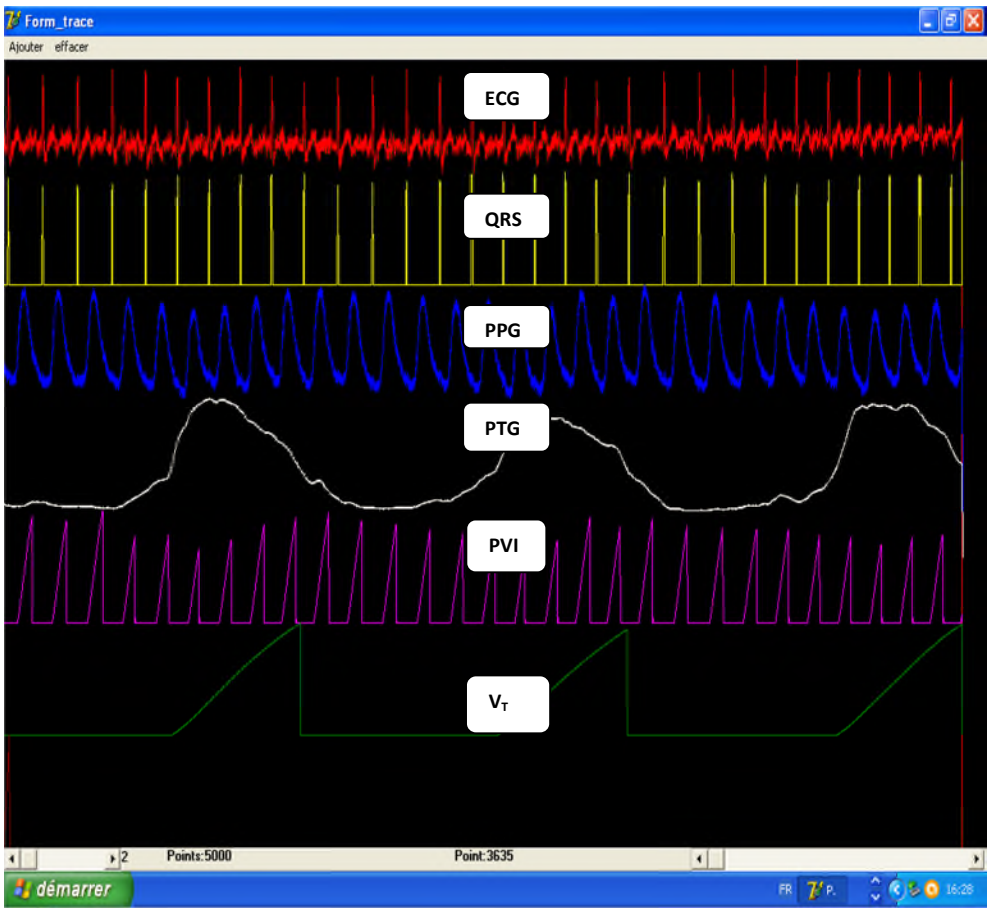


Figure V. 8: Affichage respectivement de l'ECG, du QRS, du PPG, du PTG

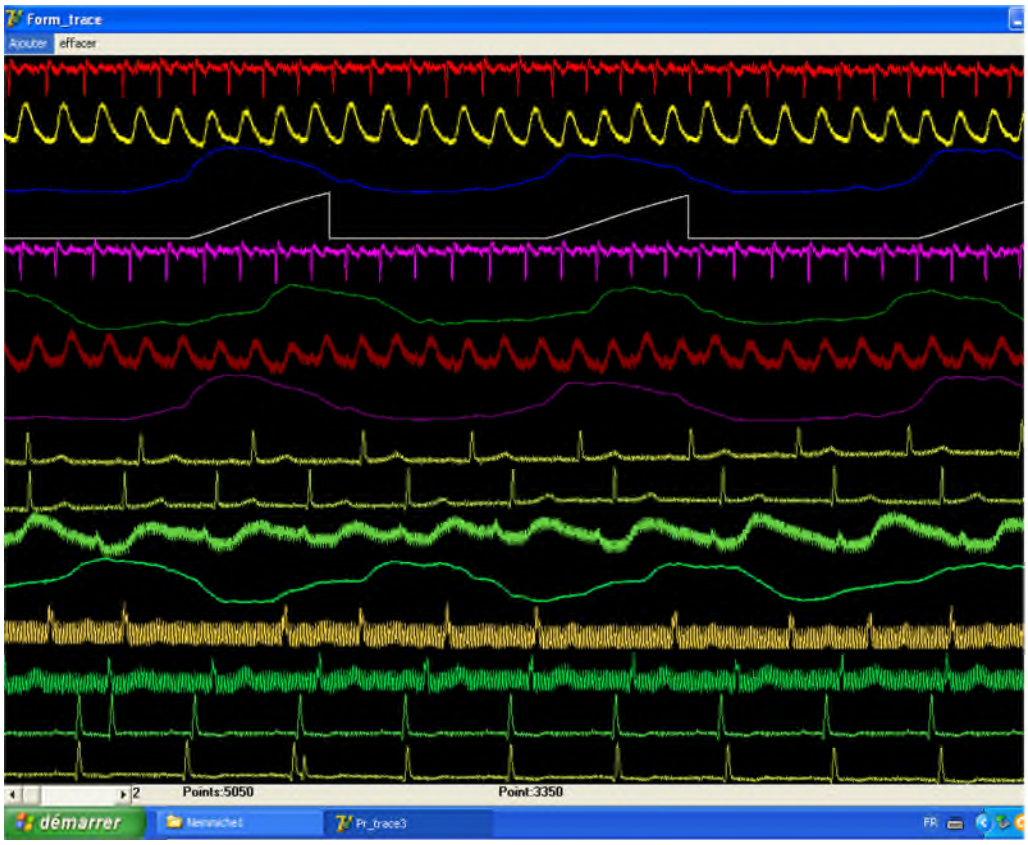


Figure V. 9: Affichage de 16 signaux archivés pris au hasard

V REALISATION PRATIQUE

La figure V.10 représente le tracé du circuit imprimé ,tandis que la figure V.11 montre une vue 3D de la platine réalisée.

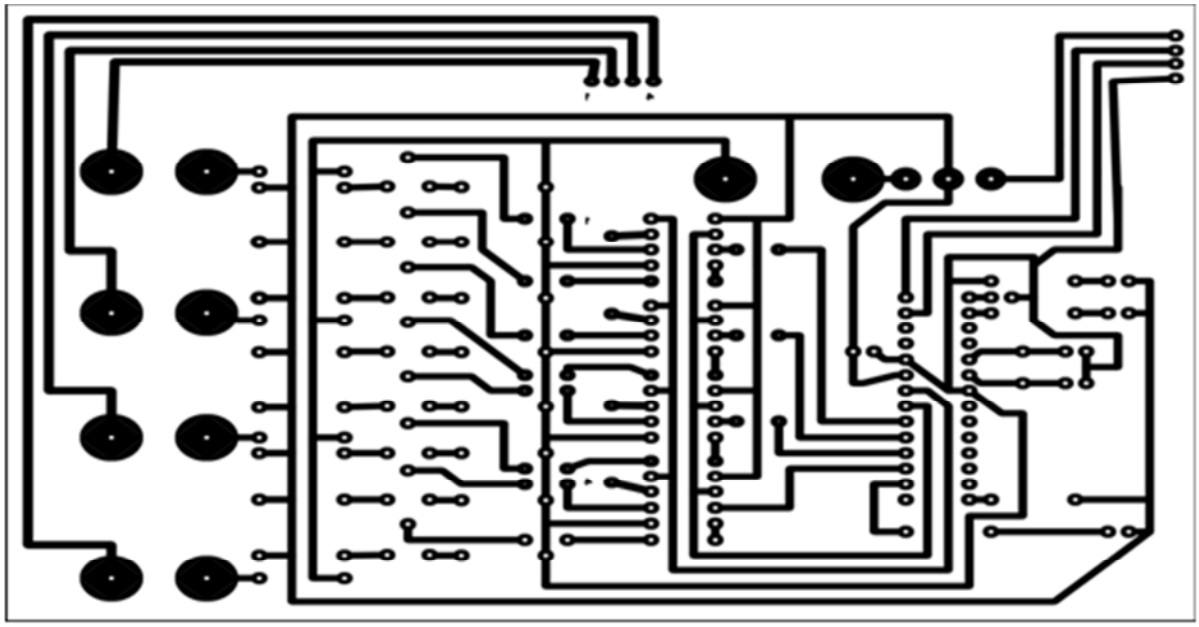


figure V. 10 Circuit imprimé

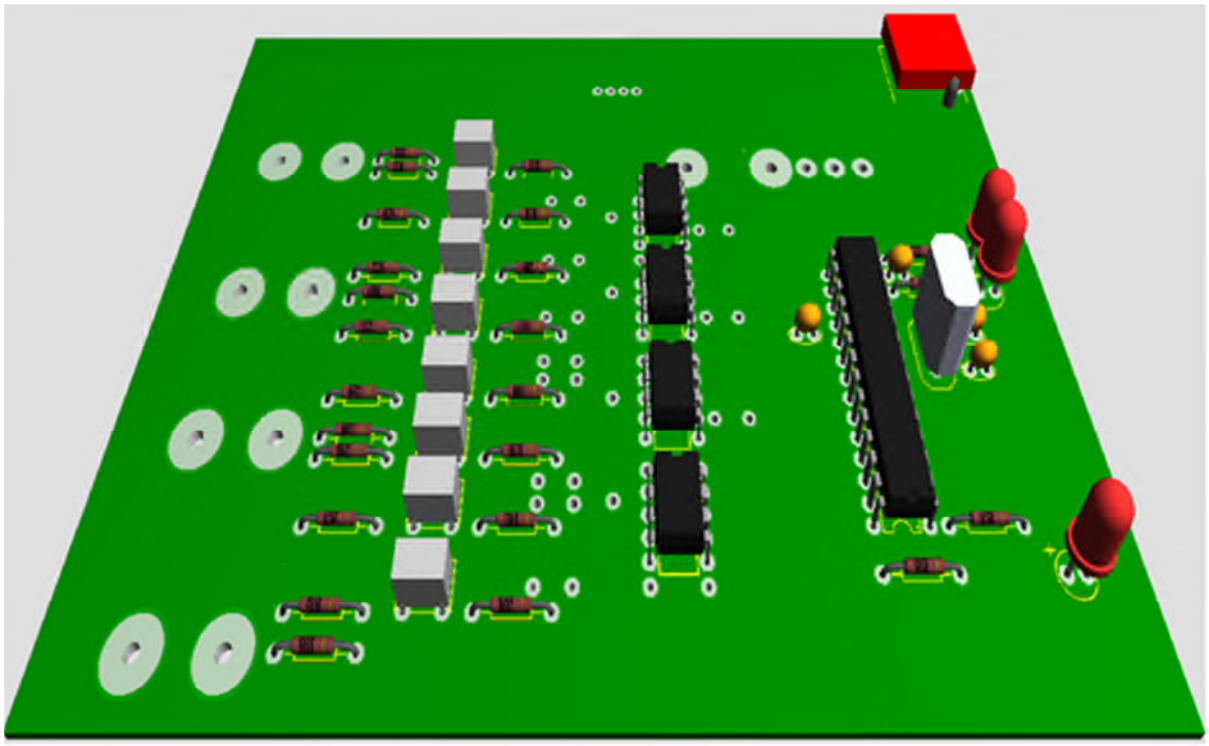


figure V. 11 Carte réalisée

Chapitre 6 : IMPLEMENTATION DES ALGORITHMES DE TRAITEMENT DES SIGNAUX

VI QUELQUES FONCTIONNALITES SOUS L'INTERFACE GRAPHIQUE UTILISATEUR

VI.1 Inversion de signal

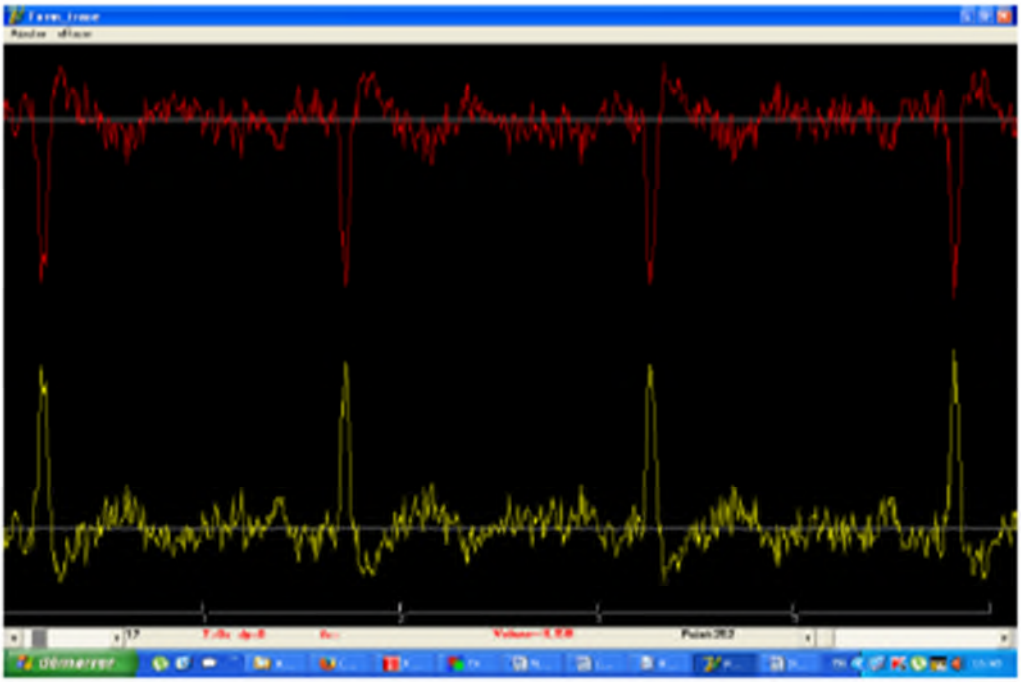


Figure VI. 1: Inversion d'un signal

VI.2 affichage de la fréquence cardiaque

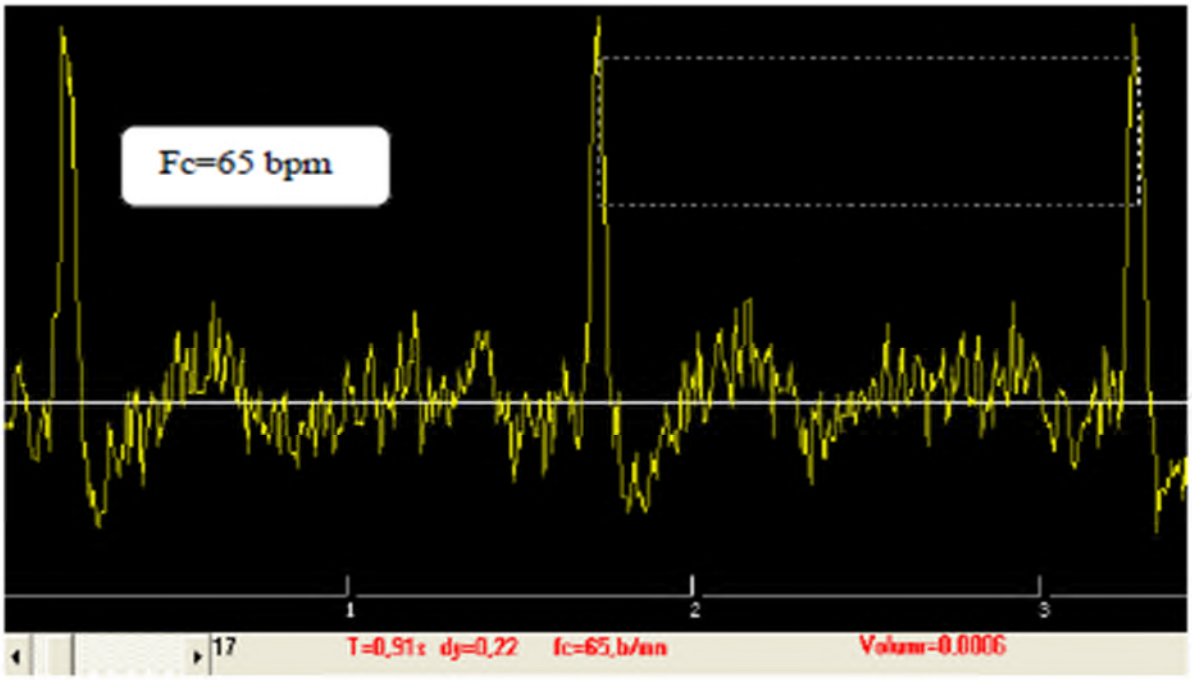


Figure VI. 2: Calcul et affichage de la fréquence cardiaque

VI.3 Superposition de 2 signaux

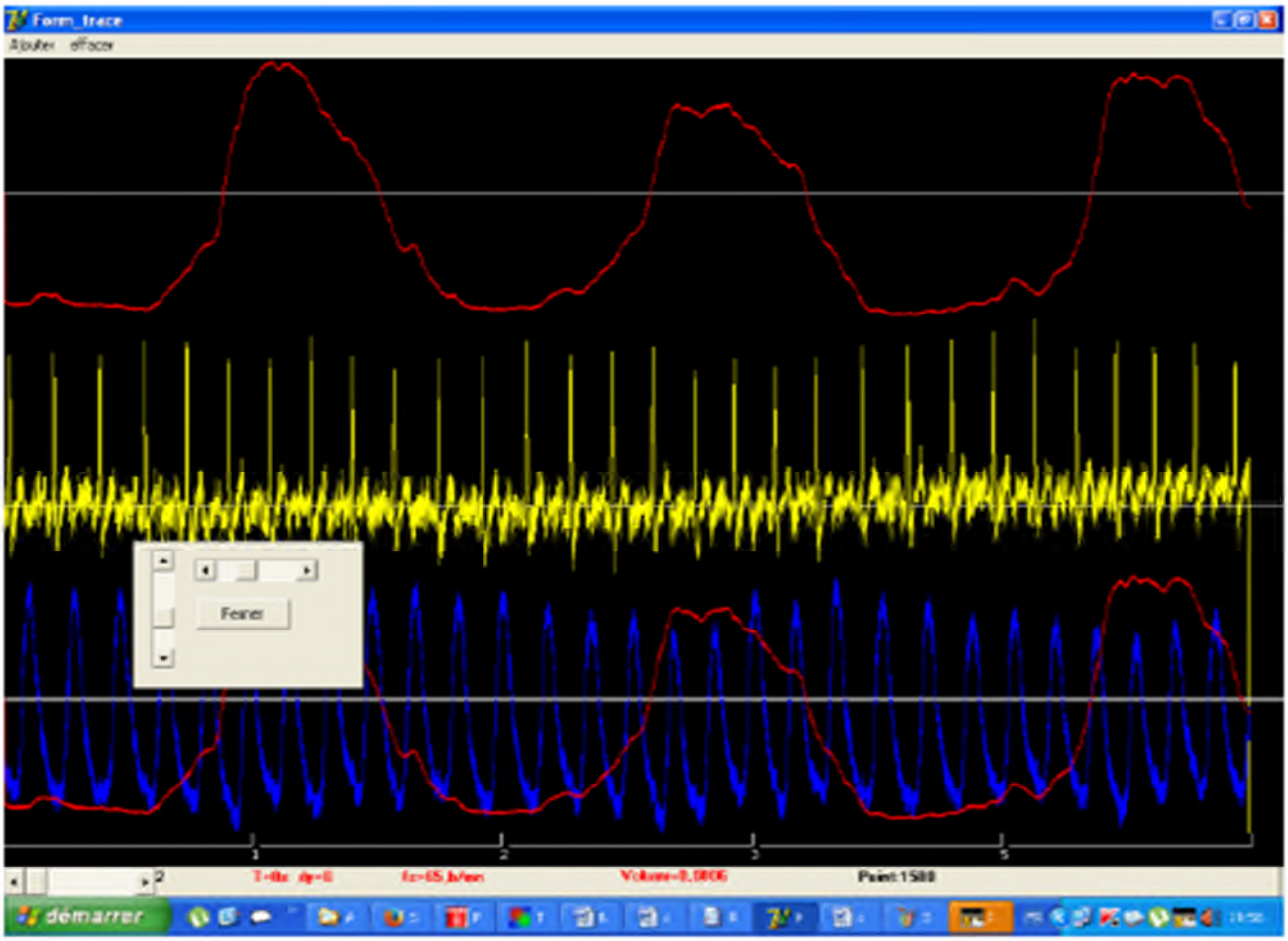


Figure VI. 3: Superposition de deux signaux

VI.4 Analyse spatiale

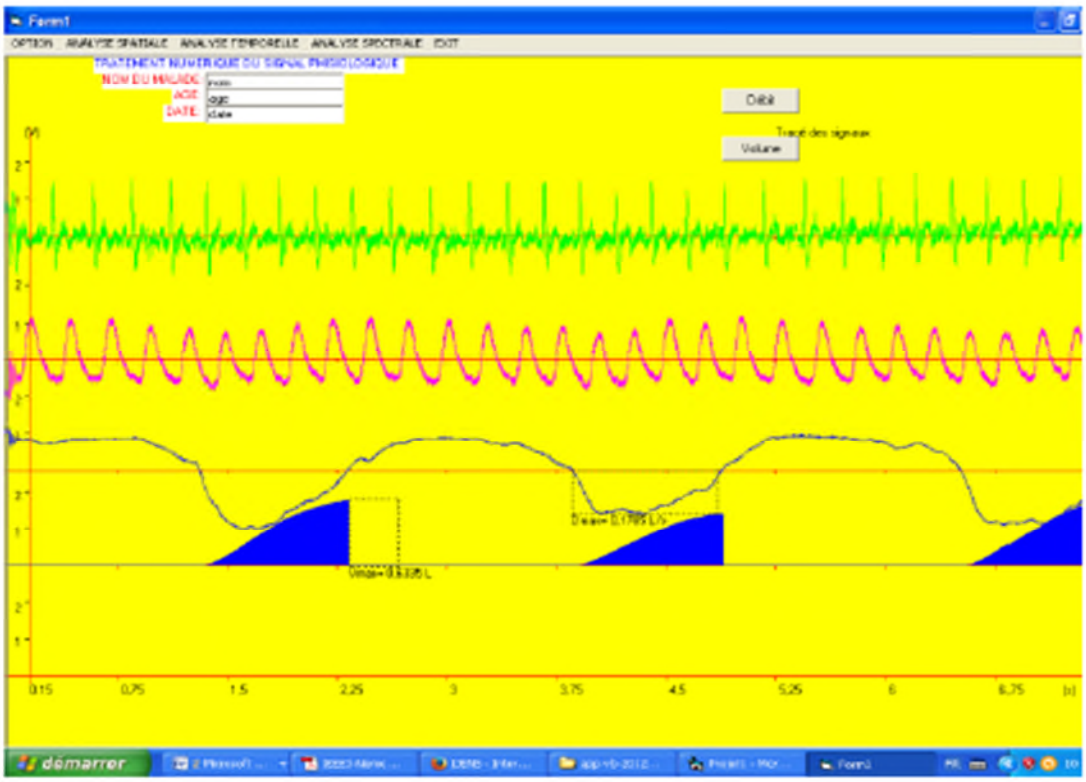


Figure VI. 4: Analyse spatiale

VI.5 Analyse temporel

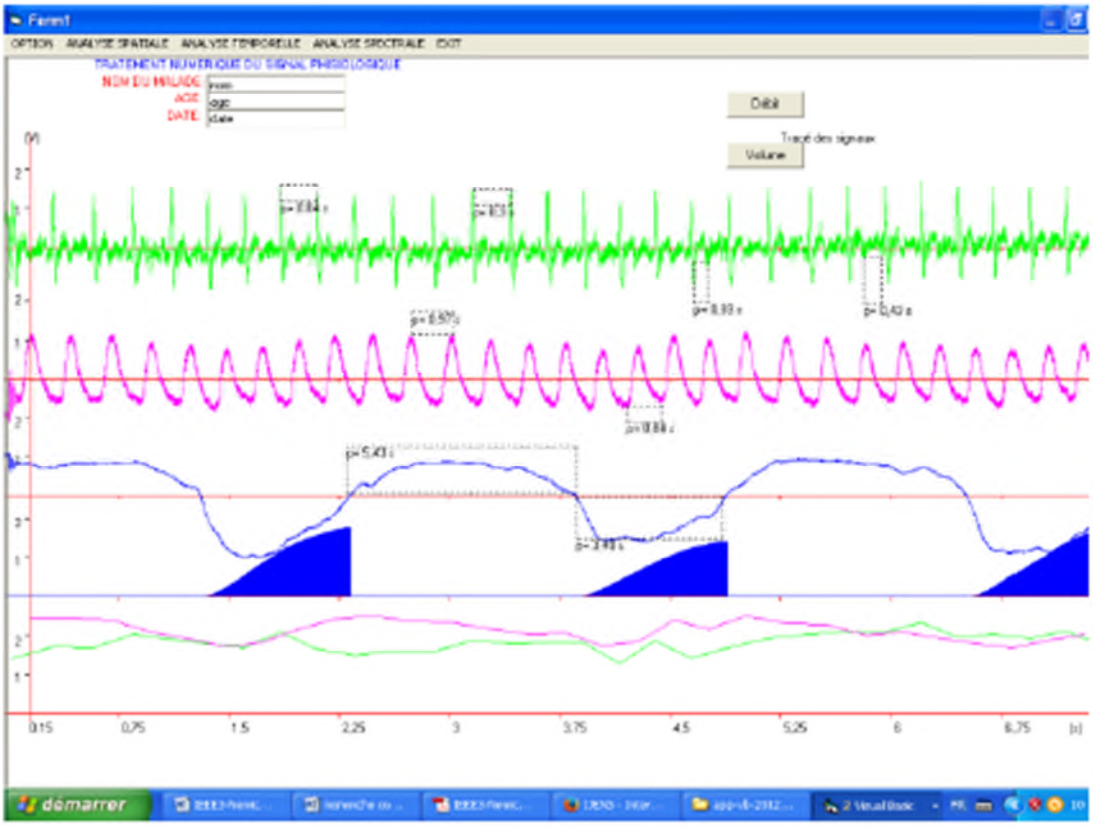


Figure VI. 5: Analyse temporelle

VI.6 Analyse spectrale

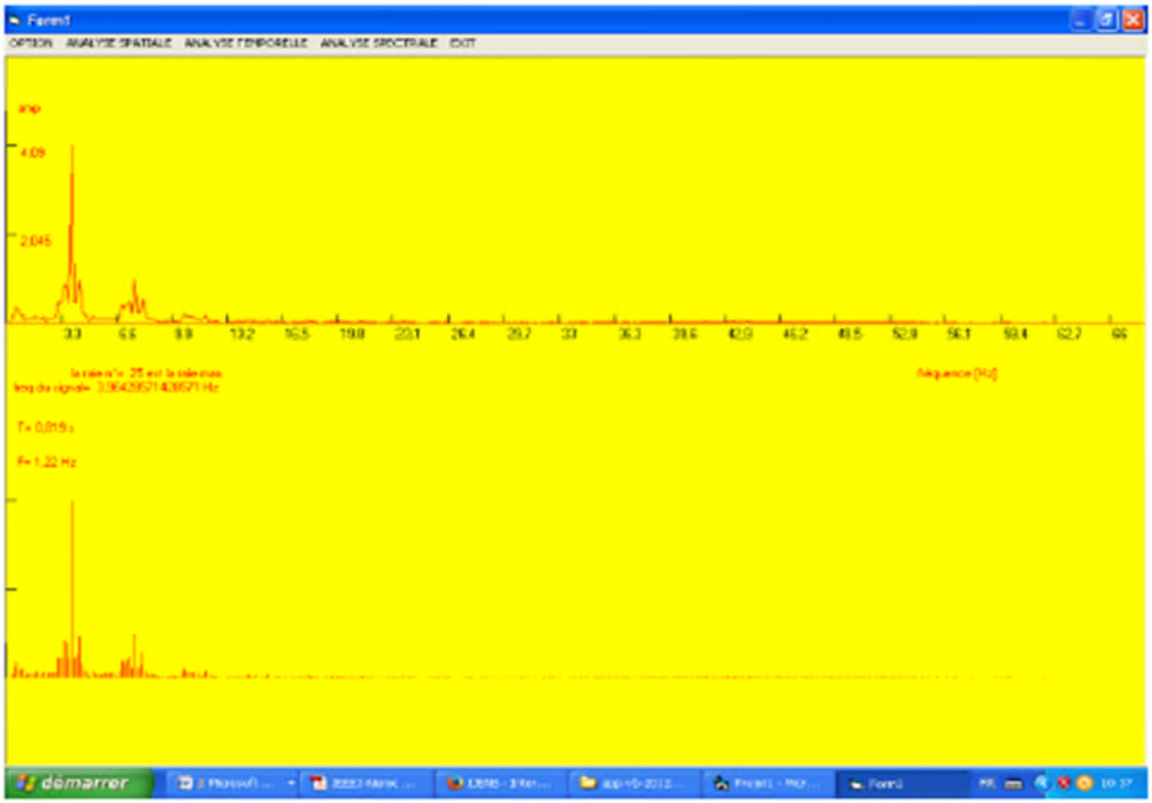


Figure VI. 6: : Analyse spectral

Chapitre 7 : IMPLEMENTATION DE LA PLATEFORME TELE MEDICALE SOUS ARCHITECTURE DE COMMUNICATION DISTANTE TCP-IP

I Le protocole TCP/IP

I.1 Introduction

Quand on parle de TCP/IP, on se réfère à différents concepts. En première analyse, le terme TCP/IP peut signifier protocole de communication pour la transmission des données. TCP signifie Transmission Control Protocol ; et IP Internet Protocol. Le terme TCP/IP n'est pas limité à l'expression Transmission Control Protocol/Internet Protocol.TCP/IP recouvre, en fait, toute une famille de protocoles comme UDP (User Datagramme Protocol), FTP (File Transfer Protocol), Telnet (Terminal Emulation Protocol), HTTP (Hyper Text Transfer Protocol), etc **[44]**.

Les réseaux utilisant TCP/IP sont souvent appelés des réseaux internet TCP/IP. Ne confondons pas un internet TCP/IP et l'Internet. Un internet TCP/IP est un réseau utilisant les protocoles TCP/IP, qu'il soit ou non relié à d'autres réseaux, tandis que l'Internet (avec I majuscule) représente le plus grand réseau mondial reliant des milliers d'ordinateurs entre eux sur tous les continents. Il est basé sur TCP/IP.

Quand les entreprises utilisent les services de l'Internet (en particulier le World Wide Web) basé sur le protocole HTTP) sur leur propre réseau privé, on dit que leur réseau forme un intranet.

Les technologies TCP/IP décrites dans cette thèse couvrent aussi bien les réseaux internet, intranets ou l'Internet.

I.2 **Origine [41][43][45]**

L'architecture TCP/IP a été développée, dans le milieu des années 70, par la DARPA (Defense Advanced Research Projets Agency – États-Unis) pour les besoins de communication et d'interfonctionnement des applications entre les systèmes informatiques de l'armée (DoD « Department of Defense »). Pour cela, il fallait définir un format d'échange des données commun à tous les systèmes tout en préservant l'existant, c'est-à- dire sans modifier les réseaux réels. L'architecture à définir devant garantir une grande résistance à la défaillance de n'importe quel nœud du réseau, le mode datagramme s'imposait pour la couche réseau. Pour conclure, TCP/IP, du nom de ses deux protocoles principaux (TCP, Transmission Control Protocol et IP, Internet Protocol), est un ensemble de protocoles permettant de résoudre les problèmes d'interconnexion en milieu hétérogène. À cet effet, TCP/IP décrit un réseau logique (réseau IP) au-dessus du ou des réseaux physiques réels, auxquels sont effectivement connectés les ordinateurs (fig.VII.1).

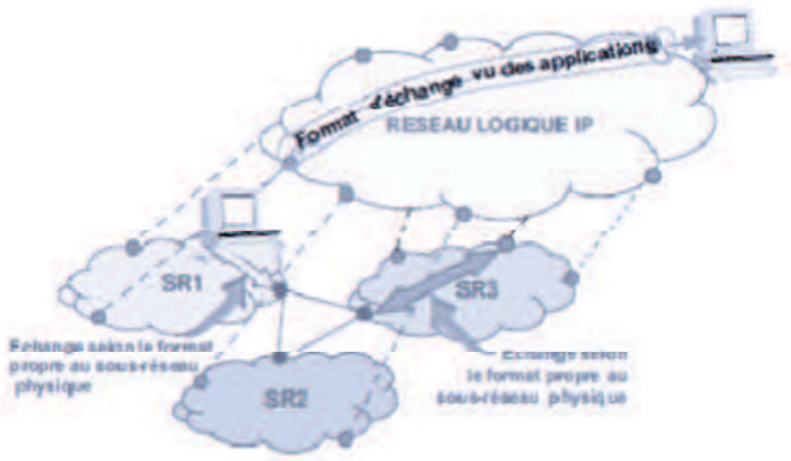


Figure VII. 1: Le réseau logique IP et sous-réseaux physiques réels (SRx).

Dans cette approche, les échanges entre applications sont réalisés selon le format défini par TCP/IP, alors que l'échange des données dans les sous-réseaux physiques réels se réalise selon le format propre à chaque sous-réseau. L'intégration de TCP/IP à UNIX BSD 4, par l'université de Berkeley, en fit le standard de la communauté UNIX (1980). En 1983, TCP/IP a remplacé le protocole NCP (Network Control Program) dans ARPANET, ancêtre de l'Internet. Aujourd'hui, TCP/IP est le protocole standard de tous les réseaux, du LAN au WAN. De récentes adaptations autorisent les flux multimédias et, en particulier, les services voix sur IP (ToIP, Telephony over TCP/IP).

I.3 Principe architectural

Précédant le modèle OSI, TCP en diffère fortement, non seulement par le nombre de couches, mais aussi par l'approche. Le modèle OSI spécifie des services (approche formaliste), TCP/IP des protocoles (approche pragmatique). Développé au-dessus d'un environnement existant, TCP/IP ne décrit, à l'origine, ni de couche physique ni de couche liaison de données. Les applications s'appuient directement sur le service de transport. Aussi l'architecture TCP/IP ne comprend que deux couches : la couche transport (TCP) et la couche inter-réseau (IP). La fig.VII.2 compare les deux architectures.

Il n'y a pas de couche application au sens OSI du terme, c'est-à-dire de couche présentant des « API » (Application Programming Interface) aux applications qui rendent transparents à ces dernières le ou les sous-réseaux réels de transport utilisés. Cependant, un mécanisme particulier, les sockets, assure une communication d'application à application en masquant les éléments réseaux.

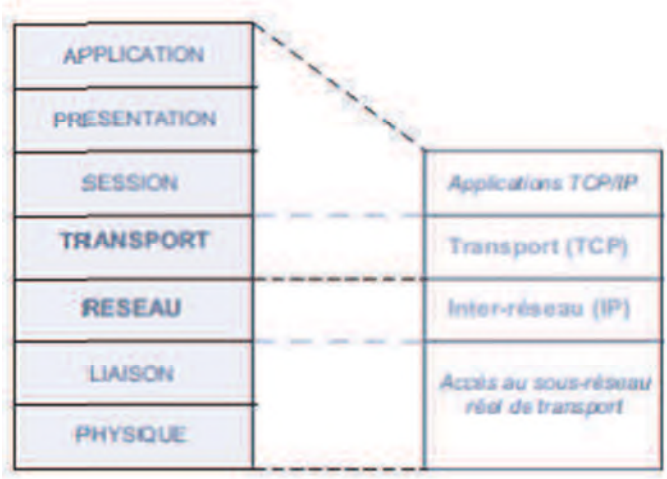


Figure VII. 2: Le modèle OSI et l'architecture TCP/IP.

La couche transport fournit deux types de service : un service en mode connecté (TCP) comparable, en ce qui concerne les services rendus, à TP4 d'ISO et un service de transport

allégé UDP (User Datagram Protocol) qui n'offre qu'un service de type best effort (datagramme).

La couche réseau (Internet Protocol, IP) présente les mêmes fonctionnalités que la couche réseau d'ISO en mode non connecté (mode datagramme), et les services rendus sont comparables à ceux de la norme ISO 8473 (couramment appelé CLNP/CLNS, Connectionless Network Protocol/ Connectionless Network Services).

I.4 La description générale de la pile et les applications TCP/IP

L'architecture TCP/IP comprend de nombreux programmes applicatifs, utilitaires et protocoles complémentaires (fig.VII.3). À l'origine TCP/IP ne spécifiait aucun protocole deligne, il s'appuyait sur les réseaux existants. L'utilisation massive de TCP/IP a fait apparaître le besoin de liaisons tout IP et donc la nécessité de disposer de protocoles de liaison spécifiques (SLIP, PPP). De même, TCP/IP a été adapté aux protocoles dits « haut débit » comme le Frame Relay2 et l'ATM (*Asynchronous Transfer Mode*) qui constituent aujourd'hui le cœur de la plupart des réseaux privés et d'opérateurs.

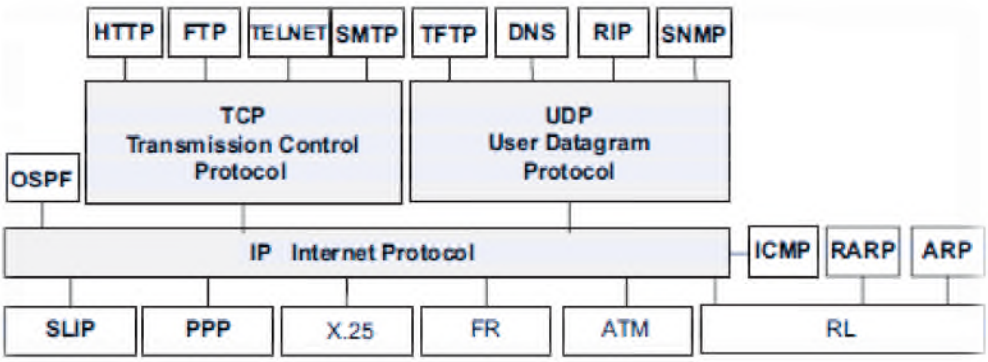


Figure VII. 3: . Les protocoles et les applications de TCP/IP..

Les principaux protocoles et applications de l'environnement TCP/IP sont les suivants:

– **ARP**, *Address Resolution Protocol*, met en correspondance une adresse logique IP avec

une adresse physique **MAC** (*Medium Access Control*, adresse de l'interface dans les réseaux locaux) ;

– **DNS**, *Domain Name System*, est un système de bases de données réparties assurant la correspondance d'un nom symbolique et d'une adresse internet (adresse IP) ;**FTP**, *File Transfer Protocol*, est un système de manipulation de fichiers à distance (transfert, suppression, création...) ;**HTTP**, *HyperText Transport Protocol*, assure le transfert de fichiers hypertextes entre un serveur Web et un client Web ;

– **ICMP**, *Internet Control and error Message Protocol*, assure un dialogue IP/IP et permet notamment : la signalisation de la congestion, la synchronisation des horloges et l'estimation des temps de transit... Il est utilisé par l'utilitaire **Ping** qui permet de tester la présence d'une station sur le réseau.

– **OSPF**, *Open Shortest Path First*, est un protocole de routage du type état des liens, il a succédé, dans le réseau Internet, au protocole RIP ;

– **PPP**, *Point to Point Protocol*, protocole d'encapsulation des datagrammes IP, il assure la délimitation des trames, identifie le protocole transporté et la détection d'erreurs.

– **RARP**, *Reverse Address Resolution Protocol*, permet l'attribution d'une adresse IP à une station ;

– **RIP,** *Routing Information Protocol*, est le premier protocole de routage (vecteur distance) utilisé dans Internet ;

– **SLIP**, *Serial Line Interface Protocol*, protocole d'encapsulation des paquets IP, ce protocole n'assure que la délimitation des trames ;

– **SMTP**, *Simple Mail Transfer Protocol*, offre un service de courrier électronique ;

– **SNMP,** *Simple Network Management Protocol*, est devenu le standard des protocoles d'administration de réseau ;

– **TELNET**, *TELetypewriter NETwork protocol* (ARPA) ou *TERminaL NETwork protocol*, système de terminal virtuel, permet l'ouverture de sessions avec des applications distantes ;

– **TFTP**, *Trivial FTP*, est une version allégée du protocole FTP.

I.5 Les mécanismes de base de TCP/IP [41]

I.5.1 *Le mode de mise en relation*

Désirant alléger au maximum la couche inter-réseau, les concepteurs de TCP/IP n'ont spécifié qu'une couche réseau en mode non connecté (mode datagramme). Ce mode de mise en relation optimise l'utilisation des ressources réseaux mais ne permet pas d'assurer ni un contrôle d'erreur, ni un contrôle de

flux. Au niveau du réseau, ces tâches peuvent être assurées par le ou les sous-réseaux réels de transport. Cependant, compte tenu qu'IP ignore la qualité de service offerte par ces sous-réseaux, la couche TCP pallie les insuffisances de la couche inter-réseau (*Internet Protocol*) en assurant le contrôle d'erreur, le contrôle de flux et de congestion. Cette approche, illustrée par la fig.VII.4, reporte sur les systèmes d'extrémité les tâches normalement dévolues à la couche réseau et en particulier le contrôle de congestion.

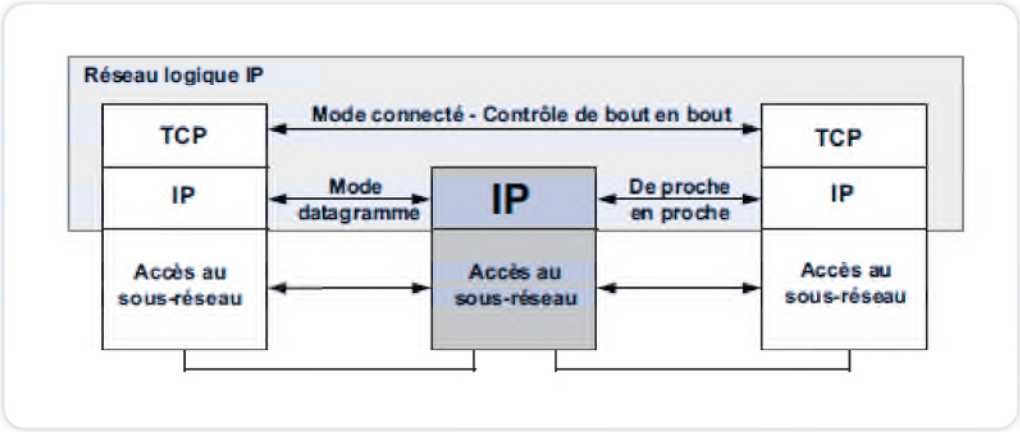


Figure VII. 4: Le réseau logique IP et les modes de mise en relation.

I.5.2 L'encapsulation des données

L'encapsulation consiste à transporter les données d'une couche dans une unité de données de la couche inférieure.Un en-tête contient les informations nécessaires à l'entité homologue distante pour

extraire et traiter les données. Dans le modèle TCP/IP, les données de l'application constituent des messages, ceux-ci sont transportés dans des segments TCP qui seront émis sur le réseau sous forme de datagrammes IP. L'unité de transport élémentaire est (fig.VII.5).La trame qui constitue au niveau physique un train de bits

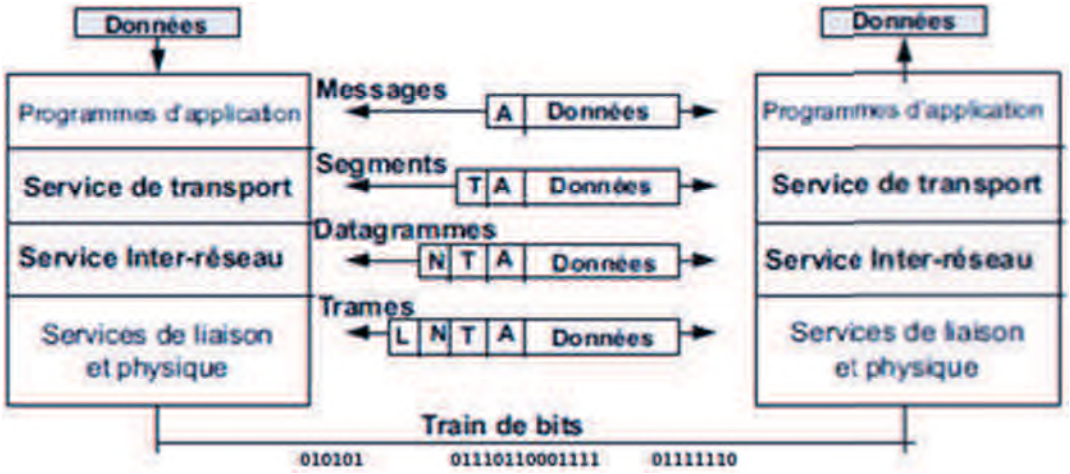


Figure VII. 5: L'encapsulation des données dans TCP/IP.

Contrairement au modèle OSI, TCP/IP utilise un format unique d'en-tête de taille fixe. Cette approche, en nécessitant des en-têtes relativement importants, pénalise le débit (*overhead* protocolaire),

mais optimise le traitement des blocs de données.

II Implémentation de l'application vidéoconférence

II.1 Les flux d'information

L'évolution des besoins et des applications informatiques conduit à l'acheminement, dans un même réseau de données informatiques traditionnelles (texte), de la voix et de la vidéo. Transporter sur un même système physique des flux d'information de natures différentes nécessite que chacun d'eux ait une représentation physique identique et que le système de transmission ait la capacité de prendre en compte les contraintes spécifiques à chaque type de flux (figure VII.6) **[41].**

Figure VII. 6: Le réseau et les différents flux d'information

Afin de qualifier ces différents flux vis-à-vis du système de transmission, nous définirons succinctement les caractéristiques essentielles d'un réseau de transmission1. Ensuite, nous examinerons le mode de représentation de ces informations. Enfin, nous appliquerons les résultats aux données, à la voix et à l'image pour en déduire les contraintes de transfert spécifiques à chaque type de flux.

II.1.1 Les différents types d'information

Les informations peuvent être réparties en deux grandes catégories selon ce qu'elles représentent et les transformations qu'elles subissent pour être traitées dans les systèmes informatiques.aprés **[41]** on distingue :

1 **Les données discrètes,** l'information correspond à l'assemblage d'une suite d'éléments indépendants les uns des autres (suite discontinue de valeurs) et dénombrables (ensemble fini).

Par exemple, un texte est une association de mots eux-mêmes composés de lettres (symboles élémentaires).

2 Les **données continues ou analogiques** (figure VII.7) résultent de la variation continue d'un

phénomène physique : température, voix, image... Un capteur fournit une tension électrique qui varie de manière analogue à l'amplitude du phénomène physique à analyser : signal analogique.

Dans un intervalle déterminé (bornes) aussi petit que possible, un signal analogique peut toujours prendre une infinité de valeurs. Par exemple pour passer 10° C à 11° C, la température prend, entre ces deux valeurs, une infinité de valeurs sans aucune discontinuité entre elles (fonction continue).

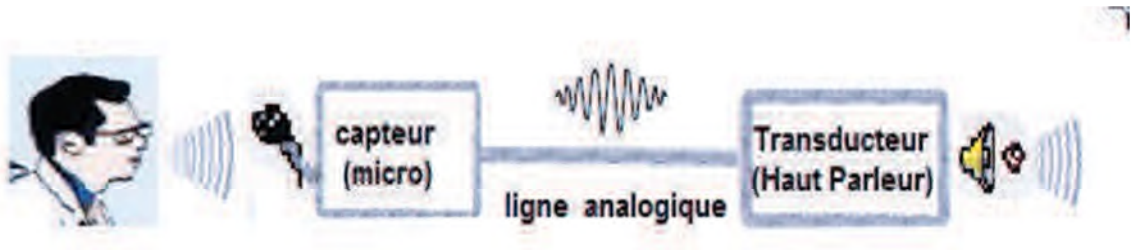


Figure VII. 7: Le signal analogique.

Pour être traitées par des équipements informatiques, les informations doivent être représentées par une valeur binaire (**codage à la source**). Le codage à la source est plus spécifiquement appelé codage de l'information pour les informations discrètes et numérisation de l'information pour les informations analogiques.

II.1.2 Les données et les contraintes de transmission [41]

Les réseaux transportent des flux numériques informationnels. Selon le type de données, les exigences en termes de débit (volume), de temporalité (temps de transfert et variation de celui-ci) et fiabilité (taux d'erreur) diffèrent. Un système de transmission multimédia doit être capable de garantir à chaque type de flux le respect de ses contraintes spécifiques.

Ainsi, un transfert de fichiers correspond à un flux binaire constant. Il requiert un débit relativement important mais est très peu sensible au temps de transmission. Plus exigeantes en termes de temps de transfert (interactivité), les applications informatiques de type conversationnel sont caractérisées par la sporadicité des flux qu'elles soumettent au système de transmission.

Moins sensibles aux erreurs, la voix et la vidéo ont des exigences strictes en matière de débit (débit minimal garanti), de temps de transfert et surtout de récurrence temporelle (gigue), elles sont qualifiées de données isochrones[4]. La compression opérée sur ces types de données engendre des flux variables. Le tableau 7.1 **[41]** résume ces différentes caractéristiques.

Type de transfert	Débit requis	Débit requis	Sensibilité à la variation de du temps transfert	Sensibilité
Voix	Constant	Faible	Élevée (isochrone)	Faible
Voix compressée	variable	Faible	Élevée (isochrone)	Faible
Vidéo non compressée	Constant	Élevé	Élevée (isochrone)	Faible
Vidéo compressée	variable	Élevé	Élevée (isochrone)	Faible
Transactionnel et transfert de fichiers	En rafale (bursty)	Moyen à élevé	Faible	Élevée
Interconnexion de réseaux locaux	En rafale, débit de la source élevé.	Élevé	Faible	Élevée

Tabl.7.1 Types de données et contraintes de transmission.

II.1.3 Notion de débit binaire

Les systèmes de traitement de l'information emploient une logique à deux états dite « binaire ».Pour y être traitée, l'information doit être traduite en symboles compréhensibles et manipulables par ces systèmes. Selon le type d'information à transformer, l'opération qui consiste à transformer les données en éléments binaires s'appelle le codage ou numérisation.

On appelle débit binaire (D) le nombre d'éléments binaires, ou nombre de bits, émis sur le support de transmission pendant une unité de temps. Le débit binaire est généralement la grandeur utilisée en premier pour qualifier un système de transmission ; il s'exprime par la relation :

$$D = \frac{V}{t}$$

Avec D : (débit) en bits[5] par seconde (bit/s), V : volume à transmettre exprimé en bits, t : durée de la transmission en seconde. Le débit binaire mesure le nombre d'éléments binaires transitant sur le canal de transmission pendant l'unité de temps.

II.2 Représentation général de l'Habitat Intelligent pour la Santé (HIS)

La figure VII.8 donne une illustration globale de l'Habitat Intelligent pour la Santé.

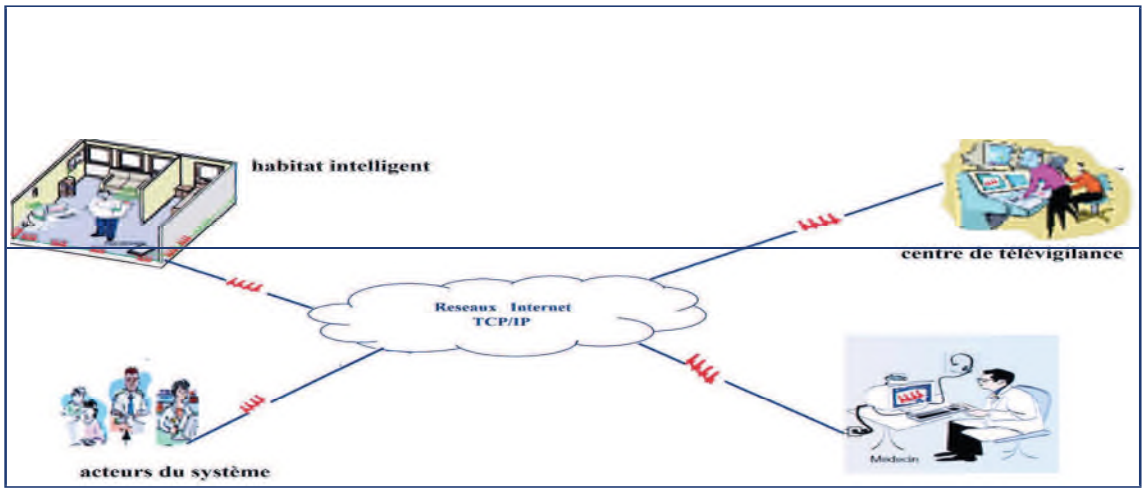


Fig.VII.8 I m plémentation de HIS dans le réseau global

Dans le chapitre précédent nous avons décrit de manière détaillée l'implémentation hardware et software d'un HIS. La figure V I I.9 donne un exemple de configuration d'un patient en domiciliation HIS poursuivant normalement ses activités journalières et bénéficiant d'une télé surveillance permanente des ses paramètres vitaux.

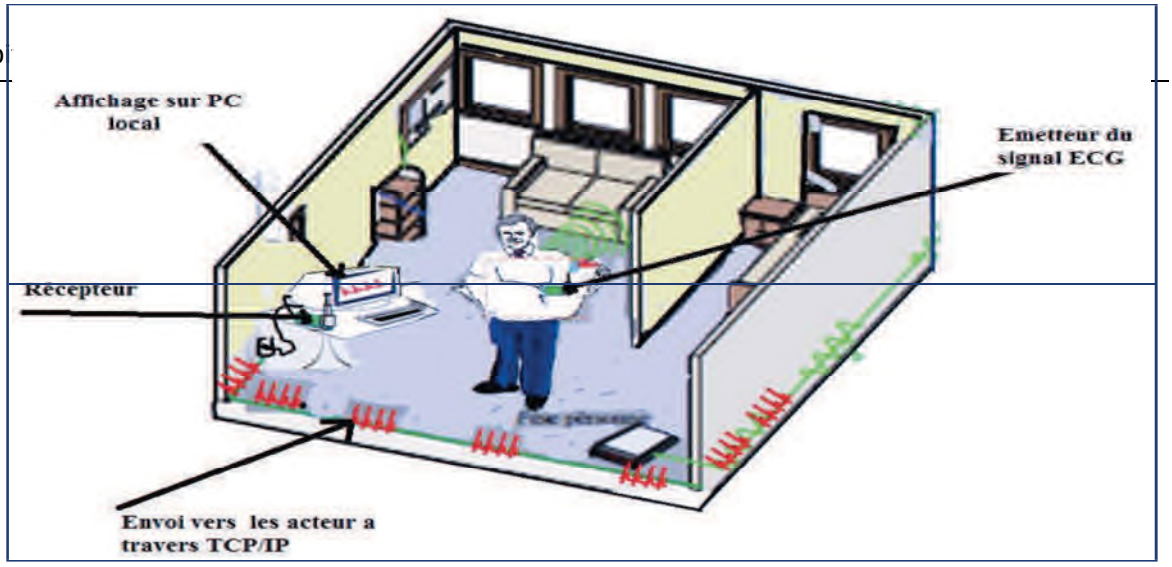


Fig.VII.9 Implémentation de HIS dans le réseau local

Après avoir réalisé la transmission sans fil des paramètres vitaux prélevés sur le patient au poste local de l'HIS nous présentons dans ce qui suit l'implémentation software de l'interface de communication multimédia sous environnement Visual Basic comprenant :

1 – Une liaison textuelle. 2

- Une liaison vocale

3 – Une liaison vidéo

4- Un transfert de fichiers multiformes.

Toutes ces liaisons sont configurées selon l'architecture Client – Serveur, mettant en œuvre la technologie des sockets.

II.3 **Présentation rapide du système vidéoconférence**

D'une manière générale un système de visioconférence met en jeu une caméra, un microphone, un écouteur, la carte son ; la carte réseau, la carte graphique et le port USB de deux ou plusieurs terminaux informatiques interconnectés au moyen d'un logiciel dédié assurant la capture et le

transfert des différent flux (textuel, vocal, vidéo, données), d'un système d'exploitation, (Dans notre cas Windows), de la technologie des sockets (Dans notre cas le composant Winsock de VB) et d'un protocole de communication (Dans notre cas le protocole TCP/IP).

La figure 7.17 donne une illustration de système de vidéoconférence.

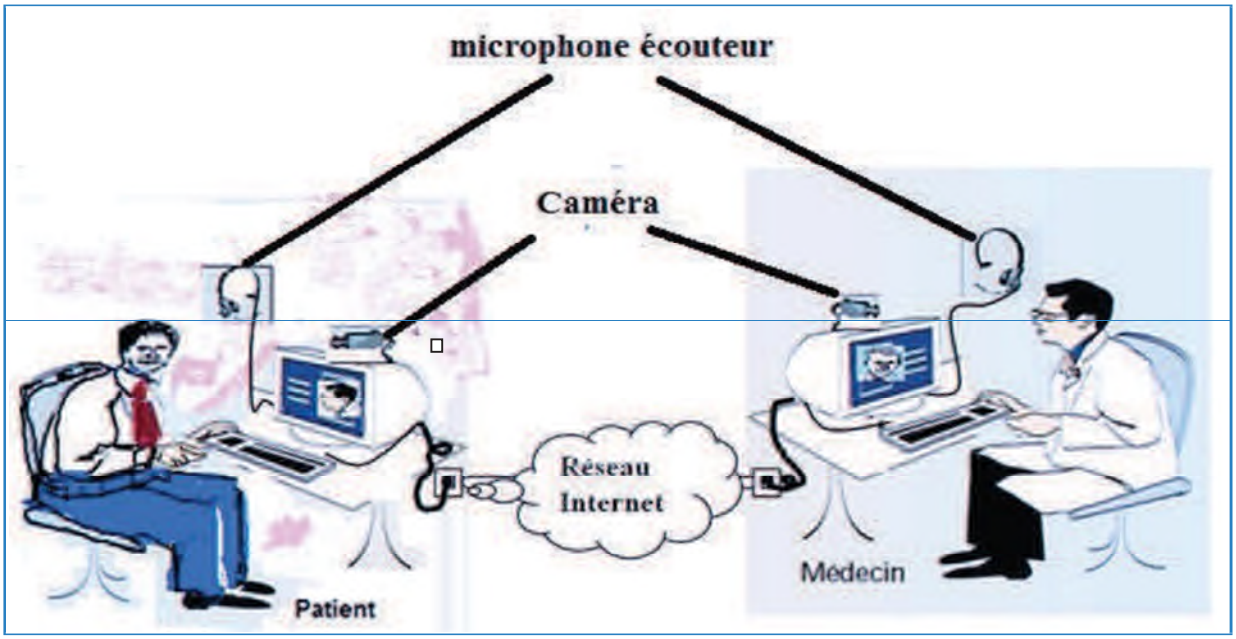


Figure VII. 10: Représentation de vidéoconférence

II.4 L'interface logicielle de communication

Notre programme est composé de deux parties, l'émission et la réception. Il se présentée sous la forme d'un transfert en temps réel et permet donc à un utilisateur d'envoyer vers l'ordinateur récepteur les différents flux.

II.4.1 Représentation de l'interface de communication entre le patient et le médecin

Dans cette partie nous présentons l'interface développée sous environnement VB. Les figures VII.11 et VII.12 montrent que celle-ci se compose de deux parties.

Chaque coté est constitué d'une fenêtre principale pour gérer la mise en route de la connexion ainsi que les différents types de communication. Elle comporte :

- Une fenêtre pour le transfert textuel.
- Une fenêtre pour le transfert vocal.
- Une fenêtre pour le transfert vidéo
- Une fenêtre pour le transfert de fichiers

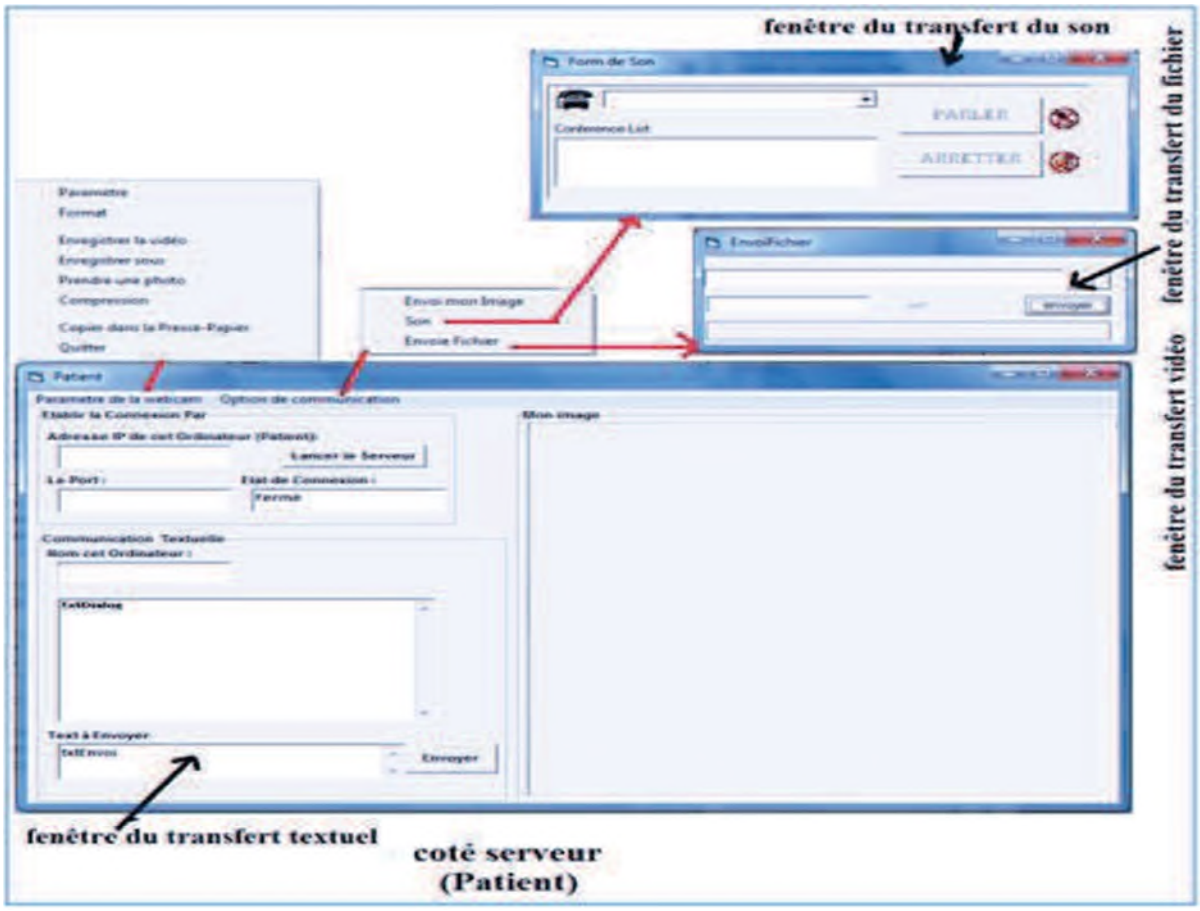


Figure VII. 11: L'interface principal côté Patient

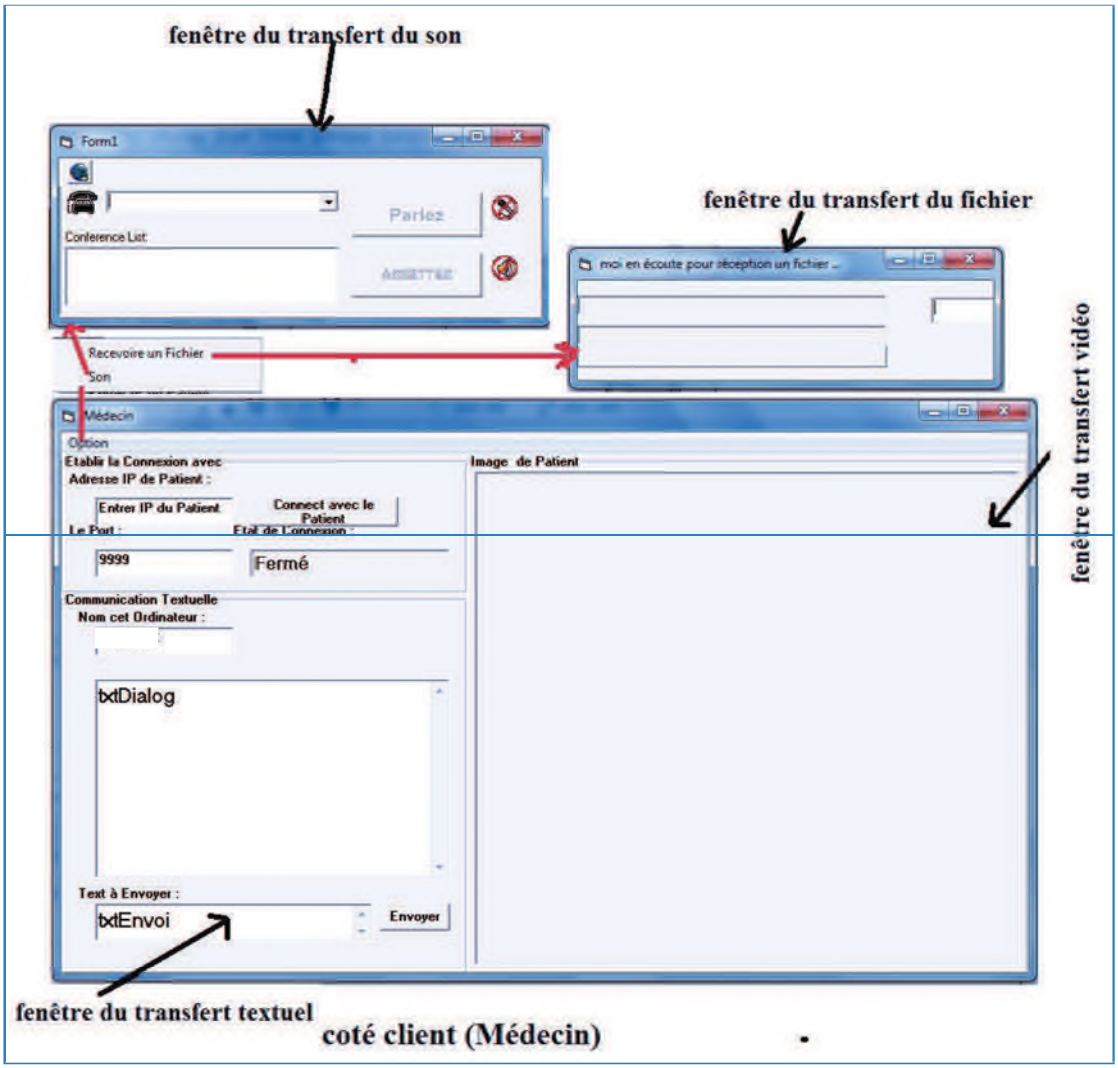


Fig.VII.12 L'interface principal côté Médecin

Toutes ces fenêtres se partage avec des procédures et des fonctions accessibles et d'autres routine masquées.

Dans ce qui suit nous expliquons en détail chaque fenêtre de l'interface, ce qu'elle comporte comme difficultés auxquelles nous avons apportés des solutions.

II.4.2 Procédure et Organigramme de l'établissement de la connexion Internet entre le serveur et le client (Patient/Médecin)

Avant l'échange de n'importe quelle information par internet sous protocole TCP/IP entre le patient et le médecin, il faut tout d'abord établir une connexion entre les deux correspondants.

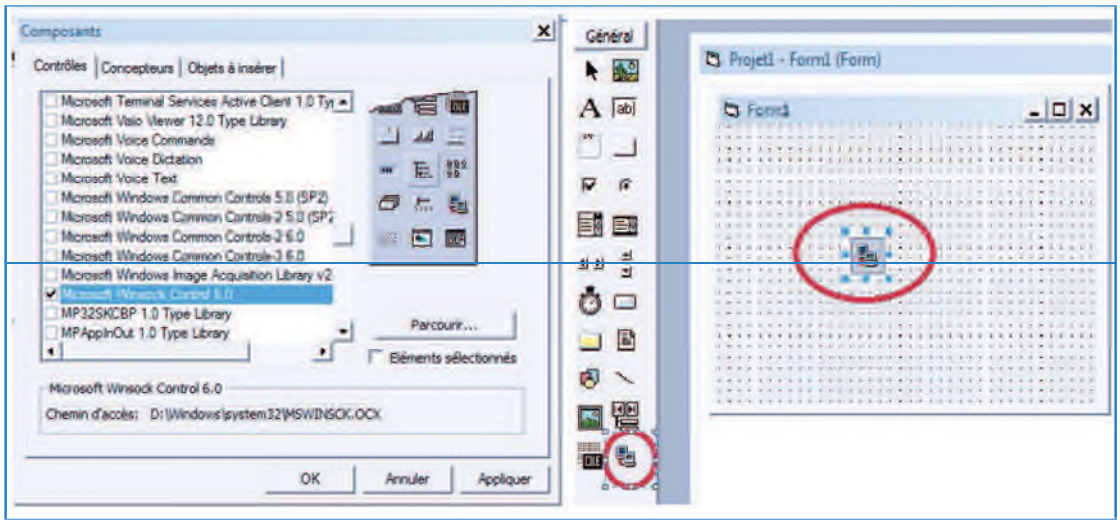
La procédure suivante décrit comment nous établissons cette connexion :

Implémentation coté patient :

Le contrôle Winsock, déjà expliqué au chapitre IV, permet à des applications client- serveur (Patient/Médecin) de communiquer *via* les protocoles *TCP* ou *UDP*. Nous n'utiliserons dans notre thèse que le protocole *TCP*.

Plusieurs étapes sont à effectuer pour construire un serveur ou un client:

3 **L'étape 1** consiste à ajouter un contrôle Winsock à la feuille. Ce contrôle, non intrinsèque, est accessible à travers le composant Microsoft Winsock Control 6.0



4 **L'étape 2** consiste à détecter et à afficher l'adresse IP du serveur (côté patient).

5 **L'étape 3** consiste à spécifier le numéro de port, utilisé lors deS connexions avec des clients, à travers la propriété Winsock1.LocalPort du contrôle Winsock. Chaque client fera une requête au serveur via ce port.

6 **L'étape 4** permet au serveur d'attendre une demande de connexion par un client à travers la propriété Winsock1.Listen du contrôle Winsock . Le serveur est informé d'une demande de connexion par un client à travers l'exécution de la procédure événementielle Connection_Request .

7 **L'étape 5** consiste à accepter la connexion entrante à travers la méthode Winsock1.Accept du

contrôle Winsock .Une fois cette méthode appelée, des données peuvent être transmises entre le serveur et le client. Avant d'accepter la connexion, IL faut toujours vérifier l'état du contrôle Winsock et s'assurer qu'il correspond à une fermeture du port de communication **WINSOCK1.CLOSED** . Dans le cas contraire, il s'agit d'appeler la méthode Close pour fermer la précédente connexion. Si la connexion entrante est refusée, l'homologue (client) reçoit l'événement Close.

8 **L'ETAPE** 6 est relative à la phase de communication entre le serveur et le client. La procédure événementielle **DATA_ARRIVAL** du contrôle Winsock est exécutée lorsque des données parviennent au serveur. L'instruction **WINSOCK1.GETDATA**(Data) place les données dans la chaîne de caractères Data arguments optionnels permettant de spécifier le type de données reçues et la longueur maximale autorisée des données).Afin d'envoyer des données au client, on utilise la méthode **SENDDATA** du contrôle Winsock. Par exemple, l'instruction **WINSOCK1.SENDDATA** (Data) permet de transmettre les données contenues dans Data.

9 **L'ETAPE** 7 se produit lorsque la transmission des données est terminée. Le fait que le client ferme la connexion provoque l'exécution de la procédure événementielle Close du contrôle Winsock. La connexion serveur devra être fermée via l'instruction **WINSOCK1**.Close

Un contrôle Winsock doit être attribué à chaque connexion avec un client. Le fait de pouvoir disposer dans Visual Basic de tableau de contrôles permet la création de serveurs capables de gérer simultanément plusieurs connexions, ceci sans créer a priori un ensemble suffisant de contrôles Winsock.

Implémentation cote médecin

Plusieurs étapes sont à effectuer pour construire un client ou bien coté médecin) :-

3 **L'étape 1** consiste à ajouter un contrôle Winsock à la feuille (accessible à travers le composant Microsoft Winsock Control 6.0).

4 **L'étape 2**, le contrôle Winsock côté client doit pouvoir localiser :

4.11 l'ordinateur distant sur lequel un contrôle envoie, ou reçoit des données. Nous pouvons fournir soit un nom d'hôte, par exemple "www.microsoft.com", soit une adresse IP sous forme de chaîne ponctuée, telle que "127.0.1.1". Ce nom est placé dans la propriété RemoteHost du contrôle Winsock ;

4.12 le numéro de port distant auquel la connexion doit être faite. Ce numéro est placé dans la propriété RemotePort du contrôle Winsock .

- L'étape 3, la connexion au serveur est demandée via un appel de la méthode Connect du contrôle Winsock. En cas de succès, la procédure événementielle Connect du contrôle Winsock s'exécute ; en cas d'erreur, la procédure événementielle Error du contrôle Winsock s'exécute.

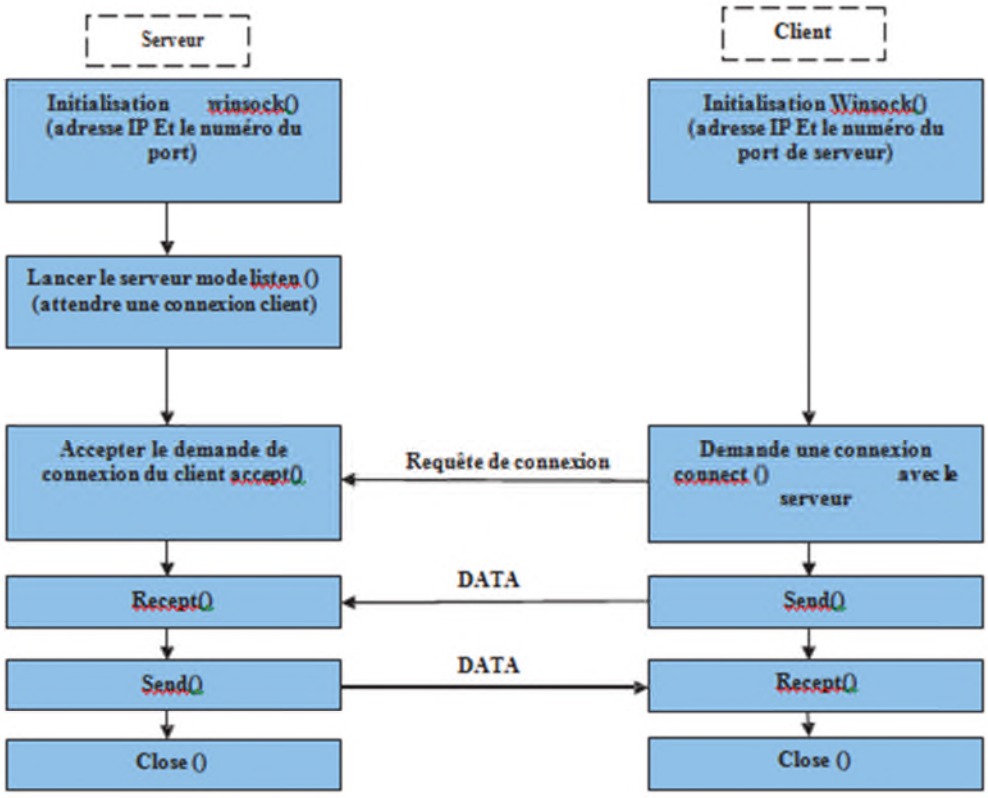
5 **L'étape 4** est relative à la phase de communication entre le serveur et le client. Comme du

côté serveur, la procédure événementielle Data_Arrival du contrôle Winsock est exécutée lorsque des données parviennent au client. L'instruction Winsock1.GetData (Data) place les données dans Data. Des données sont envoyées au serveur en utilisant la méthode SendData du contrôle Winsock. Par exemple, l'instruction Winsock1.SendData (Data) envoie au serveur des données contenues dans Data.

6 **L'étape 5** se produit lorsque la transmission des données est terminée. Le fait que le serveur ferme la connexion provoque l'exécution de la procédure événementielle Close du contrôle Winsock . La connexion client devra être fermée via l'instruction Winsock1.Close.

Après réussite de la connexion (patient ''connecté'' et médecin ''connecté'') tous les deux peuvent échanger des informations en respectant les procédures et les contraintes de chaque type de donnée (Texte, image, son, fichier ou signal physiologique).

Nous représentons ci-dessous l'organigramme relatif à toutes ces étapes :



II.4.3 Procédure d'envoi des textes

Les textes sont très simples à envoyer après l'établissement d'une connexion.

Côté Patient : Avec l'instruction **Winsock1.SendData** txtPseudo & " a dit >" & " " & vbCrLf & **txtEnvoi.Text**

Le patient envoie au Médecin les textes contenus dans **txtEnvoi.Text**.

Coté Médecin : la procédure événementielle Winsock1_DataArrival est exécutée lorsque des données parviennent du patient. L'instruction Winsock1.GetData (Data) place les données dans Data pour être affichées dans la zone textuelle de médecin par l'instruction ;

```
txtDialog.Text = txtDialog.Text + Data + vbCrLf + "----------" + vbCrLf
```

Les es données textuelles sont envoyées au Patient en utilisant la même méthode et les mêmes instructions comme le montre la figure VII.13.

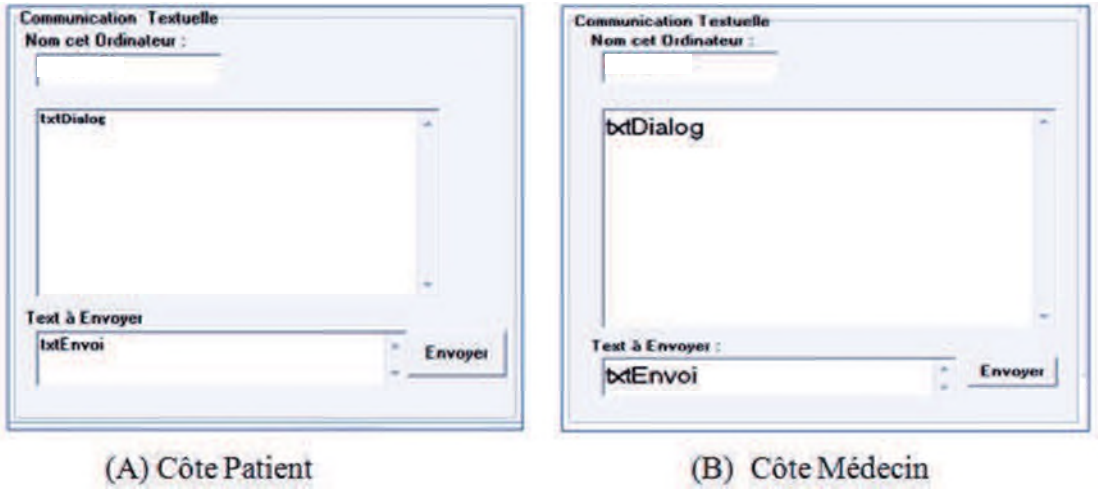


Fig.VII.13 Fenêtre de transfert textuel

II.4.4 Procédure et organigramme de Visioconférence

La visioconférence consiste à réaliser une conférence à distance entre le Médecin et le patient associant l'image et le son, à l'aide d'un système comportant une caméra et un microphone. Elle nécessite la mise en place d'une connexion Internet de bonne qualité pour ne pas rencontrer de problème de qualité (image ou son).

II.4.4.1 0rganigramme et procédure pour l'envoi des images vidéo

La premier partie à décrire dans cette section concerne les problèmes peuvent être rencontrés dans le développement d'une l'interface de communication vidéo.

Le premier problème à résoudre c'est la communication de l'interface avec la camera à travers le port USB dans le système d'exploration Windows car on ne peut pas relier notre interface à la caméra directement étant donné que le système Windows interdit toutes communications ou accès au port de manière direct pour une protection générale du système.

Le deuxième problème est inhérent au fait que notre programme est composé de deux parties, l'émission et la réception. Pour cela il est nécessaire de programmer de telle sorte à donner au système la diversité, l'adaptation et la sensibilité de communication pour arriver à un système qui

fonctionne en temps réel avec une marge d'erreur acceptable pour des communications de vidéoconférence.

Le troisième problème est inhérent au volume des données générées par la caméra qui sont considérables, ce qui induit des problèmes de stockage et de transport

Prenons l'exemple d'une image au standard VGA (640 × 480 en 256 couleurs). Un codage sur 8 bits (1 octet) de chaque pixel de l'image donne une gamme de 256 couleurs (ou 256 niveaux de gris dans le cas d'une image en noir et blanc). Il faudra donc :

640 × 480 = 307 200 octets

1 Ko = 1 024 octets donc 307 200/1 024 = 300 Ko pour stocker une telle image. Avec 25 images (soit une seconde de vidéo), nous obtenons un poids de 7,5 Mo. Nous constatons sur cet exemple qu'il est impossible de transmettre en temps réel la vidéo sans passer par une technique de compression au d'autre technique de codage.

Pour résolution de ces problèmes on peut citer différentes méthodes avec des systèmes de développement différents mais la solution doit avoir la simplicité et la rapidité d'utilisation.

Pour résoudre le premier problème nous faisons appel au DLL (Direct Link Library, bibliothèque de liaison dynamique) qui sont des composantes de programmation avec des architectures prêtes à être utilisées Ces DLL sont des composantes développées par le fournisseur d'origine de notre système de développement.

Pour notre application nous avons trouvé une DLL pour la communication entre notre interface et la caméra à travers le port USB. Cette DLL est dénommée « **avicap32.dll** ».

L'autre difficulté à surmonter est celle de la synchronisation des flux vidéo transmis et reçus en temps réel

Toutes les fonctions utilisées dans cette DLL sont décrites dans ce qui suit :

4 Nous déclarons les variables et la fonction de la DLL come le suit :

```
Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias "capCreateCaptureWindowA"
(ByVal lpszWindowName As String, ByVal dwStyle As Long, ByVal X As Long, ByVal y As Long, ByVal
nWidth As Long, ByVal nHeight As Integer, ByVal hWndParent As Long, ByVal nID As Long) As Long
```

5 On définie la variable nécessaire au bon fonctionnement de la capture vidéo. On lui donne un nom, son positionnement, sa taille...

6
```
mCapHwnd = capCreateCaptureWindow("Fenetre de capture", &H50000000, 85, 30, 352, 288,
Me.Hwnd, 1)
```

7 On vérifie qu'il y a un périphérique connecté (on regarde si 'la webcam est branché et on déclenche son usage au moyen de la su rutine suivante.

```
If SendMessage(mCapHwnd, WM_CAP_DRIVER_CONNECT, 0, 0) = 0 Then
                                                                    Msg
Box ("La camera n'est pas connéctée")  retvale = SendMessage(mCapHwnd,
WM_CAP_DRIVER_DISCONNECT, 0, 0)
  DestroyWindow (mCapHwnd)  End If
```

8 On définit la fréquence de rafraichissement de la prévisualisation ;

```
SendMessage mCapHwnd, WM_CAP_SET_PREVIEWRATE, 50, 0
```

À ce stade nous pouvons dire que le premier problème a été résolu et on peut voir la vidéo sur la fenêtre image de notre interface (cote serveur).

Pour résoudre les deux problèmes qui restent, nous prenons en compte le fait que l'image est composé de petits points appelés pixels (en anglaise *picture element*) et que chaque pixel est défini par ses abscisses et ordonnées.

Par ailleurs La persistance rétinienne moyenne est d'environ 0,1 seconde. Dans ces conditions, si le temps entre l'illumination du premier point en haut de limage (pixel) à gauche et celle du dernier point en bas à droite est inférieure à 0,1 seconde, l'œil verra une image et non pas un point. En conséquence, le nombre d'images doit être d'au moins 10 images/s. Cependant, l'impression de mouvement fluide n'est obtenue qu'à partir de 16 images/secondes (cinéma muet). Pour avoir une reproduction du son correcte, le cinéma sonore a nécessité d'augmenter le nombre d'images et de le porter à 24 images/seconde (longueur physique du film pour l'empreinte sonore).

Le signal vidéo lorsqu'il est analogique est converti en paquets de données numériques qui seront plus faciles à transporter sur un réseau. Un des atouts majeurs du format numérique est notamment la non-dégradation du signal.

Le principe fondamental de la compression vidéo est de réduire autant que possible les redondances d'informations dans les données, sans entraîner de défauts trop perceptibles pour l'œil humain. Toute la difficulté est là, dans le dosage entre un taux de compression qui s'améliore en même temps que la qualité d'image devient médiocre.

Une séquence d'images vidéo contient une certaine quantité de redondance spatiale.

Il est possible de représenter ou d'encoder cette information sous une forme plus compacte qui élimine cette redondance.

Une séquence vidéo en mouvement contient une redondance temporelle (les trames successives sont habituellement très semblables). Il est donc possible de compresser efficacement en n'envoyant seulement que la partie de l'image qui a varié par rapport à l'image précédemment envoyée. Généralement, le changement entre images est dû aux mouvements dans un décor qui peut être considéré comme en mouvement linéaire simple. En prédisant les mouvements de certaines régions par rapport aux images précédemment envoyées et en ne transmettant que les parties qui ont varié dans les images et non leur intégralité, la quantité d'information vidéo transmise est considérablement diminuée.

La redondance spatiale est celle dans chaque image prise indépendamment des autres. On peut diminuer cette redondance en codant chaque image séparément en JPEG.

On peut aller plus loin en constatant que deux images qui se suivent dans une séquence vidéo sont quasiment identiques : c'est la redondance temporelle. Le but est alors de stocker que ce qui est modifié lors du passage d'une image à une autre.

Les images ainsi compressées sont de deux types :

1 les images I (intracodées),
2 les images P (prédictives).

Les images I sont des images complètes codées en JPEG, on les appelle aussi images clés. Les images P, par contre, ne contiennent que les pixels modifiés par rapport à l'image précédente, qui peut être elle-même une image I ou une image P.

Autre méthode consiste à jouer sur la taille de l'image. Ainsi, une image de dimension 320 × 240 ne compte qu'un quart des pixels composant une image de dimension 640 × 480. Une séquence vidéo comprenant 12 images par seconde occupera un espace moindre qu'une séquence de 25 images par seconde.

Donc après avoir cité les résolutions nous expliquons les points principaux de notre programme personnel. Toutes les fonctions et les procédures principales que nous avons utilisé, sont décrites dans ce qui suit :

- Capturer une image et coder en JPEG (Nous trouvons le JPEG plus utilisé dans les secteurs de l'informatique, de la communication, etc.) puis l'enregistrer dans un répertoire, tous cela s'effectue par les instructions suivantes :

```
SavePicture EnleverImageCam, "C:\serveur\jpg.jpeg"
Function EnleverImageCam() As Picture
On Error Resume Next
    SendMessage mCapHwnd, WM_CAP_GRAB_FRAME, 0, 0
```

```
    SendMessage mCapHwnd, WM_CAP_EDIT_COPY, 0, 0
    Set EnleverImageCam = Clipboard.GetData

End Function

Open "C:\serveur\jpg.jpeg" For Binary As #2

    Set image = LoadPicture("C:\serveur\jpg.jpeg")

Picture1.Cls

    Picture1.PaintPicture image, 0, 0, Picture1.ScaleWidth, Picture1.ScaleHeight
```

Après la capture et la sauvegarde d'images nous envoyons image par image ou plus précisément paquet par paquet.

- Du côte serveur nous ouvrons l'image enregistrée dans le répertoire par l'instruction ;

```
Open "C:\serveur\jpg.jpeg" For Binary As #2
```

- Le reste de la procédure d'envoi est semblable à la procédure relatives aux textes avec certaines différent dans le nomination.

```
Sub EnvDonCam(Data As String)
If Cam.State = 7 Then Cam.SendData Data & Ettiquet
```

- Procédure d'envoi d'image paquet par paquet

```
Sub ExcEnvDon(Data As String)
 Static ETAT, Photo As String
 Dim L3 As String
 L3 = Left(Data, 3)
                If ETAT = 0 And Cam1.Enabled = False Then
                  SavePicture EnleverImageCam, "C:\serveur\jpg.jpeg"
                   Open "C:\serveur\jpg.jpeg" For Binary As #2
                   MonImage
                   ETAT = 1
                   EnvDonCam "NOU"
Exit Sub
               ElseIf ETAT = 1 Then
                   Photo = String(LOF(2), " ")
                   ETAT = 2
                  Get #2, , Photo
```

```
                    EnvDonCam "ENC" & Photo
Exit Sub
                ElseIf ETAT = 2 Then
                    Close #2
                    EnvDonCam "FIN"
                    ETAT = 0
Exit Sub
   End If   End Sub
End Sub
```

- Attendre un événement pour exécution d'envoyer le baquet suivant `Private Sub Cam_DataArrival(ByVal bytesTotal As Long)`

```
Static temp As String
Dim VAR() As String, Data As String  Cam.GetData Data
temp = temp & Data
If Right(temp, Len(Ettiquet)) <> Ettiquet Then Exit Sub
                VAR = Split(temp, Ettiquet)
                  ExcEnvDon VAR(1)
                temp = ""
End Sub
```

Du côte client dès que les données sont reçues les procédures suivantes s'exécutent :

- Réception de donnée :

```
Private Sub Cam_DataArrival(ByVal bytesTotal As Long)  Static temp As String
Dim VAR() As String, Data As String  Cam.GetData Data
  temp = temp & Data
 If Right(temp, Len(Ettiquet)) <> Ettiquet Then Exit Sub
                    VAR = Split(temp, Ettiquet)
                    ExcEnvFlag VAR(0)
                    temp = ""
End Sub
```

- Envoie des accusées de réception de chaque paquet ; `Private Sub ExcEnvFlag(Data As String)`

```
Dim F As String  F =
Left(Data, 3)
 If F = "NOU" Then
                On Error Resume Next
                Kill "C:\client\Image.jpg"
```

- Ouvrir un fichier .JPEG dans réperatoire

```
                Open "C:\client\Image.jpg" For Binary As #2
 ElseIf F = "ENC" Then
  On Error Resume Next
                Data = Right(Data, Len(Data) - 3)
```

- Poser le paquet reçu dans le fichier ouvert

```
                Put #2, , Data
 ElseIf F = "FIN" Then
                Close #2
                ImageServeur
 End If
 If Cam.State = 7 Then
   EnvDaccord "DAC"
 End If  End Sub
Private Sub EnvDaccord(Data As String)
If Cam.State = 7 Then Cam.SendData Data & Ettiquet  End Sub
```

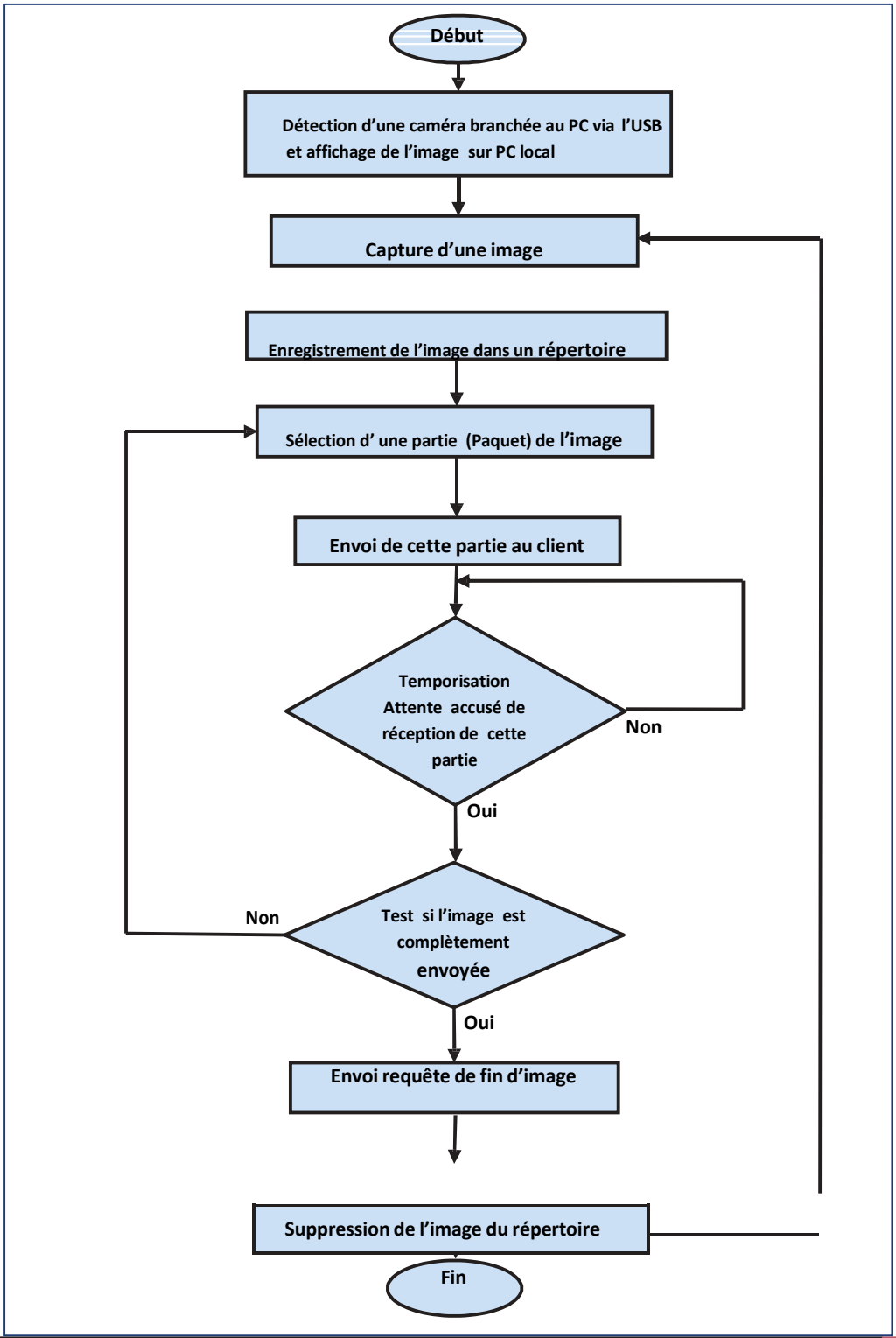
- Affichage et suppression d'image du répertoire pour remplacement par l'image suivante ; `Private Sub ImageServeur()`

```
Dim imag As Picture  On Error
Resume Next
   Set imag = LoadPicture("C:\client\Image.jpg")
   Picture1.Cls
   Picture1.PaintPicture imag, 0, 0, Picture1.ScaleWidth, Picture1.ScaleHeight
                If Photo.Enabled = True Then Exit Sub
                Photo.Enabled = True
            End Sub
```

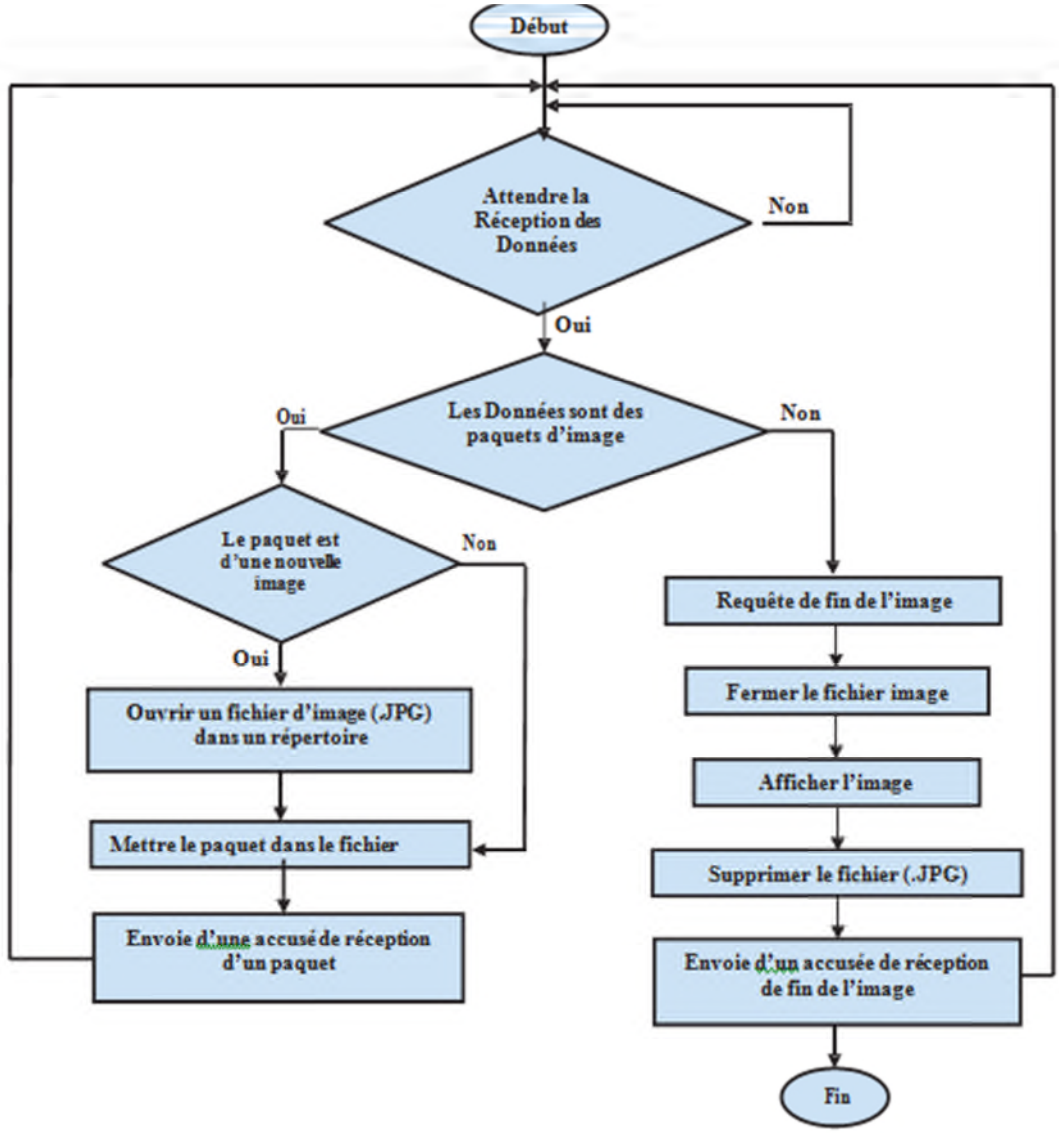
Il y a ensuite une exécution cyclique de cette procédure à de chaque côte.

l'organigramme suivant explique ce fonctionnement ;

Coté serveur (Patient)



Coté client (Médecin)



II.4.4.2 0rganigramme et procédure pour le signal vocal

La première partie à décrire dans cette section concerne les problèmes qu'ont peut retrouver dans le développement de l'interface de la transmission vocale.

Le premier problème à résoudre est celui de la communication de l'interface avec le microphone à travers la carte son, ou a travers le port USB, pour les mêmes raisons citées précédemment dans le cas du transfert vidéo.

Le deuxième problème réside dans la très forte contrainte temporelle dûe à l'interaction entre individus. Le temps de latence doit être inférieur à 300 ms si l'on veut garder une interaction humaine acceptable. Si l'on souhaite une bonne qualité de la conversation, la latence ne doit pas dépasser 150 ms

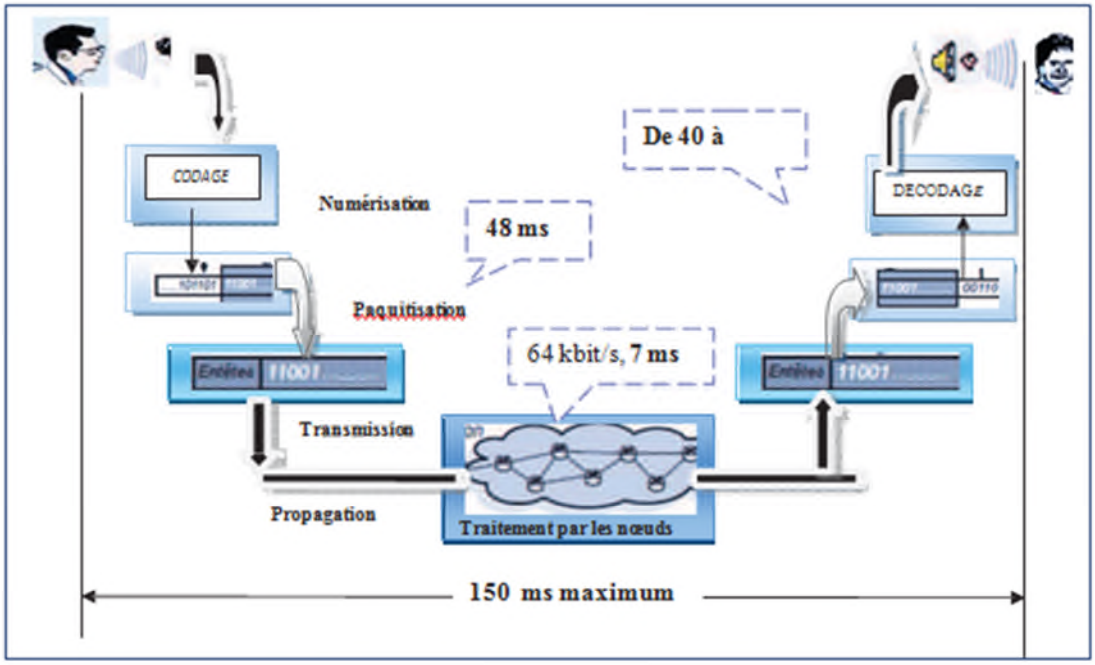


Figure VII. 14: Temps de transfert d'un paquet

De la même manière que pour le transfert vidéo nous faisons appel au DLL.

Pour le son nous avons mis en œuvre les des DLL intégrées dans le système d'exploitation Windows.

Ces DLL sont :

winmm.dll, Kernel32.dll et user32.dll.

Nous pouvons accéder à l'API Windows (ou à d'autres DLL externes) en déclarant les procédures externes dans notre application Visual Basic. Une fois que nous avons déclaré une procédure, nous pouvons l'utiliser comme n'importe quel autre élément du langage compris dans le produit.

Les procédures externes les plus couramment utilisées sont celles que constituent Microsoft Windows lui-même. L'API Windows contient des milliers de fonctions, de procédures, de types et de constantes que nous pouvons déclarer et utiliser dans nos projets.

Le fichier Win32api.txt, qui se trouve dans le sous-dossier \Winapi du dossier principal Visual Basic, contient des déclarations pour un grand nombre des procédures de l'API Windows couramment utilisées en Visual Basic. Pour utiliser une fonction, un type, ou un autre élément à partir de ce fichier, nous copions celle-ci dans notre module Visual Basic. Nous pouvons visualiser et copier des procédures de Win32api.txt en utilisant l'application Visionneuse d'API, ou en ouvrant le fichier avec n'importe quel éditeur de texte.

Note :

L'API Windows contient une grande quantité de code. Pour trouver les informations de référence et d'autres détails sur les procédures incluses dans ce jeu d'API, nous nous reportons sur les fichiers Win32 SDK, qui se trouve sur le MSDN (Microsoft Developer Network)

Dans notre projet on utilise les fonctions API suivantes :

- Les fonctions d'utilisation des formats du son /

```
Declare Function waveInOpen Lib "winmm.dll" (lphWaveIn As Long, ByVal uDeviceID As Long, lpFormat As WAVEFORMATEX, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Long
```

```
Declare Function waveInPrepareHeader Lib "winmm.dll" (ByVal hWaveIn As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveInUnprepareHeader Lib "winmm.dll" (ByVal hWaveIn As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveInAddBuffer Lib "winmm.dll" (ByVal hWaveIn As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveInStart Lib "winmm.dll" (ByVal hWaveIn As Long) As Long Declare Function waveInStop Lib "winmm.dll" (ByVal hWaveIn As Long) As Long Declare Function waveInReset Lib "winmm.dll" (ByVal hWaveIn As Long) As Long Declare Function waveInClose Lib "winmm.dll" (ByVal hWaveIn As Long) As Long Declare Function waveOutOpen Lib "winmm.dll" (lphWaveOut As Long, ByVal uDeviceID As Long, lpFormat As WAVEFORMATEX, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Long

Declare Function waveOutPrepareHeader Lib "winmm.dll" (ByVal hWaveOut As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveOutUnprepareHeader Lib "winmm.dll" (ByVal hWaveOut As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveOutWrite Lib "winmm.dll" (ByVal hWaveOut As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveOutClose Lib "winmm.dll" (ByVal hWaveOut As Long) As Long Declare Function waveOutReset Lib "winmm.dll" (ByVal hWaveOut As Long) As Long Declare Function waveOutPause Lib "winmm.dll" (ByVal hWaveOut As Long) As Long Declare Function waveOutRestart Lib "winmm.dll" (ByVal hWaveOut As Long) As Long
```

- Les fonctions d'utilisation De la mémoire ;

```
Declare Function GlobalAlloc Lib "kernel32" (ByVal wFlags As Long, ByVal dwBytes As Long) As Long

Declare Function GlobalFree Lib "kernel32" (ByVal hMem As Long) As Long Declare Function
GlobalLock Lib "kernel32" (ByVal hMem As Long) As Long Declare Function GlobalUnlock Lib
"kernel32" (ByVal hMem As Long) As Long

Declare Sub CopyPTRtoBYTES Lib "Kernel32.dll" Alias "RtlMoveMemory" (ByRef ByteDest As Byte, ByVal
PtrSrc As Long, ByVal length As Long)

Declare Sub CopyBYTEStoPTR Lib "Kernel32.dll" Alias "RtlMoveMemory" (ByVal PtrDest As Long, ByRef
ByteSrc As Byte, ByVal length As Long)
```

- Les fonctions de codage ACM

```
Public Declare Function acmStreamOpen Lib "MSACM32" (hAS As Long, ByVal hADrv As Long, wfxSrc As
WAVEFORMATEX, wfxDst As WAVEFORMATEX, ByVal wFltr As Long, ByVal dwCallback As Long, ByVal
dwInstance As Long, ByVal fdwOpen As Long) As Long

Public Declare Function acmStreamClose Lib "MSACM32" (ByVal hAS As Long, ByVal dwClose As Long) As
Long

Public Declare Function acmStreamPrepareHeader Lib "MSACM32" (ByVal hAS As Long, hASHdr As
ACMSTREAMHEADER, ByVal dwPrepare As Long) As Long

Public Declare Function acmStreamUnprepareHeader Lib "MSACM32" (ByVal hAS As Long, hASHdr As
ACMSTREAMHEADER, ByVal dwUnPrepare As Long) As Long

Public Declare Function acmStreamConvert Lib "MSACM32" (ByVal hAS As Long, hASHdr As
ACMSTREAMHEADER, ByVal dwConvert As Long) As Long

Public Declare Function acmStreamReset Lib "MSACM32" (ByVal hAS As Long, ByVal dwReset As Long) As
Long
```

Public Declare Function acmStreamSize Lib "MSACM32" (ByVal hAS As Long, ByVal cbInput As Long, dwOutBytes As Long, ByVal dwSize As Long) As Long

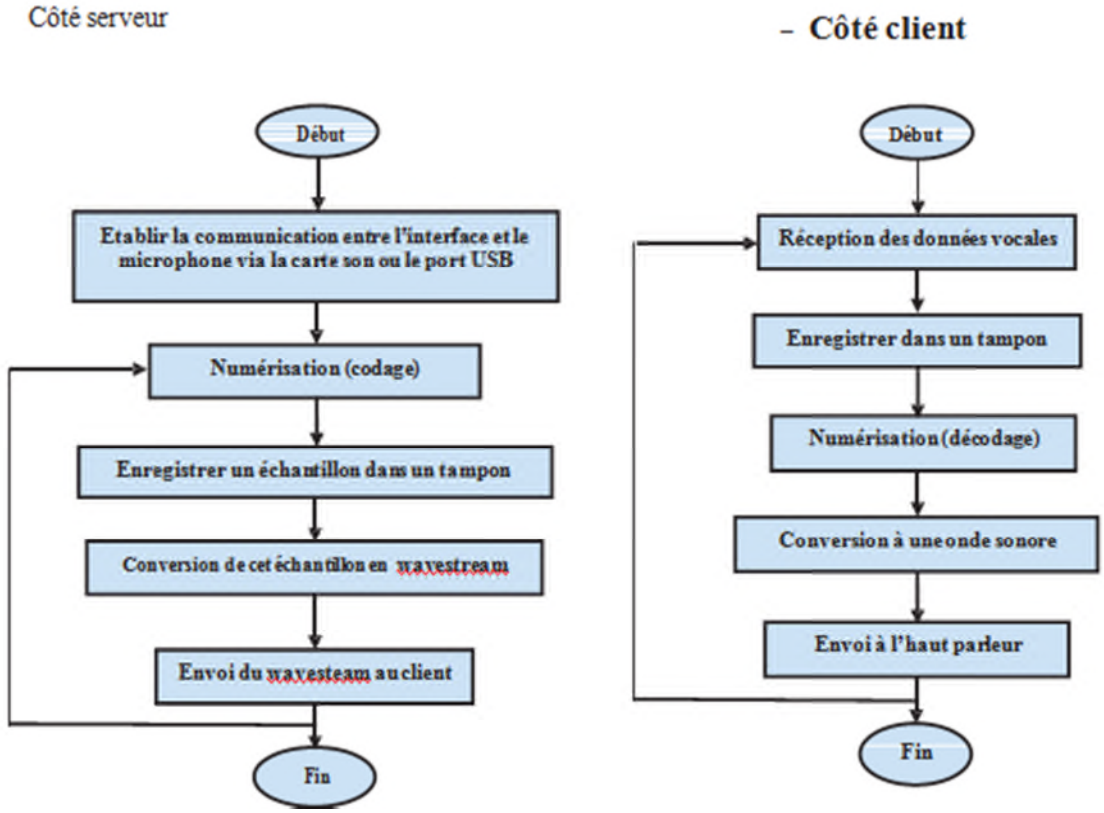
La fonction d'ouvrir des formats du son

Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal SoundData As Any, ByVal uFlags As Long) As Long

Le composant le plus important que nous avons trouvé est un module de classe dénommé wavestream.cls qui nous a facilité la procédure d'acquisition et de transmission des signaux vocaux sous forme wavestream, pour être transmis via TCP/IP (puisque TCP/IP support ce genre

de données.)

L'organigramme suivant montre les étapes de transmission de son :



II.4.5 Organigramme et procédure du transfert de fichier

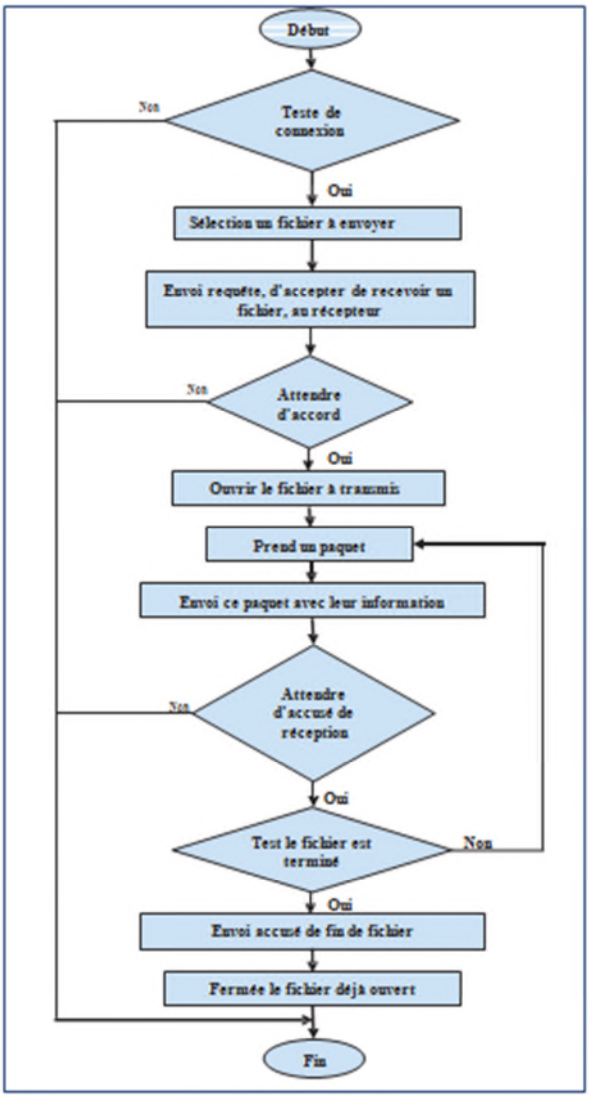
Ainsi, un transfert de fichiers correspond à un flux binaire constant. Il requiert un débit relativement important mais est très peu sensible au temps de transmission.

Pour le transfert de fichier nous avons utilisé le principe de partager ce fichier en segment de taille déterminée et puis envoyer chaque segment au récepteur tout seul, pour être regroupé coté récepteur au fichier original.

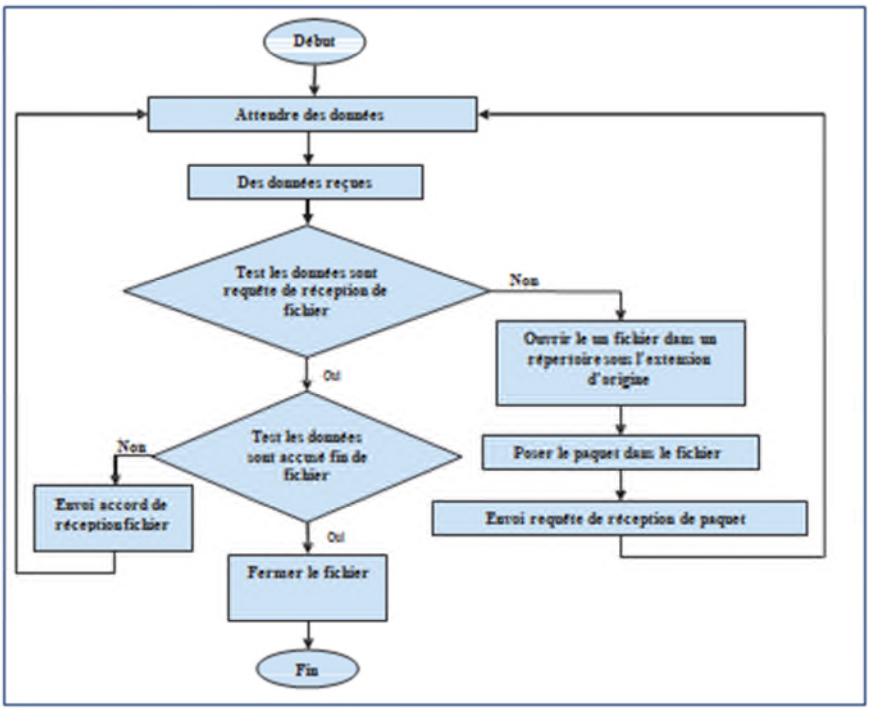
Nous n'avons utilisé aucune API dans cette partie.

Toutes les procédures d'envoi du fichier sont décrites dans l'organigramme suivant ;

Côté serveur



Côté client


Début
Attendre des données
Des données reçues
Test les données sont requête de réception de fichier
Non
Ouvrir le un fichier dans un répertoire sous l'extension d'origine
Oui
Test les données sont accusé fin de fichier
Non
Poser le paquet dans le fichier
Envoi accord de réception fichier
Envoi requête de réception de paquet
Oui
Fermer le fichier
Fin

III RESULTATS SOUS PROTOCOLE TCP/IP

l'interface de communication multimédia sous environnement Visualll Basic comprend :

1 – Une liaison textuelle.

2 - Une liaison vocale

3 – Une liaison vidéo

4- Un transfert de fichiers multi formats.

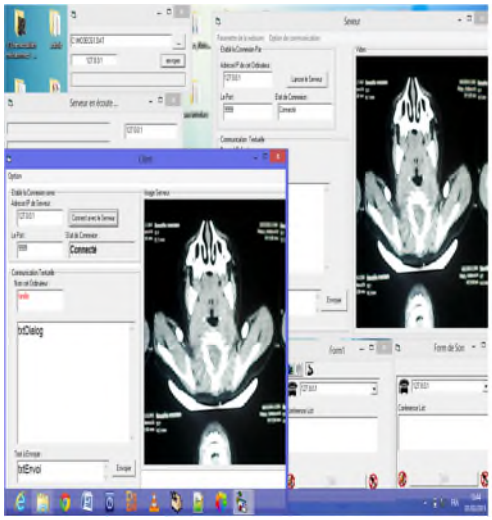


Figure VII. 15 : transfert d'images

La figure VII5 montre un exemple de transfert d'images scannographiques.

Figure VII. 15:Transfert d'images

CONCLUSION GENERALE

CONCLUSIONS

Ce travail est axé principalement autour d'un développement soft_hard qui met à la disposition des praticiens une plateforme dédiée à la télémédecine.

Le dispositif développé nécessite un terminal d'ordinateur qui contrôle les périphériques biométriques et héberge les données recueillies sur le patient et les applications spatio-temporelle spectro-traitement de ces données.

Les différents traitements qui évaluent les paramètres stochastiques pertinentes telles que les fonctions d'intercorrélation, densités interspectrales d'énergie, objectifs analyse morphologique peut être mis en œuvre selon les besoins exprimés par les médecins en relation avec des spécialistes.

Ces différents traitements sont rendus possibles grâce au protocole USB-HID qui permet l'archivage des données et le protocole TCP-IP qui permet de transmettre ces données et la mise en œuvre des différents aspects de la télémédecine: la téléconsultation, de télédiagnostic, télé-expertise et télésurveillance.

En outre, le traitement numérique des divers signaux physiologiques donnera au praticien de la télémédecine une plate-forme précieuse pour l'aide au diagnostic.

REFERENCES

Références

[1] « ETUDE N° EP 02-17 » CNPP, expert en prévention et en maîtrise des risques – www.cnpp.com.

[2] Human Interface Devices: Using Control and InterruptTransfers. http://automatepc.fr/?page=BusUsb.

[3] http://www.roboticus.org/index.php?option=com_content&view=article&id=9:carte-dinterface-sur-port-usb-protocole-hid&catid=2:electronique&Itemid=2.

[4] PIC18F2550 – informations au www.Microchip.com (datasheet, help forums, etc).

[5] Les fonctions d'API :http://vb.developpez.com/bidou/vb-api/#LI.

[6] Visual Basic TCP Client/server http://tcp.oflameron.com/tcp.htmM. Shell. (2002) IEEEtranhomepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[7] Guy PUJOLLE : Protocoles de transmission de données, Professeur à l'Université de Versailles, DOSSIER Techniques de l'Ingénieur (22/04/2012).

[8] DAVID ROS : Protocole de transport TCP, Maître de conférences à l'École nationale supérieure des télécommunications (ENST) de Bretagne, DOSSIER Techniques de l'Ingénieur (22/04/2012).

[9] VIKAS SINGH, Telemedicine& Mobile Telemedicine System: AnOverview:Health Information SystemsDepartment of Health Policy and Management, University of Arkansas for Medical Sciences (2006)

[10] Livre RÉSEAUX ET TÉLÉCOMS (Cours et exercices corrigés), Claude Servin (Chargé de cours au CNAM de Paris et en écoles d'ingénieur Ancien responsable télécom au ministère de la Défense), DUNOD.

[11] N. HAMLILl, M. BENABDELLAH, ―Implementation of Telemedical Network: Application for Health Smart Home,‖IJERD,Volume 7, Issue 4 (May 2013), PP. 24-31.

[12] Implementation of a software of digital acquisition, processing , and transmission of the one-dimensional signal in Telemedicine. M.Benabdellah., S.Rerbal, A.MezianeTani.A.Nemmiche. Applications of Broadband Optical and Wireless Network. Georgi Graschew, Peter M..Shlag. 8/2002. Editor proceedings of SPIE Volume 4912. p. 37-46.

[13] Internet embarqué_ Systèmes embarqués_ © pk-enseirb-2002.mht.

[14] Hand book of télémédecine, Olga Ferrer-Rocca Marcello Sosa. Indicissa ISO 1998

[15] Patient monitoring system using 16F877. Electronics and Communication Engineering Copyrights : Mahesh Bharath,2005.

[16] Monirorage respiratoire. Pr Jean-Pierre HABERER. Hôtel-Dieu, Paris. http://feea.net

[17] Reflectance Pulse OximetrySensor for the Electronic Patch. M.Sc. Thesis. Rasmus G. Haahr. November 2006.

[18] E-Health, Telehealth, and **Telemedicine**: A Guide to Startup and **...**, http://downloadbookz.com/e-health-telehealth-and-telemedicine-a-guide-to-startup-and-success-jossey-bass-health-series.html.

[19] http://www.abcelectronique.com/acquier/

[20] http://www.usb.org

[21] R.DUBOIS, Application des nouvelles méthodes d'apprentissage à la détection précoce d'anomalies en électrocardiographie,THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS 6 , Soutenue le 27 janvier 2004. Disponible sur le net : http://www.neurones.espci.fr/Theses_PS/DUBOIS_R/Chapitre1.pdf

[22] A. SABATIER « Études sur le coeur et la circulation centrale dans la série des vertébrés: anatomie et physiologie comparées, philosophie naturelle », ed. : C. Coulet, Université d'Oxford, 26 mai 2006.

[23] H. V. PIPBERGER, M.C. Manus & H. A. Pipberger, Methodology of ECG interpretation in the AVA program, 1990 Sep;29(4):337-40.

[24] L.ROUGEN, J.P. SABATIE. L'électrocardiogramme. La courbe électrocardiographique, le
vocabulaire du cardiologue. Ed: Documentation Médicale Labaz.

[25] A. ELLRODT, «Urgences Médicales, 5Édition», Édition ESTEM, 89, boulevard Auguste Blanqui, 75013, Paris, 04/2003. ISBN: 978 2 84371 335 4. http://www.scribd.com/doc/40469809/Urgences‐medicales.

[26] R. SLAMA, G. MOTTE, C. SEBAG, «Aide mémoire de Rhythmologie», Edition Flammarion, Paris, 04/2003. ISBN 10 : 22 57124065. ISBN 13:9782257124067. http://www.unitheque.com/medecine/Rythmologie-2385.html

[27] E. P. CHAN THE, «INTÉRÊTS ET LIMITES DE L'ÉVALUATION DE LA CHARGE DE TRAVAIL À L'AIDE DES ÉCHELLES DE BORG», Thèse Présentée et soutenue publiquement dans le cadre du troisième cycle de Médecine Spécialisée pour obtenir le grade de Docteur en Médecine, Université Henri Poincaré, Nancy 1, Faculté de Médecine de Nancy, France, 24 Octobre, 2002.

[28] J‐P. BASSAND, «Introduction à la pathologie cardiaque et vasculaire», Cours de Professeur à l'université de Besançon, France, 25 Octobre 2005. http://www.besancon‐cardio.org/cours/01‐intro.php

[29] P. HERVE, S. PHAM, EXPLORATIONS FONCTIONNELLES RESPIRATOIRES, MAPAR 1998

[30] M. BAKEHE, «Système Cardiovasculaire I», ISBN-10: 1479763616,ISBN-13: 978-1479763610,Février 6, 2013

[31] P C MATHUR, A. SOMANI, Effects in Oxygen saturation (SpO2) and Heart Rate (HR) In
Hemiparetic Stroke Patients On The AffectedSide, Official Journal of Indian Academy of Neurology, pp. 393-395, Volume : 7 , Issue : 2,Year : 2004.

[32] SULTER G, ELTING JW, STEWART R, DEN AREND A, DE KEYSER J. ,Continuous pulse
oximetry in acute hemiparetic stroke, Journal of Neurological Science, 2000 Oct 1;179(S 1-2):65-9.

[33] A. LEGRAND**,** BPCO : La spirométrie en médecine générale**.**RevueMedicaleBrux . 2003; 4 : A 345-9

[34] M. FEISSEL, La pléthysmographie de l'oxymètre de pouls: un ancien tracé plein d'avenir Principes et applications cliniques, Journal Elsevier, Vol 16 - N° 2 ,P. 124-131 - avril 2007.

[41] CLAUDE SERVIN ,RÉSEAUX & TÉLÉCOMS Cours avec 129 exercices corrigés 2e édition , Dunod, Paris, 2003, 2006.

[42] RM DI SCALA , l'Essentiel en Informatique & programmation, éditions Berti à Alger, Novembre 2004.

[43] GERARDO RUBINO ET LAURENT TOUTAIN, Réseaux locaux, École Nationale Supérieure des Télécommunications de
Bretagne - Campus de Rennes, Techniques de l'Ingénieur, traité Informatique.

[44] KARANJIT S.SIYAN ,TCP/IP, le campus,campusePRESS,2002.

[45] PHILIPPE LATU , Modélisations réseau ,2000,2012 , http://www.inetdoc.net.

Annexe 1: convertisseur A/D ADC0831

www.ti.com SNAS531A – MAY 2004 –
REVISED MAY 2004

ADC0831/ADC0832/ADC0834/ADC0838 8-Bit Serial I/O A/D Converters with Multiplexer Options

Check for Samples: ADC0831-N, ADC0832-N, ADC0834-N, ADC0838-N

FEATURES

voltage supplies

- **NSC MICROWIRE compatible—direct interface to COPS family processors**
- **Easy interface to all microprocessors, or operates "stand-alone"**
- **Operates ratiometrically or with 5 V_{DC} voltage reference**
- **No zero or full-scale adjust required**
- **2-, 4- or 8-channel multiplexer options with address logic**
- **Shunt regulator allows operation with high**
- **0V to 5V input range with single 5V power supply**
- **Remote operation with serial digital data link**
- **TTL/MOS input/output compatible**
- **0.3″ standard width, 8-, 14- or 20-pin DIP package**
- **20 Pin Molded Chip Carrier Package (ADC0838 only)**
- **Surface-Mount Package**

DESCRIPTION

The ADC0831 series are 8-bit successive approximation A/D converters with a serial I/O and configurable input multiplexers with up to 8 channels. The serial I/O is configured to comply with the NSC MICROWIRE™ serial data exchange standard for easy interface to the COPS™ family of processors, and can interface with standard shift registers or µPs.

The 2-, 4- or 8-channel multiplexers are software configured for single-ended or differential inputs as well as channel assignment.

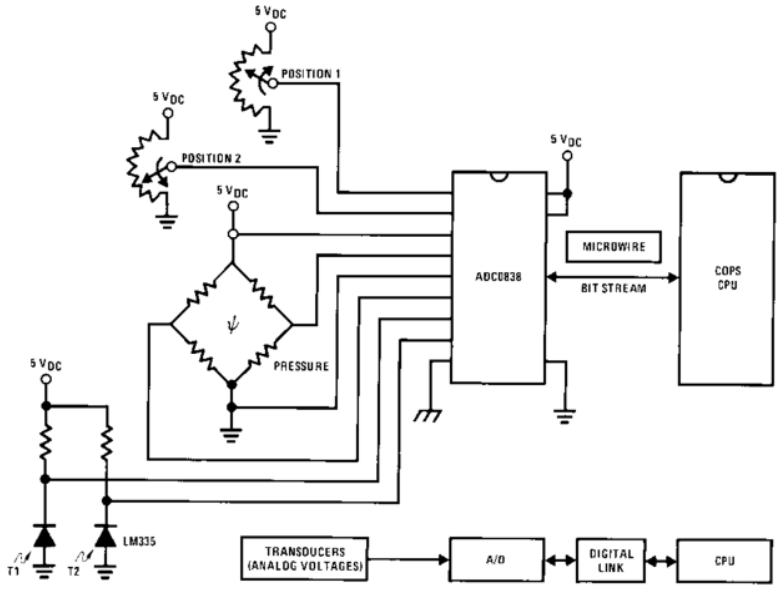
The differential analog voltage input allows increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

Table 1. Key Specifications

	VALUE	**UNIT**
Resolution	8	Bits
Total Unadjusted Error	±½ LSB and ±1	LSB
Single Supply	5 V_{DC}	
Low Power	15	mW
Conversion Time	32 µs	

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

Typical Application



Connection Diagrams

ADC0838 8-Channel Mux

Small Outline/Dual-In-Line Package (WM and N)

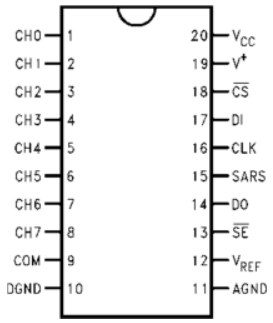
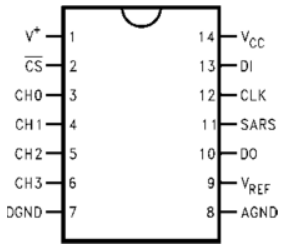


Figure 1. Top View

ADC0834 4-Channel MUX

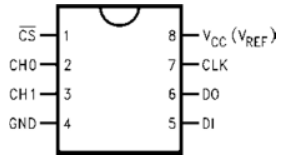
Small Outline/Dual-In-Line Package (WM and N)



COM internally connected to A GND
Top View

Figure 2. Top View

ADC0832 2-Channel MUX
Dual-In-Line Package (N)



e Package (WM)

COM internally connected to GND.
V_{REF} internally connected to V_{CC}. Top View

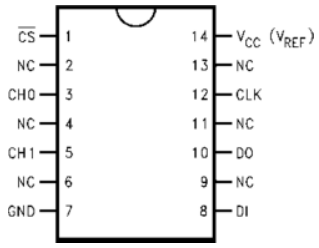


Figure 3. Top View

Figure 4. Top View

ADC0831 Single
Differential Input

Dual-In-Line Package (N)

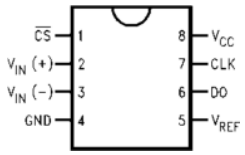
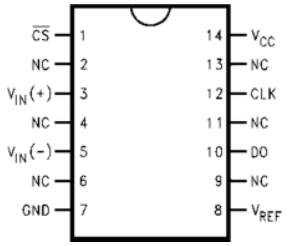


Figure 5. Top View

ADC0831 Single Differential Input

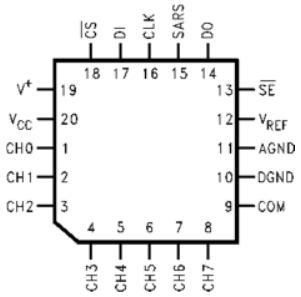
Small Outline Package (WM)



ADC0832 2-Channel MUX Small Outlin

Figure 6. Top View

Figure 7. ADC0838 8-Channel MUX
Molded Chip Carrier (PCC) Package (V)



These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

Absolute Maximum Ratings (1) (2)

Current into V^+ (3)	15 mA
Supply Voltage, V_{CC} (3)	6.5V
Voltage	
Logic Inputs	−0.3V to V_{CC} + 0.3V
Analog Inputs	−0.3V to V_{CC} + 0.3V
Input Current per Pin (4)	±5 mA
Package	±20 mA
Storage Temperature	−65°C to +150°C
Package Dissipation	
at T_A=25°C (Board Mount)	0.8W

Lead Temperature (Soldering 10 sec.)	
Dual-In-Line Package (Plastic)	260°C
Molded Chip Carrier Package	
Vapor Phase (60 sec.)	215°C

Infrared (15 sec.)	220°C
ESD Susceptibility (5)	2000V

(1) All voltages are measured with respect to the ground plugs.
(2) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.
(3) Internal zener diodes (6.3 to 8.5V) are connected from V+ to GND and V_{CC} to GND. The zener at V+ can operate as a shunt regulator and is connected to V_{CC} via a conventional diode. Since the zener voltage equals the A/D's breakdown voltage, the diode insures that V_{CC} will be below breakdown when the device is powered from V+. Functionality is therefore guaranteed for V+ operation even though the resultant voltage at V_{CC} may exceed the specified Absolute Max of 6.5V. It is recommended that a resistor be used to limit the max current into V+. (See Figure 26 in Functional Description)
(4) When the input voltage (V_{IN}) at any pin exceeds the power supply rails ($V_{IN} < V^-$ or $V_{IN} > V^+$) the absolute value of current at that pin should be limited to 5 mA or less. The 20 mA package input current limits the number of pins that can exceed the power supply boundaries with a 5 mA current limit to four.
(5) Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Operating Ratings (1) (2)

Supply Voltage, V_{CC}	4.5 V_{DC} to 6.3 V_{DC}
Temperature Range	$T_{MIN} \le T_A \le T_{MAX}$
ADC0832/8CIWM	−40°C to +85°C

ADC0834BCN,

ADC0838BCV,

(1) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.
(2) All voltages are measured with respect to the ground plugs.

Converter and Multiplexer Electrical Characteristics

The following specifications apply for V_{CC} = V+ = V_{REF} = 5V, $V_{REF} \leq V_{CC}$ +0.1V, $T_A = T_j$ = 25°C, and f_{CLK} = 250 kHz unless otherwise specified. **Boldface limits apply from T_{MIN} to T_{MAX}.**

Parameter	Conditions	CIWM Devices			BCV, CCV, CCWM, BCN and CCN Devices			Units
		Typ (1)	Tested Limit (2)	Design Limit (3)	Typ (1)	Tested Limit (2)	Design Limit (3)	
CONVERTER AND MULTIPLEXER CHARACTERISTICS								
Total Unadjusted Error	V_{REF}=5.00 V (4)							
ADC0838BCV						±½	**±½**	LSB (Max)
ADC0834BCN						±½	**±½**	
ADC0838CCV						±1	**±1**	
ADC0831/2/4/8CCN						±1	**±1**	
ADC0831/2/4/8CCWM						±1	**±1**	
ADC0832/8CIWM			**±1**					
Minimum Reference Input Resistance (5)		3.5	**1.3**		3.5	1.3	**1.3**	kΩ
Maximum Reference Input Resistance (5)		3.5	**5.9**		3.5	5.4	**5.9**	kΩ
Maximum Common-Mode Input Range (6)			**V_{CC} +0.05**			V_{CC} +0.05	**V_{CC}+0.05**	V
Minimum Common-Mode Input Range (6)			**GND −0.05**			GND −0.05	**GND−0.05**	V
DC Common-Mode Error		±1/16	**±¼**		±1/16	±¼	**±¼**	LSB
Change in zero error from V_{CC}=5V to internal zener operation (7)	15 mA into V+ V_{CC}=N.C. V_{REF}=5V		**1**			1	**1**	LSB
V_Z, internal diode breakdown (at V_+) (7) MIN	15 mA into V+		**6.3**			6.3	**6.3**	V
MAX			**8.5**			8.5	**8.5**	
Power Supply Sensitivity	V_{CC}=5V±5%	±1/16	**±¼**	±¼	±1/16	±¼	**±¼**	LSB
I_{OFF}, Off Channel Leakage Current (8)	On Channel=5V, Off Channel=0V		−0.2 **−1**			−0.2	**−1**	µA
	On Channel=0V, Off Channel=5V		+0.2 **+1**			+0.2	**+1**	µA
I_{ON}, On Channel Leakage	On Channel=0V,		−0.2			−0.2	**−1**	µA

(1) Typicals are at 25°C and represent most likely parametric norm.
(2) Tested limits are guaranteed to National's AOQL (Average Outgoing Quality Level).
(3) Guaranteed but not 100% production tested. These limits are not used to calculate outgoing quality levels.
(4) Total unadjusted error includes offset, full-scale, linearity, and multiplexer errors.
(5) Cannot be tested for ADC0832.
(6) For $V_{IN}(-) \geq V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input (see Block Diagram) which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} or V_{REF} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.
(7) Internal zener diodes (6.3 to 8.5V) are connected from V+ to GND and V_{CC} to GND. The zener at V+ can

operate as a shunt regulator and is connected to V_{CC} via a conventional diode. Since the zener voltage equals the A/D's breakdown voltage, the diode insures that V_{CC} will be below breakdown when the device is powered from V+. Functionality is therefore guaranteed for V+ operation even though the resultant voltage at V_{CC} may exceed the specified Absolute Max of 6.5V. It is recommended that a resistor be used to limit the max current into V+. (See Figure 26 in Functional Description)

(8) Leakage current is measured with the clock not switching.

Converter and Multiplexer Electrical Characteristics (continued)

The following specifications apply for V_{CC} = V+ = V_{REF} = 5V, $V_{REF} \leq V_{CC}$ +0.1V, $T_A = T_j$ = 25°C, and f_{CLK} = 250 kHz unless otherwise specified. **Boldface limits apply from T_{MIN} to T_{MAX}.**

Parameter	Conditions	CIWM Devices			BCV, CCV, CCWM, BCN and CCN Devices			Units
		Typ (1)	Tested Limit (2)	Design Limit (3)	Typ (1)	Tested Limit (2)	Design Limit (3)	
Current (8)	Off Channel=5V		**−1**					
	On Channel=5V, Off Channel=0V		+0.2 **+1**			+0.2	**+1**	µA
DIGITAL AND DC CHARACTERISTICS								
$V_{IN(1)}$, Logical "1" Input Voltage (Min)	V_{CC}=5.25V		**2.0**			2.0	**2.0**	V
$V_{IN(0)}$, Logical "0" Input Voltage (Max)	V_{CC}=4.75V		**0.8**			0.8	**0.8**	V
$I_{IN(1)}$, Logical "1" Input Current (Max)	V_{IN}=5.0V	0.005	**1**		0.005	1	**1**	µA
$I_{IN(0)}$, Logical "0" Input Current (Max)	V_{IN}=0V	−0.005	**−1**		−0.005	−1	**−1**	µA
$V_{OUT(1)}$, Logical "1" Output Voltage (Min)	V_{CC}=4.75V I_{OUT}=−360 µA I_{OUT}=−10 µA		 **2.4** **4.5**			 2.4 4.5	 **2.4** **4.5**	 V V
$V_{OUT(0)}$, Logical "0" Output Voltage (Max)	V_{CC}=4.75V I_{OUT}=1.6 mA		**0.4**			0.4	**0.4**	V
I_{OUT}, TRI-STATE Output Current (Max)	V_{OUT}=0V V_{OUT}=5V	−0.1 0.1	**−3** **3**		−0.1 0.1	−3 +3	**−3** **+3**	µA µA
I_{SOURCE}, Output Source Current (Min)	V_{OUT}=0V	−14	**−6.5**		−14	−7.5	**−6.5**	mA
I_{SINK}, Output Sink Current (Min)	V_{OUT}=V_{CC}	16	**8.0**		16	9.0	**8.0**	mA
I_{CC}, Supply Current (Max) ADC0831, ADC0834, ADC0838		0.9	**2.5**		0.9	2.5	**2.5**	mA
ADC0832	Includes Ladder Current	2.3	**6.5**		2.3	6.5	**6.5**	mA

AC Characteristics

The following specifications apply for V_{CC} = 5V, $t_r = t_f$ = 20 ns and 25°C unless otherwise specified.

Parameter		Conditions	Typ (1)	Tested Limit (2)	Design Limit (3)	Limit Units
f_{CLK}, Clock Frequency	Min			10		kHz
	Max				400	kHz
t_C, Conversion Time		Not including MUX Addressing Time		8		$1/f_{CLK}$
Clock Duty Cycle (4)	Min				40	%
	Max				60	%
t_{SET-UP}, CS Falling Edge or Data Input Valid to CLK Rising Edge					250	ns
t_{HOLD}, Data Input Valid after CLK Rising Edge					90	ns
t_{pd1}, t_{pd0}—CLK Falling Edge to Output Data Valid (5)		C_L=100 pF				
		Data MSB First	650		1500	ns
		Data LSB First	250		600	ns
t_{1H}, t_{0H},—Rising Edge of CS to Data Output and SARS Hi–Z		C_L=10 pF, R_L=10k (see TRI-STATE® Test Circuits)	125		250	ns
		C_L=100 pf, R_L=2k		500		ns
C_{IN}, Capacitance of Logic Input			5			pF
C_{OUT}, Capacitance of Logic Outputs			5			pF

(1) Typicals are at 25°C and represent most likely parametric norm.
(2) Tested limits are guaranteed to National's AOQL (Average Outgoing Quality Level).
(3) Guaranteed but not 100% production tested. These limits are not used to calculate outgoing quality levels.
(4) A 40% to 60% clock duty cycle range insures proper operation at all clock frequencies. In the case that an available clock has a duty cycle outside of these limits, the minimum, time the clock is high or the minimum time the clock is low must be at least 1 µs. The maximum time the clock can be high is 60 µs. The clock can be stopped when low so long as the analog input voltage remains stable.
(5) Since data, MSB first, is the output of the comparator used in the successive approximation loop, an additional delay is built in (see Block Diagram) to allow for comparator response time.

T

Timing Diagrams

Figure 12. Data Input Timing

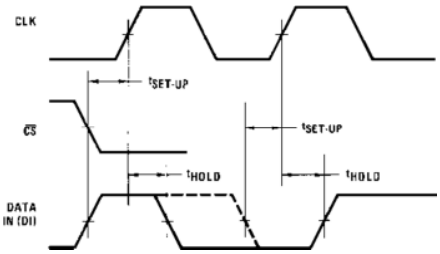


Figure 13. Data Output Timing

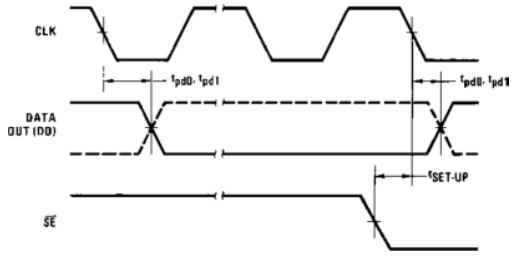


Figure 14. ADC0831 Start Conversion Timing

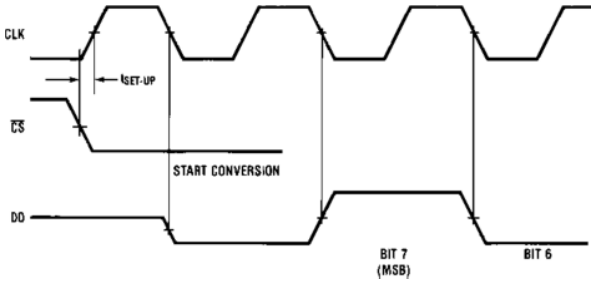
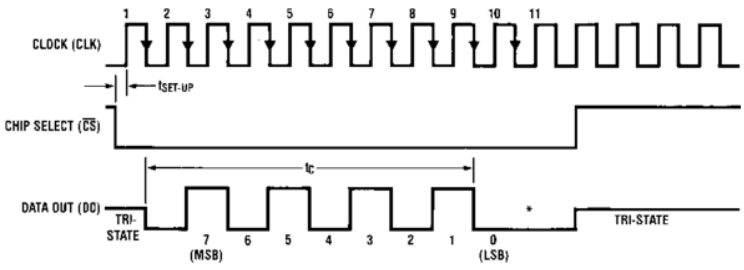


Figure 15. ADC0831 Timing



*LSB first output not available on ADC0831.

Figure 16. ADC0832 Timing

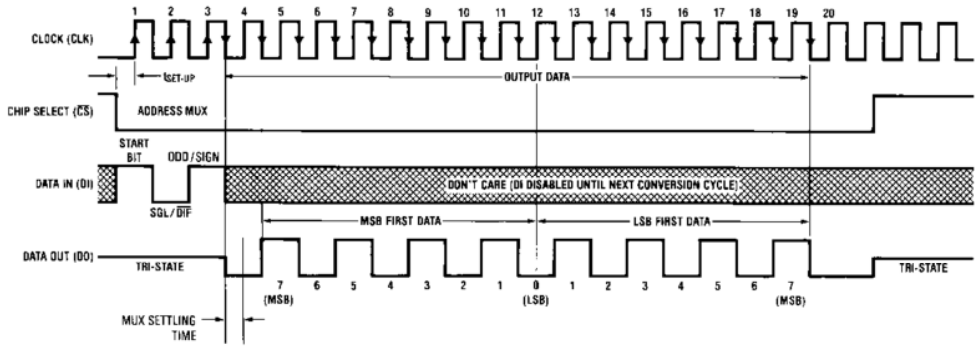


Figure 17. ADC0834 Timing

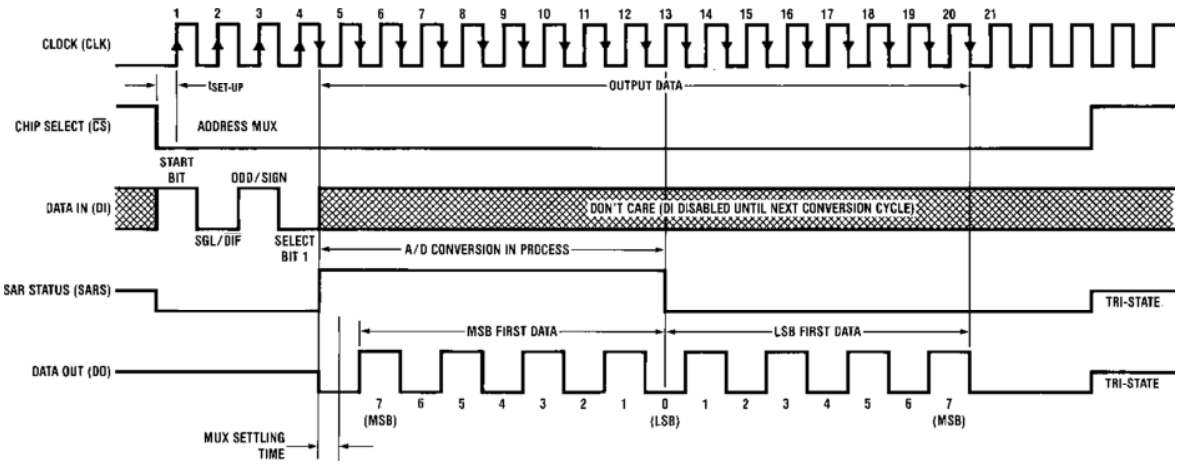
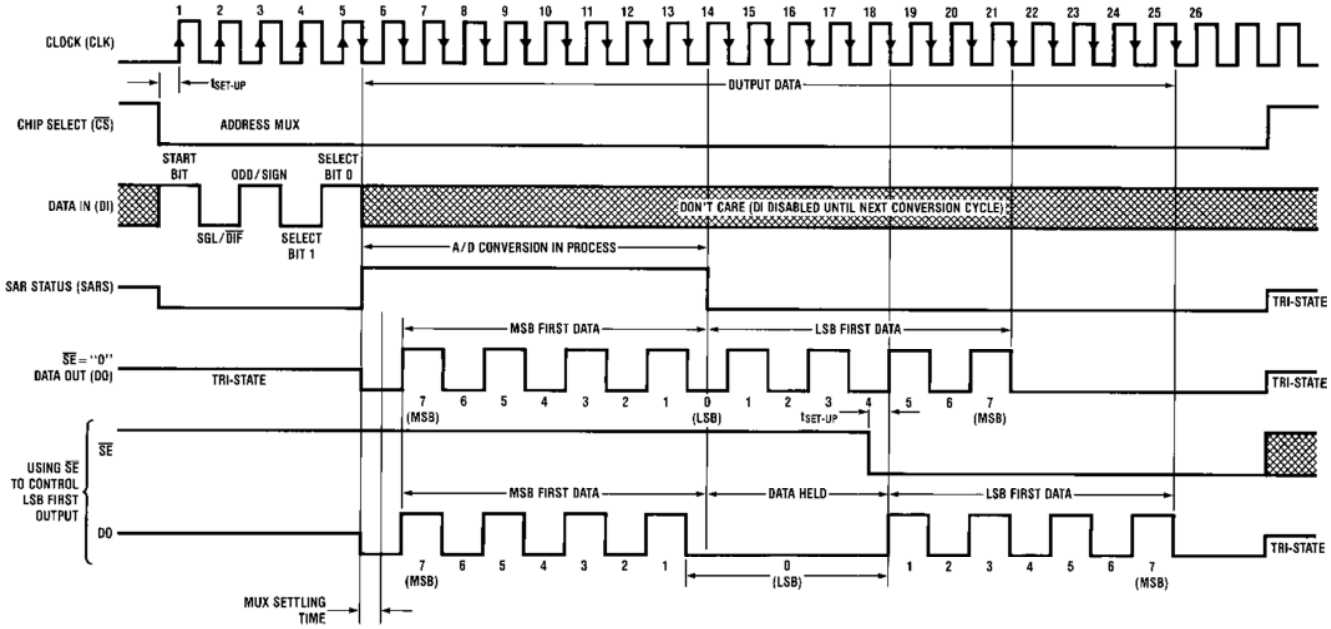
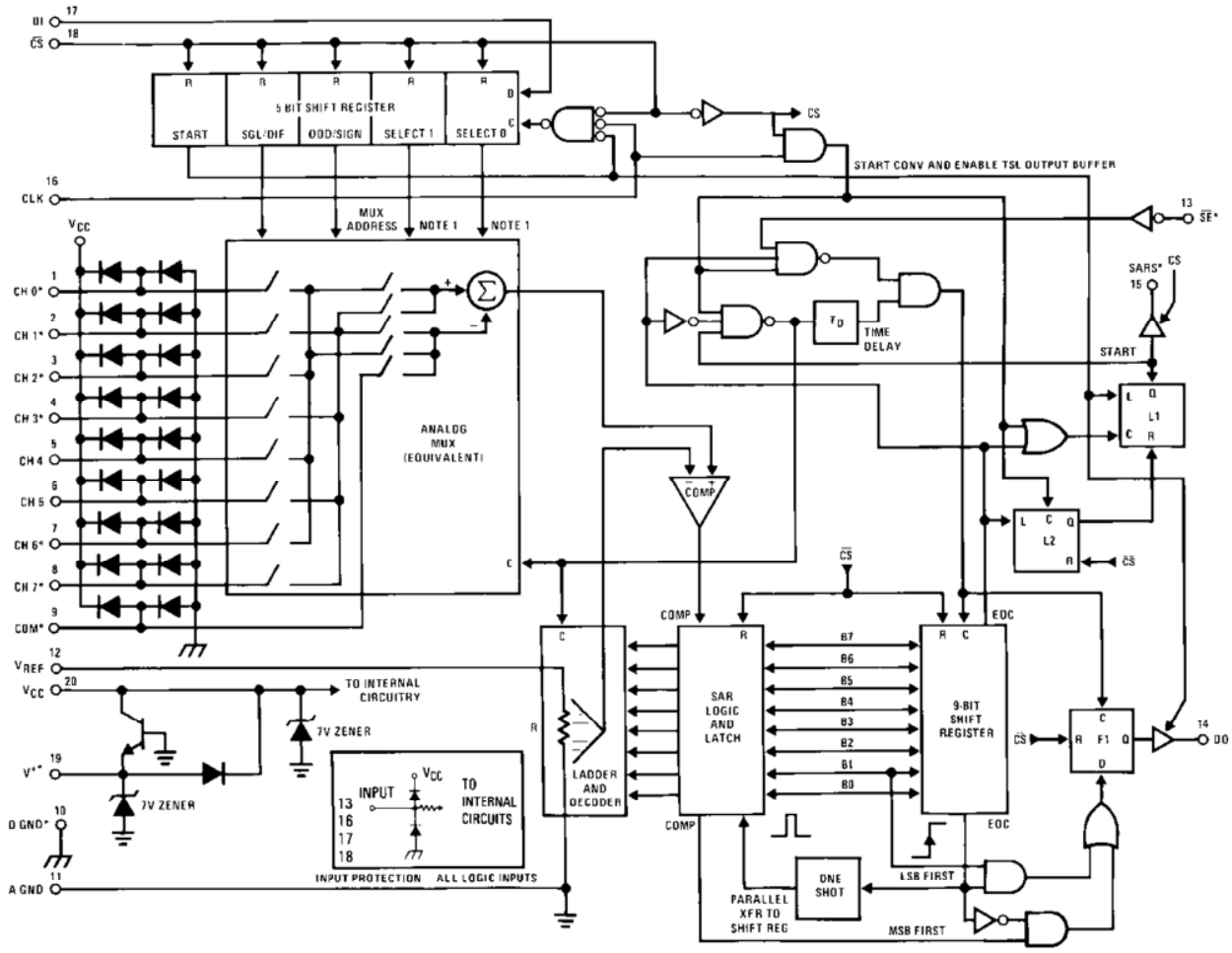


Figure 18. ADC0838 Timing



*Make sure clock edge #18 clocks in the LSB before SE is taken low

ADC0838 Functional Block Diagram



*Some of these functions/pins are not available with other options.

Note 1: For the ADC0834, D1 is input directly to the D input of SELECT 1. SELECT 0 is forced to a “1”. For the ADC0832, DI is input directly to the DI input of ODD/SIGN. SELECT 0 is forced to a “0” and SELECT 1 is forced to a “1”.

Functional Description

Multiplexer Addressing

The design of these converters utilizes a sample-data comparator structure which provides for a differential analog input to be converted by a successive approximation routine.

The actual voltage converted is always the difference between an assigned “+” input terminal and a “−” input terminal. The polarity of each input terminal of the pair being converted indicates which line the converter expects to be the most positive. If the assigned “+” input is less than the “−” input the converter responds with an all zeros output code.

A unique input multiplexing scheme has been utilized to provide multiple analog channels with software-configurable single-ended, differential, or a new pseudo-differential option which will convert the difference between the voltage at any analog input and a common terminal. The analog signal conditioning required in transducer-based data acquisition systems is significantly simplified with this type of input flexibility. One converter package can now handle ground referenced inputs and true differential inputs as well as signals with some arbitrary reference voltage.

A particular input configuration is assigned during the MUX addressing sequence, prior to the start of a conversion. The MUX address selects which of the analog inputs are to be enabled and whether this input is single-ended or differential. In the differential case, it also assigns the polarity of the channels. Differential inputs are restricted to adjacent channel pairs. For example channel 0 and channel 1 may be selected as a different pair but channel 0 or 1 cannot act differentially with any other channel. In addition to selecting differential mode the sign may also be selected. Channel 0 may be selected as the positive input and channel 1 as the negative input or vice versa. This programmability is best illustrated by the MUX addressing codes shown in the following tables for the various product options.

The MUX address is shifted into the converter via the DI line. Because the ADC0831 contains only one differential input channel with a fixed polarity assignment, it does not require addressing.

The common input line on the ADC0838 can be used as a pseudo-differential input. In this mode, the voltage on this pin is treated as the "−" input for any of the other input channels. This voltage does not have to be analog ground; it can be any reference potential which is common to all of the inputs. This feature is most useful in single-supply application where the analog circuitry may be biased up to a potential other than ground and the output signals are all referred to this potential.

Table 2. Multiplexer/Package OptionsSingle-Ended MUX Mode

Part Number	Number of Analog Channels		Number of Package Pins
	Single-Ended	Differential	
ADC0831	1	1	8
ADC0832	2	1	8
ADC0834	4	2	14
ADC0838	8	4	20

Table 3. MUX Addressing: ADC0838Single-Ended MUX Mode

MUX Address				Analog Single-Ended Channel #								
SGL/ DIF	ODD/ SIGN	SELECT 1	0	0	1	2	3	4	5	6	7	COM
1	0	0	0	+								−
1	0	0	1			+						−
1	0	1	0					+				−
1	0	1	1							+		−
1	1	0	0		+							−
1	1	0	1				+					−
1	1	1	0						+			−
1	1	1	1								+	−

Table 4. MUX Addressing: ADC0838Differential MUX Mode

MUX Address				Analog Differential Channel-Pair #							
SGL/	ODD/	SELECT		0		1		2		3	
DIF	SIGN	1	0	0	1	2	3	4	5	6	7
0	0	0	0	+	−						
0	0	0	1			+	−				

0	0	1	0					+	−		
0	0	1	1							+	−
0	1	0	0	−	+						
0	1	0	1			−	+				
0	1	1	0					−	+		
0	1	1	1							−	+

Table 5. MUX Addressing: ADC0834 Single-Ended MUX Mode

MUX Address			Channel #			
SGL/ DIF	ODD/ SIGN	SELECT 1	0	1	2	3
1	0	0	+			
1	0	1			+	
1	1	0		+		
1	1	1				+

Table 6. MUX Addressing: ADC0834 Differential MUX Mode

MUX Address			Channel #			
SGL/ DIF	ODD/ SIGN	SELECT 1	0	1	2	3
0	0	0	+	−		
0	0	1			+	−
0	1	0	−	+		
0	1	1			−	+

Table 7. MUX Addressing: ADC0832 Single-Ended MUX Mode

___ MUX Address		Channel #	
SGL/ DIF	ODD/ SIGN	0	1
1	0	+	
1	1		+

Table 8. MUX Addressing: ADC0832 Differential MUX Mode

___ MUX Address		Channel #	
SGL/ DIF	ODD/ SIGN	0	1
0	0	+	−
0	1	−	+

Since the input configuration is under software control, it can be modified, as required, at each conversion. A channel can be treated as a single-ended, ground referenced input for one conversion; then it can be reconfigured as part of a differential channel for another conversion. *Figure 23* illustrates the input flexibility which can be achieved.

The analog input voltages for each channel can range from 50 mV below ground to 50 mV above V_{CC} (typically 5V) without degrading conversion accuracy.

THE DIGITAL INTERFACE

A most important characteristic of these converters is their serial data link with the controlling processor. Using a serial communication format offers two very significant system improvements; it allows more function to be included in the converter package with no increase in package size and it can eliminate the transmission of low level analog signals by locating the converter right at the analog sensor; transmitting highly noise immune digital data back to the host processor.

To understand the operation of these converters it is best to refer to the Timing Diagrams and Functional Block Diagram and to follow a complete conversion sequence. For clarity a separate diagram is shown of each device.

1. A conversion is initiated by first pulling the ~~CS~~ (chip select) line low. This line must be held low for the entire conversion. The converter is now waiting for a start bit and its MUX assignment word.

2. A clock is then generated by the processor (if not provided continuously) and output to the A/D clock input.

17.0 UNIVERSAL SERIAL BUS (USB)

This section describes the details of USB peripheral. Because of the very specific nature of the module, knowledge of USB is high- USB informati is provide in **Section 17.10 "Overview of USB"** only for application design reference. Designers are encouraged to refer to the official specification published by the USB Imple- menters Forum (USB-IF) for the latest information. USB

Anexe 2 PIC18F2550

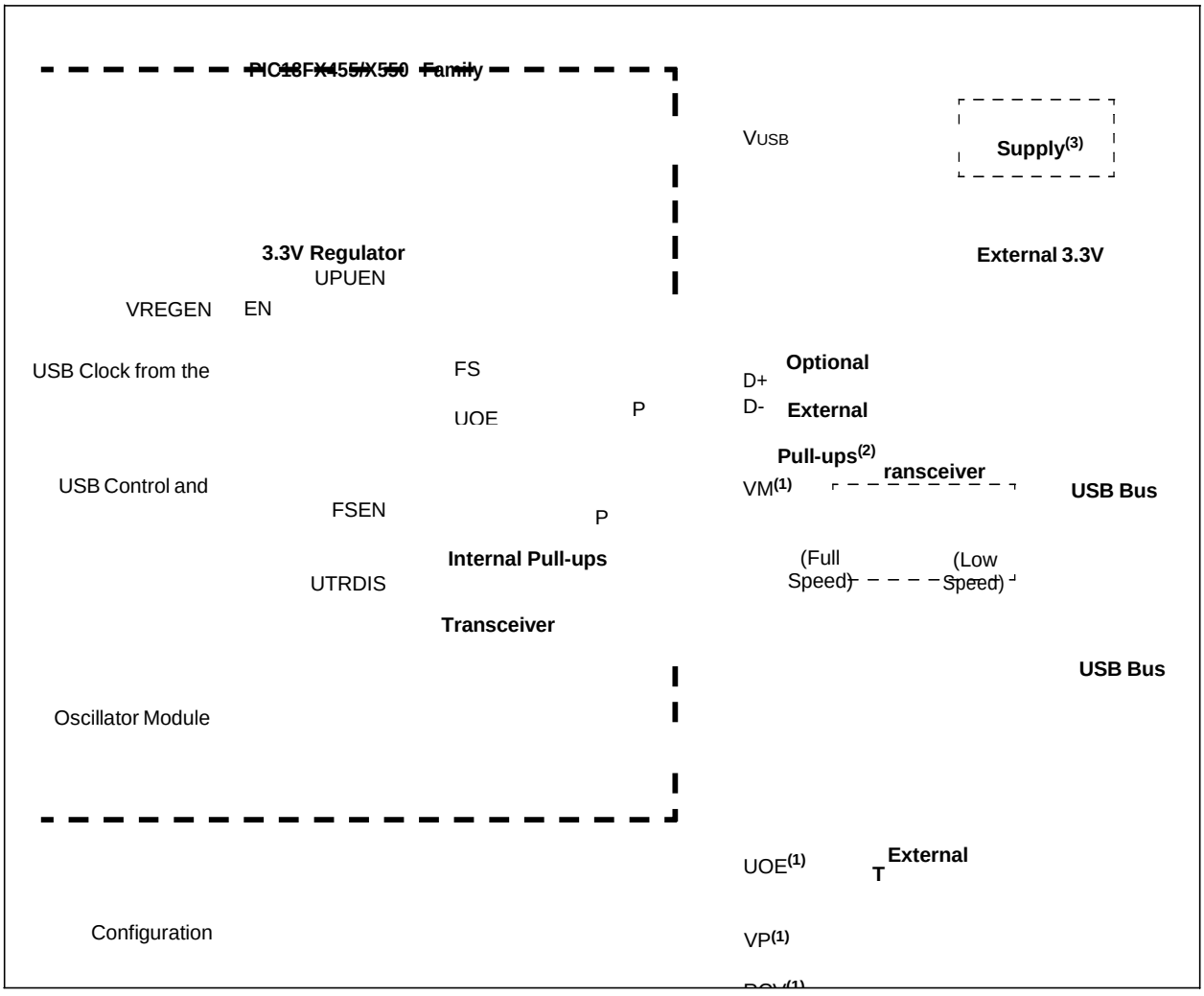
17.1 Overview of the USB Peripheral

The PIC18FX455/X550 device family contains a full-speed and low-speed compatible USB Serial Inter- face Engine (SIE) that allows fast

The SIE can be interfaced directly to the USB, utilizing the internal transceiver, or it can be connected through an external transceiver. An internal 3.3V regulator is also available to power the internal transceiver in 5V applications.

Some special hardware features have been included to improve performance. Dual port memory in the device's data memory space (USB RAM) has been supplied to share direct memory access between the microcontroller core and the SIE. Buffer descriptors are also provided, allowing users to freely program

FIGURE 17-1: USB PERIPHERAL AND OPTIONS



17.2 USB Status and Control

The operation of the USB module is configured and managed through three control registers. In addition, a total of 22 registers are

- USB Control register (UCON)
- USB Configuration register (UCFG)
- USB Transfer Status register (USTAT) USB Device Address
- register (UADDR)

17.2.1 USB CONTROL REGISTER (UCON)

The USB Control register (Register 17-1) contains bits needed to control the module

- Main USB Peripheral Enable
- Ping-Pong Buffer Pointer
- Reset Control of the

In addition, the USB Control register contains a status bit, SE0 (UCON<5>), which is used to indicate the occurrence of a single-ended zero on the bus. When the USB module is enabled, this bit should be moni- tored to determine whether the differential data lines have come out of a single-ended zero condition. This helps to differentiate the initial power-up state from the USB Reset signal.

The overall operation of the USB module is controlled by the USBEN bit (UCON<3>). Setting this bit activates the module and resets all of the PPBI bits in the Buffer Descriptor Table to '0'. This bit also activates the on-chip voltage

REGISTER 17-1: UCON: USB CONTROL REGISTER

U-0	R/W-0	R-x	R/C-0	R/W-0	R/W-0	R/W-0	U-0
—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—
bit 7							bit 0

Legend:	C = Clearable bit	
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7 **Unimplemented:** Read as '0'

PPBRST: Ping-Pong Buffers Reset bit

bit 1 = Reset all Ping-Pong Buffer Pointers to the Even Buffer Descriptor (BD) banks

bit 0 = Ping-Pong Buffer Pointers not being reset

SE0: Live Single-Ended Zero Flag bit

bit 1 = Single-ended zero active on the USB bus

bit 0 = No single-ended zero detected

PKTDIS: Packet Transfer Disable bit

bit 1 = SIE token and packet processing disabled, automatically set when a SETUP token is received

bit 0 = SIE token and packet processing enabled

USBEN: USB Module Enable bit

bit 1 = USB module and supporting circuitry enabled (device attached)

bit 0 = USB module and supporting circuitry disabled (device detached)

The PPBRST bit (UCON<6>) controls the Reset status when Double-Buffering mode (ping-pong buffering) is used. When the PPBRST bit is set, all Ping-Pong Buffer Pointers are set to the Even buffers. PPBRST has to be cleared by firmware. This bit is ignored in buffering modes not using ping-pong buffering.

The PKTDIS bit (UCON<4>) is a flag indicating that the SIE has disabled packet transmission and reception. This bit is set by the SIE when a SETUP token is received to allow setup processing. This bit cannot be set by the microcontroller, only cleared; clearing it allows the SIE to continue transmission and/or reception. Any pending events within the Buffer Descriptor Table will still be available, indicated within the USTAT register's FIFO buffer.

The RESUME bit (UCON<2>) allows the peripheral to perform a remote wake-up by executing Resume signaling. To generate a valid remote wake-up, firmware must set RESUME for 10 ms and then clear the bit. For more information on Resume signaling, see Sections 7.1.7.5, 11.4.4 and 11.9 in the USB 2.0 specification.

Note:	While in Suspend mode, a typical powered USB device is limited to 500 of current. This is the complete drawn by the PIC device and its ing circuitry. Care should be taken assure minimum current draw when device enters Suspend mode.

17.2.2 USB CONFIGURATION REGISTER (UCFG)

Prior to communicating over USB, the module's associated internal and/or external hardware must be configured. Most of the configuration is performed with the UCFG register (Register 17-2). The separate USB voltage regulator (see **Section 17.2.2.8 "Internal Regulator"**) is controlled through the Configuration registers.

- Bus Speed (full speed versus low
- speed) On-Chip Pull-up Resistor Enable

The UCFG register also contains two bits which aid in module testing, debugging and USB certifications. These bits control output enable

Note:	The USB speed, transceiver and pull-up should only be configured during the mod- ule setup phase. It is not recommended to switch these

17.2.2.1 Internal Transceiver

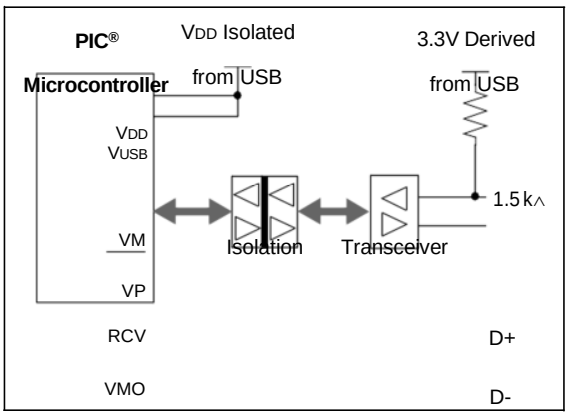
The USB peripheral has a built-in, USB 2.0, full-speed and low-speed compliant transceiver, internally con- nected to the SIE. This feature is useful for low-cost single chip applications. The UTRDIS bit (UCFG<3>) controls the transceiver; it is enabled by default (UTRDIS = 0). The FSEN bit (UCFG<2>) controls the transceiver speed; setting the bit enables full-speed operation.

The on-chip USB pull-up resistors are controlled by the UPUEN bit (UCFG<4>). They can only be selected when the on-chip transceiver is

17.2.2.2 External Transceiver

This module provides support for use with an off-chip transceiver. The off-chip transceiver is intended for applications where physical conditions dictate the loca- tion of the transceiver to be away from the SIE. For example, applications that require isolation from the USB could use an external transceiver

FIGURE 17-2: TYPICAL EXTERNAL TRANSCEIVER WITH ISOLATION



REGISTER 17-2: UCFG: USB CONFIGURATION REGISTER

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
UTEYE	UOEMON(1)	—	UPUEN(2,3)	UTRDIS(2)	FSEN(2)	PPB1	PPB0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit **UTEYE:** USB Eye Pattern Test Enable bit

1 = Eye pattern test enabled

bit 0 = Eye pattern test disabled

bit 1 = $\overline{UOE}$ signal active; it indicates intervals during which the D+/D- lines are

0 = $\overline{UOE}$ signal

bit 5 **Unimplemented:** Read as '0'

UPUEN: USB On-Chip Pull-up Enable bit(2,3)

bit 1 = On-chip pull-up enabled (pull-up on D+ with FSEN = 1 or D- with FSEN = 0) 0 = On-chip pull-up disabled

UTRDIS: On-Chip Transceiver Disable bit(2)

bit 1 = On-chip transceiver disabled; digital transceiver interface enabled

0 = On-chip transceiver active

bit 1- **FSEN:** Full-Speed Enable bit(2)

1 = Full-speed device: controls transceiver edge rates; requires input clock at 48 MHz

0 = Low-speed device: controls transceiver edge rates; requires input clock

Note 1: If UTRDIS is set, the $\overline{UOE}$ signal will be active independent of the UOEMON bit setting.

2: The UPUEN, UTRDIS and FSEN bits should never be changed while the USB module is enabled.

3: These values must be preconfigured prior to enabling the module.

There are 6 signals from the module to communicate with and control an external

- VM: Input from the single-ended D- line
- VP: Input from the single-ended D+ line
- RCV: Input from the differential receiver
- VMO: Output to the differential line driver
- VPO:

The VPO and VMO signals are outputs from the SIE to the external transceiver. The RCV signal is the output from the external transceiver to the SIE; it represents the differential signals from the serial bus translated into a single pulse train. The VM and VP signals are used to report conditions on the serial bus to the SIE

TABLE 17-1: DIFFERENTIAL OUTPUTS TO TRANSCEIVER

VPO	VMO	Bus State
0	0	Single-Ended Zero
0	1	Differential '0'
1	0	Differential '1'
1	1	Illegal Condition

TABLE 17-2: SINGLE-ENDED INPUTS FROM TRANSCEIVER

VP	VM	Bus State
0	0	Single-Ended Zero
0	1	Low Speed
1	0	High
1	1	Error

The $\overline{UOE}$ signal toggles the state of the external trans- ceiver. This line is pulled low by the device to enable the transmission of data from

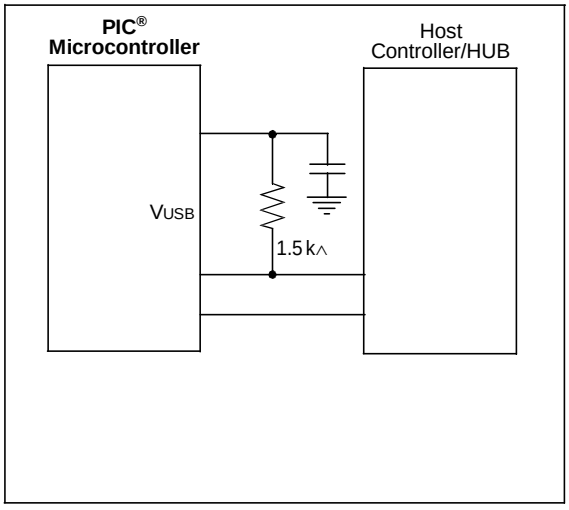
17.2.2.3 Internal Pull-up Resistors

The PIC18FX455/X550 devices have built-in pull-up resistors designed to meet the requirements for low-speed and full-speed USB. The UPUEN bit (UCFG<4>) enables the

17.2.2.4 External Pull-up Resistors

External pull-up may also be used. The V_{USB} pin may be used to pull up D+ or D-. The pull-up resistor must be

FIGURE 17-3: EXTERNAL CIRCUITRY



17.2.2.5 Ping-Pong Buffer Configuration

The usage of ping-pong buffers is configured using the PPB1:PPB0 bits. Refer to **Section 17.4.4 "Ping-Pong Buffering"** for a complete explanation of the ping-pong buffers.

17.2.2.6 USB Output Enable Monitor

The USB $\overline{OE}$ monitor provides indication as to whether the SIE is listening to the bus or actively driving the bus. This is enabled by default when using an external transceiver or when UCFG<6> = 1.

17.2.2.7 Eye Pattern Test Enable

An automatic eye pattern test can be generated by the module when the UCFG<7> bit is set. The eye pattern output will be observable based on module settings, meaning that the user is first responsible for configuring the SIE clock settings, pull-up resistor and Transceiver mode. In addition, the module has to be enabled.

Once UTEYE is set, the module emulates a switch from a receive to transmit state and will start transmitting a J-K-J-K bit sequence (K-J-K-J for full speed). The sequence will be repeated indefinitely while the Eye Pattern Test mode is enabled.

Note that this bit should never be set while the module is connected to an actual USB system. This test mode is intended for board verification to aid with USB certi- fication tests. It is intended

17.2.2.8 Internal Regulator

The PIC18FX455/X550 devices have a built-in 3.3V reg- ulator to provide power to the internal transceiver and provide a source for the

Note: The drive from V_{USB} is sufficient to only drive an external pull-up in internal addition to the

The regulator is enabled by default and can be disabled through the VREGEN Configuration bit. When enabled, the voltage is visible on pin V_{USB}. When the regulator is disabled, a 3.3V source must be provided through the V_{USB} pin for the

Note 1: Do not enable the internal regulator if an external regulator is connected to V_{USB}.

17.2.3 USB STATUS REGISTER (USTAT)

The USB Status register reports the transaction status within the SIE. When the SIE issues a USB transfer complete interrupt, USTAT should be read to determine the status of the transfer. USTAT contains the transfer endpoint number,

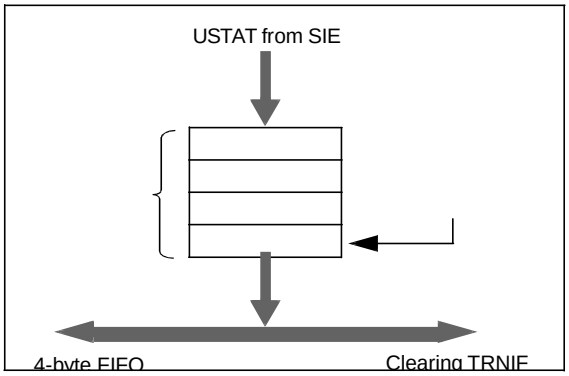
Note: The data in the USB Status register is only valid when the TRNIF interrupt

The USTAT register is actually a read window into a four-byte status FIFO, maintained by the SIE. It allows the microcontroller to process one transfer while the SIE processes additional endpoints (Figure 17-4). When the SIE completes using a buffer for reading or writing data, it updates the USTAT register. If another

Clearing the transfer complete flag bit, TRNIF, causes the SIE to advance the FIFO. If the next data in the FIFO holding register is valid, the SIE will immediately reassert the interrupt. If no additional data is present, TRNIF will remain

Note: If an endpoint request is received while the USTAT FIFO is full, the SIE will automatically issue a NAK back to

FIGURE 17-4: USTAT FIFO



REGISTER 17-3: USTAT: USB STATUS REGISTER

U-0	R-x	R-x	R-x	R-x	R-x	R-x	U-0
—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI(1)	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7 **Unimplemented:** Read as '0'

bit 6-

ENDP3:ENDP0: Encoded Number of Last Endpoint Activity bits (represents the number of the BDT updated by the last USB transfer) 1111 = Endpoint 15

1110 = Endpoint 14

....

bit

0001 = Endpoint 1

0000 = Endpoint 0

bit **DIR:** Last BD Direction Indicator bit

1 = The last transaction was an IN token

bit 0 = The last transaction was an OUT or SETUP token

Note 1: This bit is only valid for endpoints with available Even and Odd BD

17.2.4 USB ENDPOINT CONTROL

Each of the 16 possible bidirectional endpoints has its own independent control register, UEPn (where 'n' rep- resents the endpoint number). Each register has an identical complement of control bits. The prototype is shown in Register 17-4.

The EPHSHK bit (UEPn<4>) controls handshaking for the endpoint; setting this bit enables USB handshaking. Typically, this bit is always set except when using isochronous endpoints.

The EPCONDIS bit (UEPn<3>) is used to enable transactions. For Endpoint 0, this bit should always be cleared since the USB specifications identify Endpoint 0 as the default control endpoint.

The EPOUTEN bit (UEPn<2>) is used to enable or dis- able USB OUT transactions from the host. Setting this bit enables OUT transactions. Similarly, the EPINEN bit (UEPn<1>) enables or disables USB IN transactions from the host.

The EPSTALL bit (UEPn<0>) is used to indicate a STALL condition for the endpoint. If a STALL is

REGISTER 17-4: UEPn: USB ENDPOINT n CONTROL REGISTER (UEP0 THROUGH UEP15)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL(1)
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7-5 **Unimplemented:** Read as '0'

EPHSHK: Endpoint Handshake Enable bit

bit 4 1 = Endpoint handshake enabled

0 = Endpoint handshake disabled (typically used for isochronous endpoints)

EPCONDIS: Bidirectional Endpoint Control

bit 3 If EPOUTEN = 1 and EPINEN = 1:

1 = Disable Endpoint n from control transfers; only IN and OUT transfers allowed

bit 3 0 = Enable Endpoint n for control (SETUP) transfers; IN and OUT transfers also allowed

EPOUTEN: Endpoint Output Enable bit

bit 2 1 = Endpoint n output enabled

0 = Endpoint n output disabled

Note 1: Valid only if Endpoint n is enabled; otherwise, the bit is

17.2.5 USB ADDRESS REGISTER (UADDR)

The USB Address register contains the unique USB address that the peripheral will decode when active. UADDR is reset to 00h when a USB Reset is received, indicated by URSTIF, or when a Reset is received from the microcontroller. The USB address must be written by the

17.2.6 USB FRAME NUMBER REGISTERS (UFRMH:UFRML)

The Frame Number registers contain the 11-bit frame number. The low-order byte is contained in UFRML, while the three high-order bits are contained in UFRMH. The register pair is updated with the current frame number whenever a SOF token is received. For

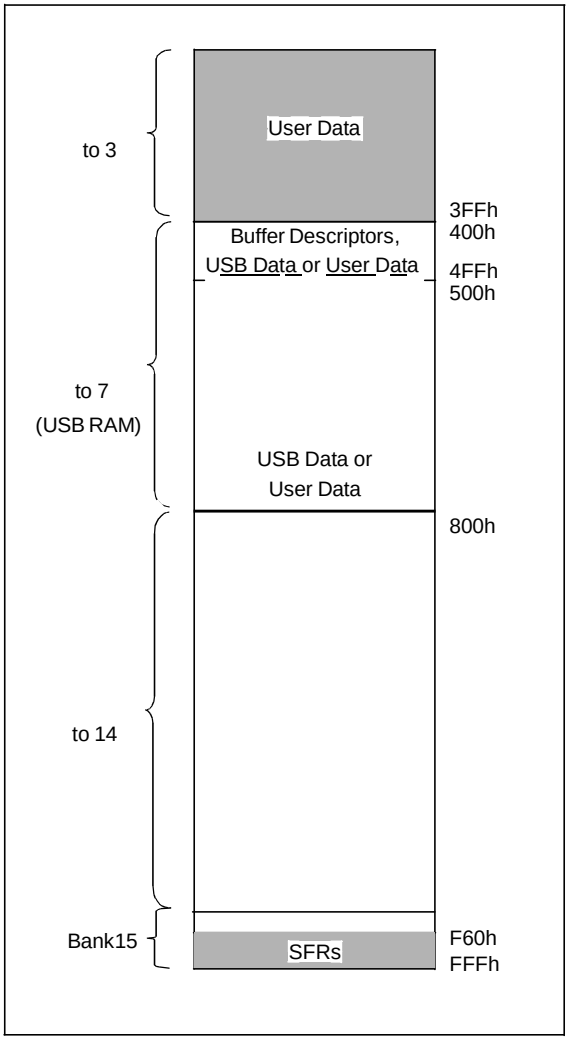
17.3 USB RAM

USB data moves between the microcontroller core and the SIE through a memory space known as the USB RAM. This is a special dual port memory that is mapped into the normal data memory space in Banks 4 through 7 (400h to 7FFh) for a total of 1 Kbyte (Figure 17-5).

Bank 4 (400h through 4FFh) is used specifically for endpoint buffer control, while Banks 5 through 7 are available for USB data. Depending on the type of buffering being used, all but 8 bytes of Bank 4 may also be available for use as USB buffer space.

Although USB RAM is available to the microcontroller as data memory, the sections

FIGURE 17-5: IMPLEMENTATION OF USB RAM IN DATA MEMORY SPACE



17.4 Buffer Descriptors and the Buffer Descriptor Table

The registers in Bank 4 are used specifically for end- point buffer control in a structure known as the Buffer Descriptor Table (BDT). This provides a flexible method for users to construct and control endpoint buffers of various lengths and configuration.

The BDT is composed of Buffer Descriptors (BD) which are used to define and control the actual

- BDnSTAT: BD Status register
- BDnCNT: BD Byte Count register
- BDnADRL: BD Address Low

BDs always occur as a four-byte block in the sequence, BDnSTAT:BDnCNT:BDnADRL:BDnADRH. The address of BDnSTAT is always an offset of (4n – 1) (in hexa- decimal) from 400h, with n being the buffer descriptor number.

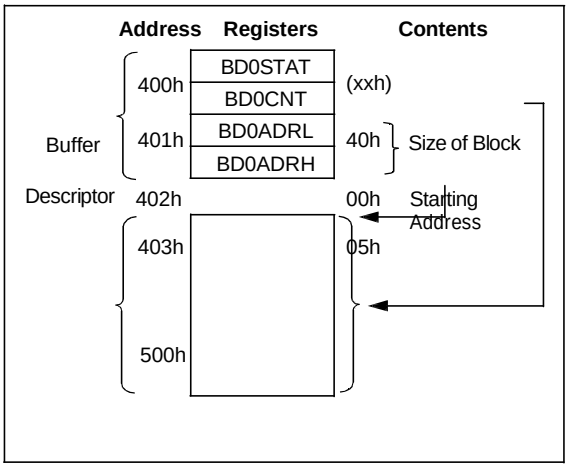
Depending on the buffering configuration used (**Section 17.4.4 "Ping-Pong Buffering"**), there are up to 32, 33 or 64 sets of buffer descriptors. At a minimum, the BDT must be at least 8 bytes long. This is because the USB specification mandates that every device must have Endpoint 0 with both input and output for initial setup. Depending on the endpoint and buffering configuration, the BDT can be as long as 256 bytes.

Although they can be thought of as Special Function Registers, the Buffer Descriptor Status and Address registers are not hardware mapped, as conventional microcontroller SFRs in Bank 15 are. If the endpoint cor- responding to a particular BD is not enabled, its registers are not used. Instead of appearing as unimplemented addresses, however, they appear as available RAM. Only when an endpoint is enabled by

17.4.1 BD STATUS AND CONFIGURATION

Buffer descriptors not only define the size of an end- point buffer, but also determine its configuration and control. Most of the configuration is done with the BD Status

FIGURE 17-6: EXAMPLE OF A BUFFER DESCRIPTOR



Unlike other control registers, the bit configuration for the BDnSTAT register is context sensitive. There are two distinct configurations, depending on whether the microcontroller or the USB module is modifying

17.4.1.1 Buffer Ownership

Because the buffers and their BDs are shared between the CPU and the USB module, a simple semaphore mechanism is used to distinguish which is allowed to update the BD and associated buffers in memory.

This is done by using the UOWN bit (BDnSTAT<7>) as a semaphore to distinguish which is allowed to update the BD and associated buffers in memory. UOWN is the only bit that is shared between the two configurations of BDnSTAT.

When UOWN is clear, the BD entry is "owned" by the microcontroller core. When the UOWN bit is set, the BD entry and the buffer memory are "owned" by the USB peripheral. The core should not modify the BD or its corresponding data buffer during this time. Note that the microcontroller core can still read BDnSTAT while the SIE owns the buffer and vice versa.

The buffer descriptors have a different meaning based on the source of the register update. Prior to placing ownership with the USB

The BDnSTAT byte of the BDT should always be the last byte updated when preparing to arm an endpoint. The SIE will clear the UOWN bit when a transaction has completed. The only exception to this is when KEN is enabled and/or BSTALL is enabled.

No hardware mechanism exists to block access when the UOWN bit is set. Thus, unexpected

17.4.1.2 BDnSTAT Register (CPU Mode)

When UOWN = 0, the microcontroller core owns the BD. At this point, the other seven bits of the register take on control functions.

The Keep Enable bit, KEN (BDnSTAT<5>), determines if a BD stays enabled. If the bit is set, once the UOWN bit is set, it will remain owned by the SIE independent of the endpoint activity. This prevents the USTAT FIFO from being updated, as well as the transaction complete interrupt from being set for the endpoint. This feature should only be enabled when the Streaming Parallel Port is selected as the data I/O channel instead of USB RAM.

The Address Increment Disable bit, INCDIS (BDnSTAT<4>), controls the SIE's automatic address increment function. Setting INCDIS disables the auto-increment of the buffer address by the SIE for each byte transmitted the SIE. When enabled, it checks the data packet's par- ity against the value of DTS (BDnSTAT<6>). If a packet arrives with an incorrect synchronization, the data will essentially be ignored. It will not be written to the USB RAM and the USB transfer complete interrupt flag will not be set. The SIE will send an ACK token back to the host to Acknowledge receipt, however. The effects of the DTSEN bit on the SIE are summarized in Table 17-3.

The Buffer Stall bit, BSTALL (BDnSTAT<2>), provides support for control transfers, usually one-time stalls on Endpoint 0. It also provides support for the SET_FEATURE/CLEAR_FEATURE commands speci- fied in Chapter 9 of the USB specification; typically, continuous STALLs to any endpoint other than the default control endpoint.

The BSTALL bit enables buffer stalls. Setting BSTALL causes the SIE to return a STALL token to the host if a received token would use the BD in that location. The EPSTALL bit in the

Section 17.4.2 "BD Byte Count" for mor information.

TABLE 17-3: EFFECT OF DTSEN BIT ON ODD/EVEN (DATA0/DATA1) PACKET RECEPTION

OUT Packet from Host	BDnSTAT Settings		Device Response after Receiving Packet			
	DTSEN	DTS	Handshake	UOWN	TRNIF	BDnSTAT and USTAT Status
DATA0	1	0	ACK	0	1	Update
DATA1	1	0	ACK	1	0	Not Updated
DATA0	1	1	ACK	0	1	Update
DATA1	1	1	ACK	1	0	Not Updated
Either	0	x	ACK	0	1	Update
Either, with error	x	x	NAK	1	0	Not Updated

Legend: x = don't care

REGISTER 17-5: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), CPU MODE (DATA IS WRITTEN TO THE SIDE)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN(1)	DTS(2)	KEN	INCDIS	DTSEN	BSTALL	BC9	BC8
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit **UOWN:** USB Own bit(1)

bit 0 = The microcontroller core owns the BD and its corresponding buffer

DTS: Data Toggle Synchronization bit(2)

bit 1 = Data 1 packet

0 = Data 0 packet

bit **KEN:** BD Keep Enable bit

bit 1 = USB will keep the BD indefinitely once UOWN is set (required for SPP endpoint configuration)

0 = USB will hand back the BD once a token has been processed

INCDIS: Address Increment Disable bit

bit 1 = Address increment disabled (required for SPP endpoint configuration)

0 = Address increment enabled

DTSEN: Data Toggle Synchronization Enable bit

bit 1- 1 = Data toggle synchronization is enabled; data packets with incorrect Sync value will be ignored except for a SETUP transaction, which is accepted even if the data toggle bits do not match

Note 1: This bit must be initialized by the user to the desired value prior to enabling the USB module.

17.4.1.3 BDnSTAT Register (SIE Mode)

When the BD and its buffer are owned by the SIE, most of the bits in BDnSTAT take on a different meaning. The configuration is shown in Register 17-6. Once UOWN is set, any data or control settings previously written there by the user will be overwritten with data from the SIE.

The BDnSTAT register is updated by the SIE with the token Packet Identifier (PID) which is stored in BDnSTAT<5:3>. The transfer count in

17.4.2 BD BYTE COUNT

The byte count represents the total number of bytes that will be transmitted during an IN transfer. After an IN transfer, the SIE will return the number of bytes sent to the host.

For an OUT transfer, the byte count represents the maximum number of bytes that can be received and stored in USB RAM. After an OUT transfer, the SIE will return the actual number of bytes received. If the number of

The 10-bit byte count is distributed over two registers. The lower 8 bits of the count reside in the BDnCNT register. The upper two bits

17.4.3 BD ADDRESS VALIDATION

The BD Address register pair contains the starting RAM address location for the corresponding endpoint buffer. For an endpoint starting location to be valid, it must fall in the range of the USB RAM, 400h to 7FFh. No mechanism is available in hardware to validate the BD address.

If the value of the BD address does not point to an address in the USB RAM, or if it points to unexpected results. developin USB applications, the user may want to consider the inclusion of software-based address

REGISTER 17-6: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), SIE MODE (DATA RETURNED BY THE SIDE TO THE MICROCONTROLLER)

R/W-x	U-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN	—	PID3	PID2	PID1	PID0	BC9	BC8
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit **UOWN:** USB Own bit

bit 6 1 = The SIE owns the BD and its corresponding buffer

bit 5- **Reserved:** Not written by the SIE

bit 1- **PID3:PID0:** Packet Identifier bits

The received token PID value of the last transfer (IN, OUT or SETUP transactions only).

17.4.4 PING-PONG BUFFERING

An endpoint is defined to have a ping-pong buffer when it has two sets of BD entries: one set for an Even transfer and one set for an Odd transfer. This allows the CPU to process one BD while the SIE is processing the other BD. Double-buffering BDs in this way allows for maximum

- No ping-pong support
- Ping-pong buffer support for OUT Endpoint
- 0 only Ping-pong buffer support for all endpoints

The ping-pong buffer settings are configured using the PPB1:PPB0 bits in the UCFG register.

The USB module keeps track of the Ping-Pong Pointer individually for each endpoint. All the completion of a transaction (UOWN cleared by the SIE), the pointer is toggled to the Odd BD. After the completion of the next transaction, the pointer is toggled back to the Even BD and so on.

The Even/Odd status of the last transaction is stored in the PPBI bit of the USTAT register. The user can reset all Ping-Pong Pointers to Even using the PPBRST bit.

Figure 17-7 shows the four different modes of operation and how USB RAM is filled with the BDs.

BDs have a fixed relationship to a particular endpoint depending on the buffering

FIGURE 17-7: BUFFER DESCRIPTOR TABLE MAPPING FOR BUFFERING MODES

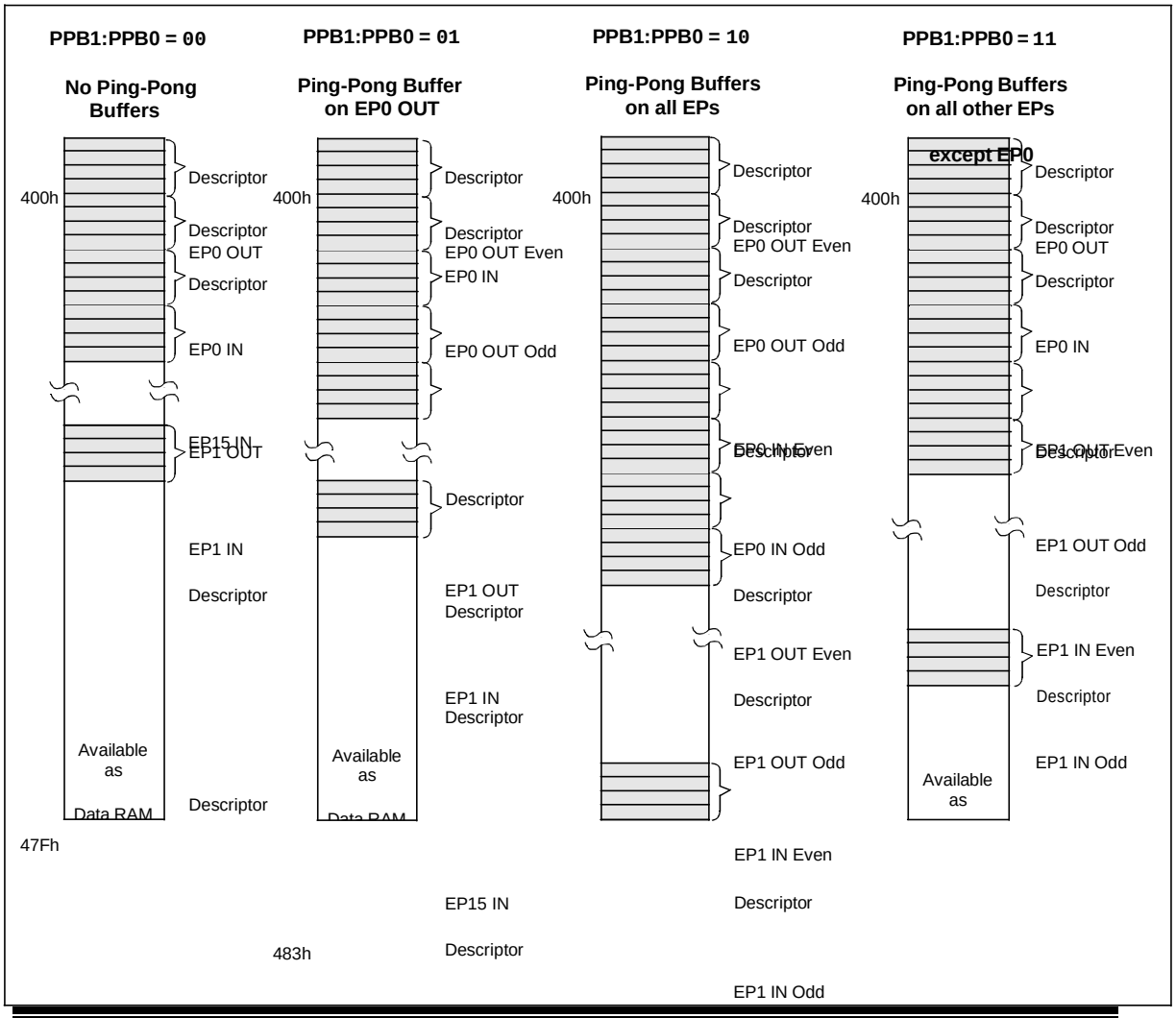


TABLE 17-4: ASSIGNMENT OF BUFFER DESCRIPTORS FOR THE DIFFERENT BUFFERING MODES

	BDs Assigned to Endpoint							
	Mode 0		Mode 1		Mode 2		Mode 3 (Ping-Pong on all other EPs,	
	Out	In	Out	In	Out	In	Out	In
0	0	1	0 (E), 1 (O)	2	0 (E), 1 (O)	2 (E), 3 (O)	0	1
1	2	3	3	4	4 (E), 5 (O)	6 (E), 7 (O)	2 (E), 3 (O)	4 (E), 5 (O)
2	4	5	5	6	8 (E), 9 (O)	10 (E), 11 (O)	6 (E), 7 (O)	8 (E), 9 (O)
3	6	7	7	8	12 (E), 13 (O)	14 (E), 15 (O)	10 (E), 11 (O)	12 (E), 13 (O)
4	8	9	9	10	16 (E), 17 (O)	18 (E), 19 (O)	14 (E), 15 (O)	16 (E), 17 (O)
5	10	11	11	12	20 (E), 21 (O)	22 (E), 23 (O)	18 (E), 19 (O)	20 (E), 21 (O)
6	12	13	13	14	24 (E), 25 (O)	26 (E), 27 (O)	22 (E), 23 (O)	24 (E), 25 (O)
7	14	15	15	16	28 (E), 29 (O)	30 (E), 31 (O)	26 (E), 27 (O)	28 (E), 29 (O)
8	16	17	17	18	32 (E), 33 (O)	34 (E), 35 (O)	30 (E), 31 (O)	32 (E), 33 (O)
9	18	19	19	20	36 (E), 37 (O)	38 (E), 39 (O)	34 (E), 35 (O)	36 (E), 37 (O)
10	20	21	21	22	40 (E), 41 (O)	42 (E), 43 (O)	38 (E), 39 (O)	40 (E), 41 (O)
11	22	23	23	24	44 (E), 45 (O)	46 (E), 47 (O)	42 (E), 43 (O)	44 (E), 45 (O)
12	24	25	25	26	48 (E), 49 (O)	50 (E), 51 (O)	46 (E), 47 (O)	48 (E), 49 (O)
13	26	27	27	28	52 (E), 53 (O)	54 (E), 55 (O)	50 (E), 51 (O)	52 (E), 53 (O)
14	28	29	29	30	56 (E), 57 (O)	58 (E), 59 (O)	54 (E), 55 (O)	56 (E), 57 (O)
15	30	31	31	32	60 (E), 61 (O)	62 (E), 63 (O)	58 (E), 59 (O)	60 (E), 61 (O)

Legend: (E) = Even transaction buffer, (O) = Odd transaction buffer

TABLE 17-5: SUMMARY OF USB BUFFER DESCRIPTOR TABLE REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BDnSTAT(1)	UOWN	DTS(4)	PID3(2) KEN(3)	PID2(2) INCDIS(3)	PID1(2) DTSEN(3)	PID0(2) BSTALL(3)	BC9	BC8
BDnCNT(1)	Byte Count							
BDnADRL(1)	Buffer Address Low							
BDnADRH(1)	Buffer Address High							

Note 1: For buffer descriptor registers, n may have a value of 0 to 63. For the sake of brevity, all 64 registers are shown as one generic prototype. All registers have indeterminate Reset values (xxxx xxxx).

2: Bits 5 through 2 of the BDnSTAT register are used by the SIE to return PID3:PID0 values once the register is turned over to the SIE (UOWN bit is set). Once the registers have been under SIE control, the values written for KEN, INCDIS, DTSEN and BSTALL are no longer valid.

3:

4:

17.5 USB Interrupts

The USB module can generate multiple interrupt con- ditions. To accommodate all of these interrupt sources, the module is provided with its own interrupt logic structure, similar to that of the microcontroller. USB interrupts are enabled with one set of control registers and request, (PIR2<5>), in the USBIF

Figure 17-8 shows the interrupt logic for the USB module. There are two layers of interrupt registers in the USB module. The top level consists of overall USB status interrupts; these are enabled and flagged in the UIE and UIR registers, respectively. The second level consists of USB error conditions, which are enabled and flagged in the UEIR and UEIE registers. An interrupt condition in any of these triggers a USB Error Interrupt Flag (UERRIF) in the top level

FIGURE 17-8: USB INTERRUPT LOGIC FUNNEL

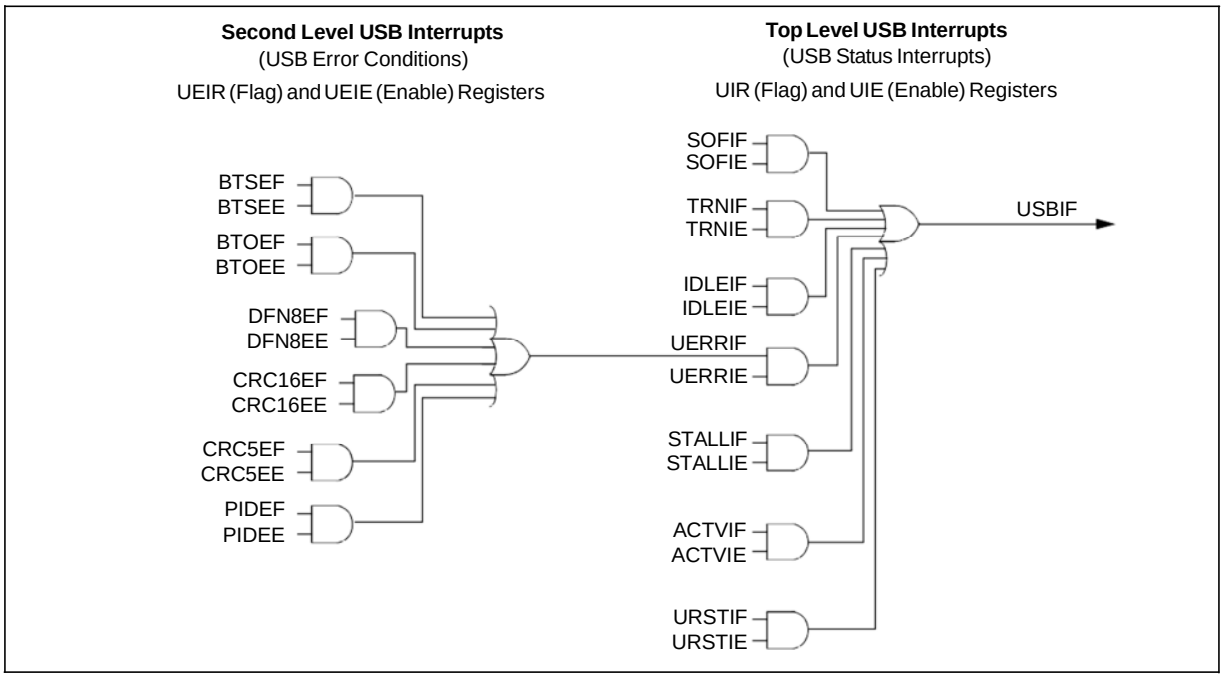
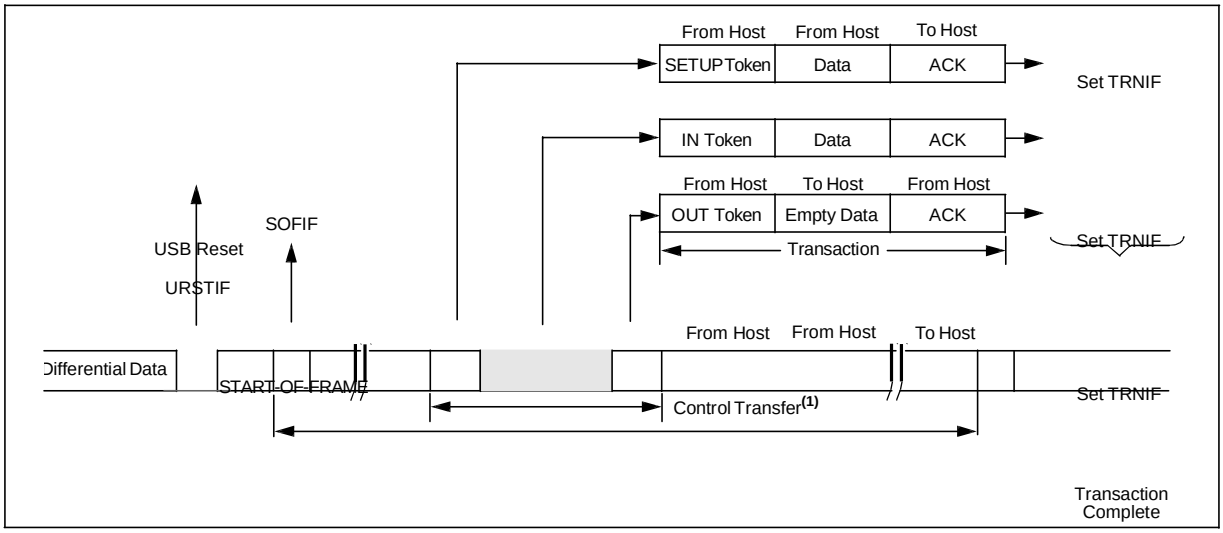


FIGURE 17-9: EXAMPLE OF A USB TRANSACTION AND INTERRUPT EVENTS



17.5.1 USB INTERRUPT STATUS REGISTER (UIR)

The USB Interrupt Status register (Register 17-7) con- tains the flag bits for each of the USB status interrupt sources. Each of these sources has a corresponding interrupt enable bit in the UIE register. All of the USB status flags are

Once an interrupt bit has been set by the SIE, it must be cleared by software by writing a '0'. The flag bits can also be set in software which

REGISTER 17-7: UIR: USB INTERRUPT STATUS REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0
—	SOFIF	STALLIF	IDLEIF(1)	TRNIF(2)	ACTVIF(3)	UERRIF(4)	URSTIF
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7 **Unimplemented:** Read as '0'

SOFIF: START-OF-FRAME Token Interrupt bit

bit 1 = A START-OF-FRAME token received by the SIE

0 = No START-OF-FRAME token received by the SIE

STALLIF: A STALL Handshake Interrupt bit

bit 1 = A STALL handshake was sent by the SIE 0 = A STALL handshake has not been sent

bit **IDLEIF:** Idle Detect Interrupt bit(1)

1 = Idle condition detected (constant Idle state of 3 ms or more)

bit 0 = No Idle condition detected

TRNIF: Transaction Complete Interrupt bit(2)

bit 1 = Processing of pending transaction is complete; read USTAT register for endpoint information

0 = Processing of pending transaction is not complete or no transaction is pending

bit **ACTVIF:** Bus Activity Detect Interrupt bit(3)

Note 1: Once an Idle state is detected, the user may want to place the USB module in Suspend mode.

2: Clearing this bit will cause the USTAT FIFO to advance (valid only for IN, OUT and SETUP tokens).

3: This bit is typically unmasked only following the detection of a UIDLE interrupt event.

17.5.1.1 Bus Activity Detect Interrupt Bit (ACTVIF)

The ACTVIF bit cannot be cleared immediately after the USB module wakes up from Suspend or while the USB module is suspended. A few clock cycles are required to synchronize the internal hardware state machine before the hardware is synchronized may not have an effect on the value of ACTVIF. Additionally, if the USB module uses the clock from the 96 MHz PLL source, then after clearing the SUSPND bit, the USB module may not be immediately operational while waiting for the 96 MHz PLL

EXAMPLE 17-1: CLEARING ACTVIF BIT (UIR<2>)

Assembly:

```
    BCF     UCON, SUSPND
LOOP:

    BTFSS   UIR, ACTVIF
    BRA     DONE

    BCF     UIR, ACTVIF

    BRA     LOOP
DONE:
```

17.5.2 USB INTERRUPT ENABLE REGISTER (UIE)

The USB Interrupt Enable register (Register 17-8) contains the enable bits for the USB status interrupt sources. Setting any of these bits will

The values in this register only affect the propagation of an interrupt condition to the microcontroller's inter- rupt logic. The flag bits are still set by their interrupt conditions,

REGISTER 17-8: UIE: USB INTERRUPT ENABLE REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit 7 **Unimplemented:** Read as '0'

bit **SOFIE:** START-OF-FRAME Token Interrupt Enable bit

1 = START-OF-FRAME token interrupt enabled

0 = START-OF-FRAME token interrupt disabled

bit **STALLIE:** STALL Handshake Interrupt Enable bit

1 = STALL interrupt enabled

bit 0 = STALL interrupt disabled

bit **IDLEIE:** Idle Detect Interrupt Enable bit

1 = Idle detect interrupt enabled

bit 0 = Idle detect interrupt disabled

bit **TRNIE:** Transaction Complete Interrupt Enable bit

1 = Transaction interrupt enabled

0 = Transaction interrupt disabled

17.5.3 USB ERROR INTERRUPT STATUS REGISTER (UEIR)

The USB Error Interrupt Status register (Register 17-9) contains the flag bits for each of the error sources within the USB peripheral. Each of these sources is controlled by a corresponding interrupt enable bit in the UEIE register. All of the USB error flags are ORed

Each error bit is set as soon as the error condition is detected. Thus, the interrupt will typically not correspond with the end of a token being processed.

REGISTER 17-9: UEIR: USB ERROR INTERRUPT STATUS REGISTER

R/C-0	U-0	U-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF
bit 7							bit 0

Legend:		
R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'

bit **BTSEF:** Bit Stuff Error Flag bit

1 = A bit stuff error has been detected

bit 6-5 0 = No bit stuff error

Unimplemented: Read as '0'

BTOEF: Bus Turnaround Time-out Error Flag bit

bit 1 = Bus turnaround time-out has occurred (more than 16 bit times of Idle from previous EOP elapsed)

bit 0 = No bus turnaround time-out

DFN8EF: Data Field Size Error Flag bit

bit 1 = The data field was not an integral number of bytes

0 = The data field was an integral number of bytes

bit **CRC16EF:** CRC16 Failure Flag bit

1 = The CRC16 failed

17.5.4 USB ERROR INTERRUPT ENABLE REGISTER (UEIE)

The USB Erro Interru Enable registe (Register 17-10) contains the enable bits for each of the USB error interrupt sources. Setting any of these bits will enable the respective error interrupt source in the UEIR

As with the UIE register, the enable bits only affect the propagation of an interrupt condition to the micro- controller's interrupt logic. The flag bits are still set by their interrupt

REGISTER 17-10: UEIE: USB ERROR INTERRUPT ENABLE REGISTER

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'

bit **BTSEE:** Bit Stuff Error Interrupt Enable bit

1 = Bit stuff error interrupt enabled

bit 6-5 0 = Bit stuff error interrupt disabled

Unimplemented: Read as '0'

BTOEE: Bus Turnaround Time-out Error Interrupt

bit Enable bit

1 = Bus turnaround time-out error interrupt enabled

bit 0 = Bus turnaround time-out error interrupt disabled **DFN8EE:** Data Field Size Error

bit Interrupt Enable bit 1 = Data field size error interrupt enabled

bit 0 = Data field size error interrupt disabled

CRC16EE: CRC16 Failure Interrupt Enable bit

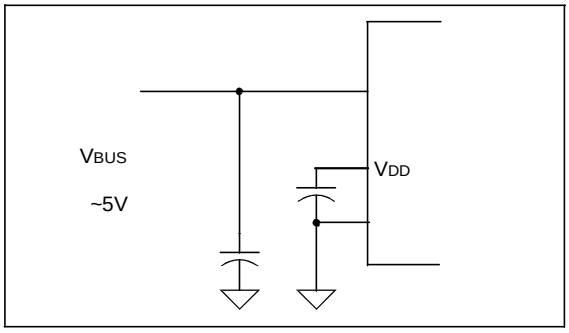
17.6 USB Power Modes

Many USB applications will likely have several different sets of power requirements and configuration. The most common power modes encountered are Bus Power Only, Self-Power Only and Dual Power with Self-Power

17.6.1 BUS POWER ONLY

In Bus Power Only mode, all power for the application is drawn from the USB (Figure 17-

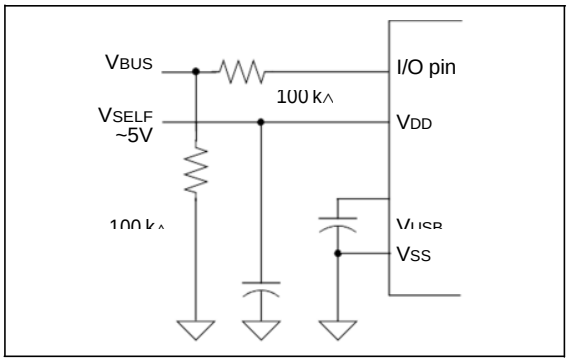
FIGURE 17-10: BUS POWER ONLY



17.6.2 SELF-POWER ONLY

In Self-Power Only mode, the USB application provides its own power, with very little power being pulled from the USB. Figure 17-11 shows an example. Note that an attach indication is

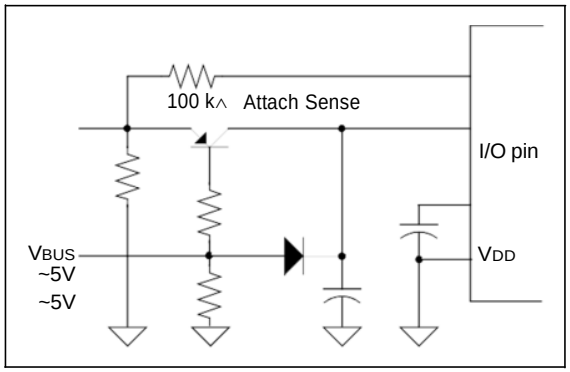
FIGURE 17-11: SELF-POWER ONLY



17.6.3 DUAL POWER WITH SELF-POWER DOMINANCE

Some applications may require a dual power option. This allows the application to use internal power prima- rily, but switch to power from the USB when no internal power is available. Figure 17-12 shows a simple Dual Power with Self-Power Dominance example.

FIGURE 17-12: DUAL POWER EXAMPLE



Note: Users should keep in mind the limits for devices drawing power from the USB. According to USB Specification 2.0,

17.7 Streaming Parallel Port

The Streaming Parallel Port (SPP) is an alternate route option for data besides USB RAM. Using the SPP, an endpoint can be configured to send data to or receive data directly from external hardware.

This methodology presents design possibilities where the microcontroller acts as a data manager, allowing the SPP to pass large blocks of data without the micro- controller actually processing it. An application example might include a data acquisition system, where data is streamed from an external FIFO through USB to the host computer. In this case, endpoint

1. Set BDnADRL:BDnADRH to point to FFFFh.
2. Set the KEN bit (BDnSTAT<5>) to let SIE
3. keep control of the buffer.

Set the INCDIS bit (BDnSTAT<4>) to

Refer to **Section 18.0 "Streaming Parallel Port"** for more information about the SPP.

Note 1: If an endpoint is configured to use the SPP, the SPP module must also be configured to use the USB module. Otherwise, unexpected operation may occur.

2: In addition, if an endpoint is

17.8 Oscillator

The USB module has specific clock requirements. For full-speed operation, the clock source must be 48 MHz. Even so, the microcontroller core and other peripherals are not required to run at that clock speed or even from the same clock source. Available clocking

17.9 USB Firmware and Drivers

Microchip provides a number of application specific resources, such as USB firmware and driver support. Refer to www.microchip.com for

TABLE 17-6: REGISTERS ASSOCIATED WITH USB MODULE OPERATION(1)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Details on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	51
IPR2	OSCFIP	CMIP	USBIP	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	54
PIR2	OSCFIF	CMIF	USBIF	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	54
PIE2	OSCFIE	CMIE	USBIE	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	54
UCON	—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—	55
UCFG	UTEYE	UOEMON	—	UPUEN	UTRDIS	FSEN	PPB1	PPB0	55
USTAT	—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI	—	55
UADDR	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	55
UFRML	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0	55
UFRMH	—	—	—	—	—	FRM10	FRM9	FRM8	55
UIR	—	SOFIF	STALLIF	IDLEIF	TRNIF	ACTVIF	UERRIF	URSTIF	55
UIE	—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE	55
UEIR	BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF	55
UEIE	BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE	55
UEP0	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP1	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP2	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP3	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP4	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP5	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP6	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP7	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP8	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP9	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP10	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP11	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP12	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP13	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP14	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55
UEP15	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	55

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the USB module.

Note 1: This table includes only those hardware mapped SFRs located in Bank 15 of the data memory space. The Buffer

17.10 Overview of USB

This section presents some of the basic USB concepts and useful information necessary to design a USB device. Although much information is provided in this section, there is a plethora of information provided within the USB specifications and class specifications. Thus, the reader is encouraged to refer to the USB

17.10.1 LAYERED FRAMEWORK

USB device functionality is structured into a layered framework graphically shown in Figure 17-13. Each level is associated with a functional level within the device. The highest layer, other than the device, is the configuration. A device may have multiple configura- tions. For example, a particular device may have multiple power requirements based on Self-Power Only or Bus Power Only modes.

For each configuration, there may be multiple interfaces. Each interface could support a particular mode of that configuration.

17.10.2 FRAMES

Information communicated on the bus is grouped into 1 ms time slots, referred to as endpoints. Figure 17-9 shows an of a example transaction within a frame

17.10.3 TRANSFERS

There are four transfer types defined in the USB specification

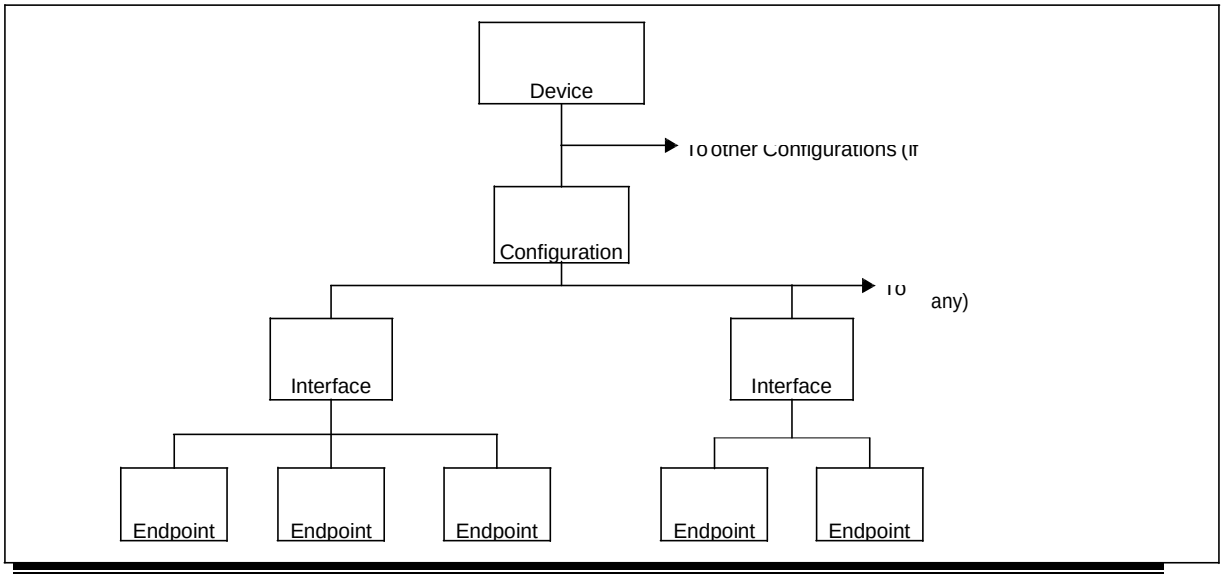
- **Isochronous:** This type provides a transfer method for large amounts of data (up to 1023 bytes) with timely delivery ensured;
- however, the data integrity is not ensured. This is good for streaming applications where small data loss is not critical, such as audio.
- **Bulk:** This type of transfer method allows for large amounts of data to be transferred with ensured data integrity; however, the

While full-speed devices support all transfer types, low-speed devices are limited to

17.10.4 POWER

Power is available from the Universal Serial Bus. The USB specification defines the bus power requirements. Devices may either be self-powered or bus powered. Self-powered devices draw power from an external source

FIGURE 17-13: USB LAYERS



The USB specification limits the power taken from the bus. Each device is ensured 100 mA at approximately 5V (one unit load). Additional power may be requested, up to a maximum of 500 mA. Note that power above one unit load is a request and the host or hub is not obligated to provide the extra current. Thus, a device capable of consuming more than one unit load must be able to maintain a low-power configuration of a one unit load or less, if necessary.

The USB specification also defines a Suspend mode. In this situation, current must be limited to 500 µA, averaged over 1 second. A

17.10.5 ENUMERATION

When the device is initially attached to the bus, the host enters an enumeration process in an attempt to identify the device. Essentially, the host interrogates the device, gathering information such as power consumption, data rates and sizes, protocol and other descriptive

1. USB Reset: Reset the device. Thus, the device is not configured and does not
2. Get Device Descriptor: The host a
3. requests small portion of the device descriptor
4. Set Address: The host assigns an address to
5. the device.
6. Get Device Descriptor: The host retrieves the device descriptor, gathering info such as manufacturer, type of device, maximum control packet size.
7. Get configuration

The exact enumeration process depends on the

17.10.6 DESCRIPTORS

There are eight different standard descriptor types of which five are most important for this

17.10.6.1 Device Descriptor

The device descriptor provides general information, such as manufacturer, product number, serial number, the class of the device

17.10.6.2 Configuration Descriptor

The configuration descriptor provides information on the power requirements of the device and how many different interfaces are supported when in this configu- ration. There

17.10.6.3 Interface Descriptor

The interface descriptor details the number of end- points used in this interface, as well as the class of the interface. There may be more than

17.10.6.4 Endpoint Descriptor

The endpoint descriptor identifies the transfer type (**Section 17.10.3 "Transfers"**) and direction, as well as some other specifics for the endpoint. There may be many endpoints in a

17.10.6.5 String Descriptor

Many of the previous descriptors reference one or more string descriptors. String (**Section 17.10.1 "Layered Framework"**) the describe. Often these strings show up in the host to help the user identify the device. String descriptors are generally optional to save

17.10.7 BUS SPEED

Each USB device must indicate its bus presence and speed to the host. This is accomplished through a

1.5 kΛ resistor which is connected to the bus at the

time of the attachment event.

Depending on the speed of the device, the

17.10.8 CLASS SPECIFICATIONS AND DRIVERS

USB specifications include class specifications which operating system vendors optionally support. Examples of classes include Audio, Mass Storage, Communications and Human Interface (HID). In most cases, a driver is required at the host side to 'talk' to the USB device. In custom applications, a driver may need

Listing programme hôte sur PC :acquisition par protocole USB-HID

```
 const

    BufferInSize  = 64;
    BufferOutSize = 64;

 type
    TBufferIn = array[0..BufferInSize] of byte;
TBufferOut = array[0..BufferOutSize] of byte;



    TForm1 = class(TForm)
     Panel1: TPanel;
     PaintBox1: TPaintBox;
     StatusBar1: TStatusBar;
     Timer1: TTimer;
     ToolbarImages: TImageList;
     MainMenu1: TMainMenu;
     Fichier1: TMenuItem;
     Configuration1: TMenuItem;
     EntresAnalog1: TMenuItem;
     Ouvrir1: TMenuItem;
     Saveas1: TMenuItem;
     Voir1: TMenuItem;
     SaveDialog1: TSaveDialog;
     OpenDialog1: TOpenDialog;
     Parametrage1: TMenuItem;
     lectrace1: TMenuItem;
     Panel2: TPanel;
     GroupBox1: TGroupBox;
     CheckBox1: TCheckBox;
     CheckBox2: TCheckBox;
     CheckBox3: TCheckBox;
     CheckBox4: TCheckBox;
     Memo1: TMemo;
     Label2: TLabel;
```

```
    Button6: TButton;
    ScrollBar2: TScrollBar;
    Label3: TLabel;
    Label4: TLabel;
      procedure FormCreate(Sender: TObject);
      procedure FormDestroy(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure PaintBox1Paint(Sender: TObject);
    procedure ScrollBar2Change(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure EntresAnalog1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure ScrollBar1Change(Sender: TObject);
    procedure Saveas1Click(Sender: TObject);
    procedure Voir1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure N11Click(Sender: TObject);
    procedure N21Click(Sender: TObject);
  private
     FBufferIn:TBufferIn;
     FBufferOut:TBufferOut;
     function USBEvent(var Msg: TMessage): Boolean;
     procedure traceur_T;
     procedure traceur_0831;
     procedure lecteur;
     procedure effacerec(x,l:integer);
     procedure effacerecz(x,l:integer);
     procedure rrpp;
  public
  end;

var
  Form1: TForm1;
  v,w:array[1..n] of byte;
```

```
  act:integer;
  recu:boolean;
  k1,k2:integer;
  f:textfile;
  t0,t1:dword;
  comm:string;
  envo:array[1..64] of byte;

  vmax:integer;
  ymax,ymin,vlu:array[1..8] of integer;
  entbyte:shortstring;
  absxr:real;
  absx,absxmax:integer;
  dat:array[1..8,1..5000] of word;
  canaux,trac:array[1..9] of byte;
  preced:array[1..12] of integer;

  aaa:array[1..12] of real;
  passe:boolean;
  modes,cas:byte;
implementation
uses  Unit_entree,
      USB_file1,
      USB_file2;

const

   VENDOR_ID           = 5336;
   PRODUCT_ID          = 21;

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
   Application.HookMainWindow(USBEvent);
   Connect(Application.Handle);
   act:=1;
```

```
timer1.Enabled:=false;
canaux[1]:=3;canaux[2]:=7;canaux[3]:=11;canaux[4]:=15;
canaux[5]:=0;canaux[6]:=0;canaux[7]:=0;canaux[8]:=0;
canaux[9]:=4;

modes:=1;
act:=1;
vmax:=1024;
end;

procedure TForm1.effacerec(x,l:integer);
var r2:Trect;
begin
r2:=rect(x+1,0,x+l,paintbox1.Height);
end;

procedure TForm1.effacerecz(x,l:integer);
var r2:Trect;
begin
r2:=rect(x+1,0,x+l,paintbox1.Height);
paintbox1.canvas.brush.color := clblack;
paintbox1.canvas.Fillrect(r2);
end;


procedure TForm1.traceur_T;
const                              coul:array[1..8]                              of
Tcolor=(clyellow,ClRed,clgreen,clteal,clwhite,clpurple,clred,clsilver);
var
  i,yy,ps,pn: integer;
  cc:byte;
  dd:dword;
begin
    if act mod 100 =0 then label4.Caption:=inttostr(act);
    if passe then label3.Caption:='passe' else label3.caption:=' ';
    ps:=scrollbar2.Position;
```

```
    pn:=1;
    i:=3;yy:=$62;
    repeat
    if pn=1 then
    if     absx<absxmax-20     then     effacerec(absx,20)     else     begin
effacerec(absx,absxmax-absx);effacerec(0,20-(absxmax-absx));end;

    cc:=envo[i] shr 2;
    dd:=FBufferIn[i]*4 + round(FBufferIn[i+1]/64);
    dat[cc+1,act]:=dd;

    i:=i+2;yy:=yy+2;
    pn:=pn+1;if pn>canaux[9] then
        begin
        absx:=absx+ps;
        if absx>absxmax then absx:=0;
        pn:=1;inc(act);if act=5001 then begin act:=1;passe:=true;end;
        end;
    until yy=FBufferIn[2];
end;


procedure TForm1.traceur_0831;
const                              coul:array[1..8]                              of
Tcolor=(clyellow,ClRed,clgreen,clteal,clwhite,clpurple,clred,clsilver);
var DevHandle:cardinal;
  y,ps,pn: integer;
  dd:dword;
  ab0:integer;
begin
   if button6.Caption='Stop' then
   begin
   FBufferOut[1]:=21;FBufferOut[2]:=15
   DevHandle := GetHandle(VENDOR_ID , PRODUCT_ID);
   Write(DevHandle,@FBufferOut);
   end;
```

```
    if act mod 100 =0 then label4.Caption:=inttostr(act);
    if passe then label3.Caption:='passe' else label3.caption:=' ';
    ps:=scrollbar2.Position;

    pn:=1;
    if pn=1 then if absx<absxmax-20 then
         effacerecz(absx,20)      else      begin      effacerecz(absx,absxmax-
absx);effacerecz(0,20-(absxmax-absx));end;
    ab0:=trunc((absxr*10+ps)/10);
    repeat
    if trac[pn]=1 then
      begin
      paintbox1.canvas.Pen.color := coul[pn];
      dd:=FBufferIn[pn+6];
      dat[pn,act]:=dd;
      paintbox1.canvas.moveto(absx,preced[pn]);
      y:=trunc(dd*aaa[pn])+ymax[pn];
      preced[pn]:=y;

      paintbox1.canvas.lineto(ab0,y);
      end;
    pn:=pn+1;
    until pn>4;
    absxr:=(absxr*10+ps)/10;
    absx:=trunc(absxr);
    if absx>absxmax then begin absx:=0;absxr:=0;end;
    inc(act);if act=5001 then begin act:=1;passe:=true;end;

end;

procedure TForm1.lecteur;
var  DevHandle:cardinal;
  i,yy,pn: integer;
  cc:byte;
  dd:dword;
begin
   begin
```

```
   for i:=1 to 64 do FBufferOut[I] :=envo[i];
   DevHandle := GetHandle(VENDOR_ID , PRODUCT_ID);
   Write(DevHandle,@FBufferOut);
   end;

    if act mod 100 =0 then label4.Caption:=inttostr(act);
    if passe then label3.Caption:='passe' else label3.caption:=' ';
    pn:=1;
    i:=3;yy:=$62;
    repeat
    cc:=envo[i] shr 2;
    dd:=FBufferIn[i]*4 + round(FBufferIn[i+1]/64);
    dat[cc+1,act]:=dd;

    i:=i+2;yy:=yy+2;
    pn:=pn+1;if pn>canaux[9] then
        begin
        pn:=1;inc(act);if act=5001 then begin act:=1;passe:=true;end;
        end;
    until yy=FBufferIn[2];
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
   Application.UnHookMainWindow(USBEvent);
end;


function TForm1.USBEvent(var Msg: TMessage): Boolean;
var
   DevHandle:cardinal;
begin
  result := False;
  if Msg.Msg = WM_HID_EVENT then
  begin
     case Msg.WParam of
```

```
        NOTIFY_PLUGGED :
        begin
          DevHandle := Msg.LParam;
           if       (GetVendorID(DevHandle)       =       VENDOR_ID)       and
(GetProductID(DevHandle) = PRODUCT_ID) then
           begin
           end;
           result := true;
        end;

        NOTIFY_UNPLUGGED :
        begin

           DevHandle := Msg.LParam;
           if       (GetVendorID(DevHandle)       =       VENDOR_ID)       and
(GetProductID(DevHandle) = PRODUCT_ID) then
           begin
           end;
           result := true;
        end;

        NOTIFY_CHANGED :
        begin
           DevHandle := GetHandle(VENDOR_ID,PRODUCT_ID);
           SetReadNotify(DevHandle,true);
           result := true;
        end;
        NOTIFY_READ :
        begin
         DevHandle := Msg.LParam;
         if (GetVendorID(DevHandle) = VENDOR_ID) and (GetProductID(DevHandle)
= PRODUCT_ID) then
         begin
         Read(DevHandle,@FBufferIn);
         if comm='fichier' then
           begin
           end else
```

```
          if comm='adc_T' then
            begin
             traceur_T;
            end else
          if comm='adc_D' then
            begin
            end else
          if comm='lec' then
            begin
            lecteur;
            end else
          if comm='adc0831' then
            begin
            traceur_0831
            end;
          end;
         end;
      end;
  end;
end;


procedure TForm1.PaintBox1Paint(Sender: TObject);
begin
memo1.Lines.Add('rr');
 rrpp;

end;


procedure TForm1.ScrollBar2Change(Sender: TObject);
begin
statusbar1.Panels[3].text:='Echelle Horiz:'+inttostr(scrollbar2.position);
label2.Caption:='Echelle :'+inttostr(scrollbar2.Position);
end;

procedure TForm1.Button3Click(Sender: TObject);
```

```
var i,j:integer;
begin
if (button6.caption='Trace_T') and (canaux[9]<>0) then
   begin
   for i:=1 to 8 do
    begin
    for j:=1 to 5000 do dat[i,j]:=0;
    end;
   act:=1;
   passe:=false;
   absx:=0;
   absxmax:=paintbox1.Width;
   timer1.Enabled:=true;
   mainmenu1.Items[1].Enabled:=false;
   comm:='adc_T';
   end else
   begin
   timer1.Enabled:=false;
   mainmenu1.Items[1].Enabled:=true;
   end;

end;

procedure TForm1.EntresAnalog1Click(Sender: TObject);
var ee:byte;
begin
timer1.Enabled:=false;
form_entree:=TForm_entree.create(self);
with form_entree do
 begin
 for ee:=1 to canaux[9] do
   begin
   case canaux[ee] of
   3:checkbox9.Checked:=true;
   7:checkbox10.Checked:=true;
   11:checkbox11.Checked:=true;
   15:checkbox12.Checked:=true;
```

```
   19:checkbox13.Checked:=true;
   23:checkbox14.Checked:=true;
   27:checkbox15.Checked:=true;
   31:checkbox16.Checked:=true;
   end;
   end;
 end;
if form_entree.ShowModal =1 then
  begin
  canaux[9]:=0;ee:=1;
  if            form_entree.CheckBox9.Checked            then            begin
canaux[ee]:=3;inc(ee);inc(canaux[9]);end else canaux[ee]:=0;
  if            form_entree.CheckBox10.Checked            then            begin
canaux[ee]:=7;inc(ee);inc(canaux[9]);end else canaux[ee]:=0;
  if            form_entree.CheckBox11.Checked            then            begin
canaux[ee]:=11;inc(ee);inc(canaux[9]);end else canaux[ee]:=0;
  if            form_entree.CheckBox12.Checked            then            begin
canaux[ee]:=15;inc(ee);inc(canaux[9]);end else canaux[ee]:=0;
  if            form_entree.CheckBox13.Checked            then            begin
canaux[ee]:=19;inc(ee);inc(canaux[9]);end else canaux[ee]:=0;
  if            form_entree.CheckBox14.Checked            then            begin
canaux[ee]:=23;inc(ee);inc(canaux[9]);end else canaux[ee]:=0;
  if            form_entree.CheckBox15.Checked            then            begin
canaux[ee]:=27;inc(ee);inc(canaux[9]);end else canaux[ee]:=0;
  if            form_entree.CheckBox16.Checked            then            begin
canaux[ee]:=31;inc(canaux[9]);end else canaux[ee]:=0;

  if form_entree.CheckBox1.Checked then modes:=0 else modes:=1;
  end;
if canaux[9]=0 then button6.Enabled:=false else button6.Enabled:=true;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var DevHandle:cardinal;
    i:integer;
begin
k2:=0;t0:=gettickcount;
```

```
for i:=1 to 64 do FBufferOut[I] :=envo[i];
DevHandle := GetHandle(VENDOR_ID , PRODUCT_ID);
Write(DevHandle,@FBufferOut);
end;


procedure TForm1.Saveas1Click(Sender: TObject);
var
  f: TextFile;
  i,k:integer;
  nom:string;
  cana:integer;
begin
  if SaveDialog1.Execute then
  begin
  for i:=1 to 4 do
    begin
    nom:=savedialog1.FileName;
    insert('_'+inttostr(i),nom,length(nom)-3);
    AssignFile(f, nom);
    Rewrite(f);
    if passe then k:=act+1 else k:=1;
     repeat
     writeln(f,dat[i,k]);
     inc(k);if k=5001 then k:=1;
     until k=act;
    CloseFile(f);
    end;
  end
end;

procedure TForm1.Voir1Click(Sender: TObject);
var nom:string;
    p:array[0..150] of char;
begin
if opendialog1.Execute then
  begin
```

```
  nom:='Pr_trace3.exe'+'              '+extractfilename(opendialog1.filename)+'
'+inttostr(vmax);
  strpcopy(p,nom);
  winexec(p,1);
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var i,j:integer;
    DevHandle:cardinal;
begin
if (button6.caption='Lecture') and (canaux[9]<>0) then
   begin
   for i:=1 to 8 do
    begin
    for j:=1 to 5000 do dat[i,j]:=0;
    end;
   act:=1;
   passe:=false;
   absx:=0;
   absxmax:=paintbox1.Width;
   mainmenu1.Items[1].Enabled:=false;
   comm:='lec';

   for i:=1 to 64 do FBufferOut[I] :=envo[i];
   DevHandle := GetHandle(VENDOR_ID , PRODUCT_ID);
   Write(DevHandle,@FBufferOut);

   end else
   begin
   mainmenu1.Items[1].Enabled:=true;
   end;

end;

procedure TForm1.Button2Click(Sender: TObject);
var i,j:integer;
```

```
    DevHandle:cardinal;
begin
if (button6.caption='Trace_D') and (canaux[9]<>0) then
   begin
   for i:=1 to 8 do
    begin
    for j:=1 to 5000 do dat[i,j]:=0;
    end;
   act:=1;
   passe:=false;
   absx:=0;
   absxmax:=paintbox1.Width;
   mainmenu1.Items[1].Enabled:=false;
   comm:='adc_D';
   for i:=1 to 64 do FBufferOut[I] :=envo[i];
   DevHandle := GetHandle(VENDOR_ID , PRODUCT_ID);
   Write(DevHandle,@FBufferOut);
   end else
   begin
   mainmenu1.Items[1].Enabled:=true;
   end;

end;

procedure TForm1.Button5Click(Sender: TObject);
  var DevHandle:cardinal;
begin
   comm:='';
   FBufferOut [1]:= 2 ;
   DevHandle := GetHandle(VENDOR_ID , PRODUCT_ID);
   Write(DevHandle,@FBufferOut);

end;


procedure TForm1.Button6Click(Sender: TObject);
var i,j:integer;
```

```
    DevHandle:cardinal;
    yy0,yy1,dy:integer;
begin
if (button6.caption='Adc0831')  then
   begin
   paintbox1.Canvas.Pen.width :=2;
   checkbox1.Enabled:=false;
   checkbox2.Enabled:=false;
   checkbox3.Enabled:=false;
   checkbox4.Enabled:=false;
   paintbox1.Canvas.Brush.Color:=clblack;
   paintbox1.canvas.Fillrect(rect(0,0,paintbox1.width,paintbox1.height));

   for i:=1 to 8 do
    begin
    for j:=1 to 5000 do dat[i,j]:=0;
    trac[i]:=0;canaux[i]:=0;
    end;
   act:=1;
   passe:=false;
   absx:=0;
   absxr:=0;
   absxmax:=paintbox1.Width;
   button6.caption:='Stop';
   mainmenu1.Items[1].Enabled:=false;
   comm:='adc0831';
 canaux[9]:=0;
 if checkbox1.Checked then begin inc(canaux[9]);canaux[1]:=1;trac[1]:=1;end;
 if checkbox2.Checked then begin inc(canaux[9]);canaux[2]:=1;trac[2]:=1;end;
 if checkbox3.Checked then begin inc(canaux[9]);canaux[3]:=1;trac[3]:=1;end;
 if checkbox4.Checked then begin inc(canaux[9]);canaux[4]:=1;trac[4]:=1;end;
 yy0:=0+2;yy1:=paintbox1.Height-2;
 if canaux[9]<>0 then dy:=(yy1-yy0) div canaux[9] else dy:=(yy1-yy0);
  memo1.Clear;
 i:=yy0+dy-2;
```

```
 if   canaux[1]=1   then   begin   ymax[1]:=i;ymin[1]:=i-dy+2;aaa[1]:=(ymin[1]-
ymax[1])/255;i:=i+dy;memo1.lines.add(inttostr(ymin[1])+'/'+inttostr(ymax[1]));
end;
 if   canaux[2]=1   then   begin   ymax[2]:=i;ymin[2]:=i-dy+2;aaa[2]:=(ymin[2]-
ymax[2])/255;i:=i+dy;memo1.lines.add(inttostr(ymin[2])+'/'+inttostr(ymax[2]));
end;
 if   canaux[3]=1   then   begin   ymax[3]:=i;ymin[3]:=i-dy+2;aaa[3]:=(ymin[3]-
ymax[3])/255;i:=i+dy;memo1.lines.add(inttostr(ymin[3])+'/'+inttostr(ymax[3]));
end;
 if   canaux[4]=1   then   begin   ymax[4]:=i;ymin[4]:=i-dy+2;aaa[4]:=(ymin[4]-
ymax[4])/255;i:=i+dy;memo1.lines.add(inttostr(ymin[4])+'/'+inttostr(ymax[4]));
end;

   for i:=1 to 64 do FBufferOut[I] :=envo[i];
   FBufferOut[1]:=21;FBufferOut[2]:=15;     //code 5   entrées logiques: 15
   DevHandle := GetHandle(VENDOR_ID , PRODUCT_ID);
   Write(DevHandle,@FBufferOut);

   end else
   begin
   button6.caption:='Adc0831';
   mainmenu1.Items[1].Enabled:=true;
   checkbox1.Enabled:=true;
   checkbox2.Enabled:=true;
   checkbox3.Enabled:=true;
   checkbox4.Enabled:=true;
   end;

end;

end.
```

Listing programme hôte sur PC : Traitements des signaux

```
type
  TForm_trace = class(TForm)
    Panel1: TPanel;
    ScrollBar1: TScrollBar;
    ScrollBar2: TScrollBar;
    PaintBox1: TPaintBox;
    Label1: TLabel;
    OpenDialog1: TOpenDialog;
    MainMenu1: TMainMenu;
    ajouter: TMenuItem;
    Label5: TLabel;
    enlevertout1: TMenuItem;
    Label2: TLabel;
    Ajouter1: TMenuItem;
    annuler1: TMenuItem;
    effacertout1: TMenuItem;
    lLabel10: TLabel;
    lLabel11: TLabel;
    lLabel12: TLabel;
    lLabel13: TLabel;
    lLabel5: TLabel;
    Panel3: TPanel;
    Edit1: TEdit;
    PopupMenu1: TPopupMenu;
    Button1: TButton;
    Button2: TButton;
    Label4: TLabel;
    echelleTemps1: TMenuItem;
    origine1: TMenuItem;
    inverse1: TMenuItem;
```

```
echelleY1: TMenuItem;
vertical: TLabel;
Panel4: TPanel;
Label3: TLabel;
Button3: TButton;
Button4: TButton;
ScrollBar3: TScrollBar;
Panel5: TPanel;
Label9: TLabel;
Button5: TButton;
Button6: TButton;
Edit2: TEdit;
Label10: TLabel;
test1: TMenuItem;
Memo1: TMemo;
Volume1: TMenuItem;
labx0: TLabel;
labx1: TLabel;
laby0: TLabel;
laby1: TLabel;
labpx: TLabel;
PaintBox2: TPaintBox;
Label6: TLabel;
Deplacer1: TMenuItem;
Panel2: TPanel;
ScrollBar4: TScrollBar;
ScrollBar5: TScrollBar;
Button7: TButton;
freqcardiaque1: TMenuItem;
Label7: TLabel;
delpcer1: TMenuItem;
procedure ScrollBar1Change(Sender: TObject);
procedure ScrollBar2Change(Sender: TObject);
```

```
  procedure PaintBox1Paint(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;Shift: TShiftState; X, Y:
Integer);
  procedure PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,Y: Integer);
  procedure enlever1Click(Sender: TObject);
  procedure ajouterClick(Sender: TObject);
  procedure enlevertout1Click(Sender: TObject);
  procedure Ajouter1Click(Sender: TObject);
  procedure Ajouterr1Click(Sender: TObject);
  procedure ddd1Click(Sender: TObject);
  procedure code1Click(Sender: TObject);
  procedure effacertout1Click(Sender: TObject);
  procedure annuler1Click(Sender: TObject);
  procedure Memo1Click(Sender: TObject);
  procedure Echellehoriz1Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure origine1Click(Sender: TObject);
  procedure inverse1Click(Sender: TObject);
  procedure xdcfvg1Click(Sender: TObject);
  procedure echelleY1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure echelleTemps1Click(Sender: TObject);
  procedure ScrollBar3Change(Sender: TObject);
  procedure Button5Click(Sender: TObject);
  procedure test1Click(Sender: TObject);
  procedure Volume1Click(Sender: TObject);
  procedure Memo1DblClick(Sender: TObject);
  procedure PaintBox2Paint(Sender: TObject);
  procedure ScrollBar5Change(Sender: TObject);
  procedure ScrollBar4Change(Sender: TObject);
  procedure Button7Click(Sender: TObject);
  procedure freqcardiaque1Click(Sender: TObject);
```

```
    procedure delpcer1Click(Sender: TObject);
  private
      procedure trace;
      procedure traceseul(i:byte);
      procedure calculer;
  public
  end;

var
  Form_trace: TForm_trace;
  bm1,bm2:TBitmap;
  nmax:integer;
  xlarg:integer;
  nf:integer;
  tt:array[1..16] of array of integer;
  limi:array[1..16,1..3] of integer;
  echellex:real;
  echelley:array[1..16] of real;
implementation
uses unit1;
{$R *.dfm}

procedure TForm_trace.traceseul(i:byte);
const clr:array[1..8] of Tcolor=(clred,clyellow,clBlue,clCream,clFuchsia,clGreen,clMaroon,clPurple);
var
  j,x,y,p1,p2: integer;
  yhaut,ybas:integer;
begin
bm1.width  := form_trace.width;
bm1.height := form_trace.height-panel1.Height;
 begin
 yhaut:=scrollbar4.Position+scrollbar5.Position;
 ybas:=scrollbar5.Position;
```

```
 xlarg:=paintbox1.Width;

 p1:=scrollbar1.Position;
 p2:=scrollbar2.Position;
 bm1.canvas.Pen.color := clwhite;
 j:=((tt[i][0]-limi[i][1])*(ybas-yhaut)) div (limi[i][2]-limi[i][1])+yhaut;
 bm1.Canvas.MoveTo(0,j);bm1.Canvas.LineTo(xlarg,j);
 bm1.canvas.Pen.color := clr[i];
 bm1.canvas.moveto(0,j);

 j:=1;
 repeat
  y:=((tt[i][j]-limi[i][1])*(ybas-yhaut)) div (limi[i][2]-limi[i][1])+yhaut;
  x:=((j-p2)*p1) div 10;
  bm1.canvas.lineto(x,y);
  j:=j+1;
  until (x=xlarg) or(j>limi[i][3]);
 scrollbar1.Visible:=true;
 end ;
paintbox1.canvas.draw(0,0,bm1);
paintbox2.canvas.draw(0,0,bm2);
end;


procedure TForm_trace.trace;
const clr:array[1..8] of Tcolor=(clred,clyellow,clBlue,clCream,clFuchsia,clGreen,clMaroon,clPurple);
var
  i,j,x,y,p1,p2: integer;
  yhaut,ybas:integer;
begin
bm1.width  := form_trace.width;
bm2.width  := form_trace.width;
bm1.height := form_trace.height-panel1.Height;
```

```
bm2.height := 48;
bm1.canvas.brush.color := clyellow;
bm2.canvas.brush.color := clyellow;
bm1.canvas.fillrect(rect(0,0,bm1.width, bm1.height));
bm2.canvas.fillrect(rect(0,0,bm2.width, bm2.height));
bm1.canvas.brush.color := clwhite;
bm2.canvas.brush.color := clwhite;
bm1.canvas.fillrect(rect(0,0,bm1.width, bm1.height));
bm2.canvas.fillrect(rect(0,0,bm2.width, bm2.height));

    bm2.canvas.Pen.color := clblack;
    bm2.Canvas.Font.Color:= clblack;
    bm2.canvas.moveto(0,10);
  bm2.Canvas.LineTo(200,10);bm2.Canvas.LineTo(200,0);bm2.Canvas.LineTo(200,10);
  bm2.Canvas.LineTo(400,10);bm2.Canvas.LineTo(400,0);bm2.Canvas.LineTo(400,10);
  bm2.Canvas.LineTo(600,10);bm2.Canvas.LineTo(600,0);bm2.Canvas.LineTo(600,10);
  bm2.Canvas.LineTo(800,10);bm2.Canvas.LineTo(800,0);bm2.Canvas.LineTo(800,10);
  bm2.Canvas.LineTo(1000,10);bm2.Canvas.LineTo(1000,0);bm2.Canvas.LineTo(1000,10);
  bm2.Canvas.Pen.Color:=clred;

  bm2.Canvas.TextOut(200,10,'1');
  bm2.Canvas.TextOut(400,10,'2');
  bm2.Canvas.TextOut(600,10,'3');
  bm2.Canvas.TextOut(800,10,'4');
  bm2.Canvas.TextOut(800,10,'5');

for i:=1 to nf do
 begin
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;

 xlarg:=paintbox1.Width;
```

```
 p1:=scrollbar1.Position;
 p2:=scrollbar2.Position;
 bm1.canvas.Pen.color := clblack;
 j:=((tt[i][0]-limi[i][1])*(ybas-yhaut)) div (limi[i][2]-limi[i][1])+yhaut;
 bm1.Canvas.MoveTo(0,j);bm1.Canvas.LineTo(xlarg,j);
 bm1.canvas.Pen.color := clr[i];
 bm1.canvas.moveto(0,j);

 j:=1;
 repeat
  y:=((tt[i][j]-limi[i][1])*(ybas-yhaut)) div (limi[i][2]-limi[i][1])+yhaut;
  x:=((j-p2)*p1) div 10;
  bm1.canvas.lineto(x,y);
  j:=j+1;
  until (x=xlarg) or(j>limi[i][3]);
 scrollbar1.Visible:=true;
 end ;
paintbox1.canvas.draw(0,0,bm1);
paintbox2.canvas.draw(0,0,bm2);
end;



procedure TForm_trace.xdcfvg1Click(Sender: TObject);
begin
memo1.Lines.Add('ff');
end;


procedure TForm_trace.PaintBox2Paint(Sender: TObject);
begin
trace;
end;
```

```
procedure TForm_trace.ScrollBar2Change(Sender: TObject);
begin
trace;
end;


procedure TForm_trace.ScrollBar1Change(Sender: TObject);
begin
label1.Caption:=inttostr(scrollbar1.Position);
if nmax-(10*xlarg) div scrollbar1.Position>0 then
   begin
   scrollbar2.Max:=nmax-(10*xlarg)div scrollbar1.Position ;
   scrollbar2.Visible:=true;
   end else
 begin
 scrollbar2.Visible:=false;
 scrollbar2.Position:=0;
 end;

trace;
end;

procedure TForm_trace.Ajouter1Click(Sender: TObject);
var  f:textfile;
    i,nn,lu,code:integer;
    s,s1,s2,s3,s4:string;
begin
if (opendialog1.Execute) then
 begin
 s:=opendialog1.FileName;
 AssignFile(f, OpenDialog1.FileName);Reset(f);
 nf:=nf+1;
```

```
 nn:=0;
 readln(f,s);readln(f,s1);readln(f,s2);readln(f,s3);readln(f,s4);
 if true then
  begin
  while not(eof(f)) do
   begin
   readln(f,s);s:=trim(s);
   val(s,lu,code);
   if code=0 then nn:=nn+1;
   end;
  setlength(tt[nf],nn);
  reset(f);
  limi[nf][3]:=nn;
  echelley[nf]:=1;
  limi[nf][1]:=65535;limi[nf][2]:=-1000;
  i:=0;
  while not(eof(f)) do
   begin
   readln(f,s);s:=trim(s);
   val(s,lu,code);
   if code=0 then
    begin
    tt[nf][i]:=lu;
    if lu>limi[nf][2] then limi[nf][2]:=lu;
    if lu<limi[nf][1] then limi[nf][1]:=lu;
    i:=i+1;
    end;
   end;
  if nn>nmax then nmax:=nn;
  calculer;
  trace;
  end;
closefile(f);
```

```
 end;
end;



procedure TForm_trace.Ajouterr1Click(Sender: TObject);
var  f:textfile;
    i,nn,lu,code:integer;
    s:string;
begin
if (opendialog1.Execute) then
 begin
 AssignFile(f, OpenDialog1.FileName);Reset(f);
 nf:=nf+1;
 nn:=0;
 while not(eof(f)) do
  begin
  readln(f,s);s:=trim(s);
  val(s,lu,code);
  if code=0 then nn:=nn+1;
  end;

 memo1.Lines.Add('taille='+inttostr(nn));
 setlength(tt[nf],nn);
 reset(f);
 limi[nf][3]:=nn;
 limi[nf][1]:=65535;limi[nf][2]:=-1000;
 i:=0;
 while not(eof(f)) do
   begin
   readln(f,s);s:=trim(s);
   val(s,lu,code);
   if code=0 then
    begin
```

```
      lu:=65535-lu;
      tt[nf][i]:=lu;
      if lu>limi[nf][2] then limi[nf][2]:=lu;
      if lu<limi[nf][1] then limi[nf][1]:=lu;
      i:=i+1;
      end;
    end;
 closefile(f);
 if nn>nmax then nmax:=nn;
  calculer;
 trace;
 end;
end;


procedure TForm_trace.annuler1Click(Sender: TObject);
var i:integer;
begin
if nf>1 then
  begin
  setlength(tt[nf],0);
  limi[nf][3]:=0;
  nf:=nf-1;
  end;
 nmax:=0;
 for I := 1 to nf do if nmax<limi[i][3] then nmax:=limi[i][3];
 calculer;
 trace;
 end;


procedure TForm_trace.Button2Click(Sender: TObject);
```

```
begin
panel3.Visible:=false;
panel4.Visible:=false;
panel5.Visible:=false;
end;



procedure TForm_trace.ajouterClick(Sender: TObject);
var  f:textfile;
     i,nn,lu:integer;
     s:string;
begin
if (opendialog1.Execute) then
 begin
 AssignFile(f, OpenDialog1.FileName);Reset(f);
 nf:=nf+1;
 nn:=0;
 while not(eof(f)) do
  begin
  readln(f,s);nn:=nn+1;
  end;
 memo1.Lines.Add('taille='+inttostr(nn));
 setlength(tt[nf],nn); reset(f);
 limi[nf][3]:=nn;
 limi[nf][1]:=20000;limi[nf][2]:=-1000;
 for i:=0 to nn-1  do
    begin
    readln(f,lu);
    tt[nf][i]:=lu;
    if lu>limi[nf][2] then limi[nf][2]:=lu;
    if lu<limi[nf][1] then limi[nf][1]:=lu;
    end;
 closefile(f);
```

```
 if nn>nmax then nmax:=nn;
  calculer;
 trace;
 end;
end;



procedure TForm_trace.FormCreate(Sender: TObject);
begin
 nf:=0;
 Bm1 := Tbitmap.create;
 Bm2 := Tbitmap.create;
 forcecurrentdirectory:=true;
 echellex:=24/5000;
end;



procedure TForm_trace.freqcardiaque1Click(Sender: TObject);
var dx:real;
    s:string;
begin
dx:=(strtofloat(labx1.caption)-strtofloat(labx0.caption))*echellex;
dx:=60/dx;
s:='fc='+copy(floattostr(dx),1,3)+'b/mn';
label7.Caption:=s;
end;

procedure TForm_trace.inverse1Click(Sender: TObject);
var y0,i,j,yhaut,ybas:integer;
begin
y0:=strtoint(vertical.Caption);

if nf>0 then
```

```
begin
i:=0;
repeat
 i:=i+1;
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;

 until ((y0>=ybas) and (y0<=yhaut)) or (i=nf);
limi[i][1]:=65535;limi[i][2]:=-50000;
for j:=0 to limi[i][3]-1 do
  begin
  tt[i][j]:=65535-tt[i][j];
  if tt[i][j]<limi[i][1] then limi[i][1]:=tt[i][j];
  if tt[i][j]>limi[i][2] then limi[i][2]:=tt[i][j];
  end;
trace;
end;
end;


procedure TForm_trace.Memo1DblClick(Sender: TObject);
begin
memo1.Clear;
end;

procedure TForm_trace.ScrollBar3Change(Sender: TObject);
var i:byte;
begin
i:=panel4.Tag;
tt[i][0]:=scrollbar3.Position;
trace;
end;
```

```
procedure TForm_trace.origine1Click(Sender: TObject);
var y0,i,yhaut,ybas:integer;
begin
y0:=strtoint(vertical.Caption);
if nf>0 then
begin
panel4.Left:=paintbox1.Width div 2-140;panel4.top:=paintbox1.Height div 2-75;
panel4.Visible:=true;
i:=0;
repeat
 i:=i+1;
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;
 until ((y0>=ybas) and (y0<=yhaut)) or (i=nf);
memo1.Lines.Add(inttostr(i));
panel4.Tag:=i;
scrollbar3.Max:=limi[i][2];
scrollbar3.Min:=limi[i][1];
scrollbar3.position:=tt[i][0];
trace;
end;
end;

procedure TForm_trace.calculer;
begin
xlarg:=paintbox1.Width;
if (nmax-(10*xlarg) div scrollbar1.Position>0)  then
   begin
   scrollbar2.Max:=nmax-(10*xlarg)div scrollbar1.Position ;
   scrollbar2.Visible:=true;
```

```
   end else
 begin
 scrollbar2.Visible:=false;
 scrollbar2.Position:=0;
 end;
label2.Caption:='Points:'+inttostr(nmax);
bm1.canvas.Pen.color := clyellow;
bm1.Canvas.Pen.width := 1;
bm2.Canvas.Pen.width := 1;
end;



procedure TForm_trace.code1Click(Sender: TObject);
var s:string;
lu,code:integer;
begin
lu:=11;
s:='67a';s:=trim(s);
val(s,lu,code);
label5.Caption:=inttostr(code)+':'+inttostr(lu)+':'+inttostr(lu*2);
end;

procedure TForm_trace.ddd1Click(Sender: TObject);
var  f:textfile;
    i,nn,lu,code:integer;
    s:string;
begin
if (opendialog1.Execute) then
 begin
 AssignFile(f, OpenDialog1.FileName);Reset(f);
 nf:=nf+1;
 nn:=0;
 while not(eof(f)) do
```

```
 begin
 readln(f,s);s:=trim(s);
 val(s,lu,code);
 if code=0 then nn:=nn+1;
 end;

memo1.Lines.Add('taille='+inttostr(nn));
setlength(tt[nf],nn);
reset(f);
limi[nf][3]:=nn;
limi[nf][1]:=65535;limi[nf][2]:=-1000;
i:=0;
while not(eof(f)) do
  begin
  readln(f,s);s:=trim(s);
  val(s,lu,code);
  if code=0 then
    begin
    tt[nf][i]:=lu;
    if lu>limi[nf][2] then limi[nf][2]:=lu;
    if lu<limi[nf][1] then limi[nf][1]:=lu;
    if i<5 then memo1.Lines.Add(inttostr(tt[nf][i]));
    i:=i+1;
    end;
  end;
closefile(f);
if nn>nmax then nmax:=nn;
 calculer;
trace;
end;
end;
```

```
procedure TForm_trace.delpcer1Click(Sender: TObject);
var y0,i,yhaut,ybas:integer;
begin
y0:=strtoint(vertical.Caption);
if nf>0 then
begin
panel2.Visible:=true;
i:=0;
repeat
 i:=i+1;
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;
 until ((y0>=ybas) and (y0<=yhaut)) or (i=nf);
memo1.Lines.Add(inttostr(i));
panel2.Tag:=i;
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;
 scrollbar4.Min:=0;  scrollbar4.Max:=paintbox1.Height;
 scrollbar4.Position:=yhaut-ybas;
 scrollbar5.Min:=0;  scrollbar5.Max:=paintbox1.Height;
 scrollbar5.Position:=ybas;

 xlarg:=paintbox1.Width;
 xlarg:=paintbox1.Width;

traceseul(i);
end;
end;

procedure TForm_trace.Echellehoriz1Click(Sender: TObject);
begin
if nf>0  then
  begin
```

```
  end;
end;

procedure TForm_trace.Button5Click(Sender: TObject);
var i:byte;
begin
i:=panel5.Tag;
echellex:=strtofloat(edit2.Text)/limi[i][3];
panel5.Visible:=false;
end;




procedure TForm_trace.echelleTemps1Click(Sender: TObject);
var y0,i,yhaut,ybas:integer;
begin
y0:=strtoint(vertical.Caption);
if nf>0 then
begin
panel5.Left:=paintbox1.Width div 2-140;panel5.top:=paintbox1.Height div 2-75;
panel5.Visible:=true;
i:=0;
repeat
 i:=i+1;
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;
 until ((y0>=ybas) and (y0<=yhaut)) or (i=nf);
memo1.Lines.Add(inttostr(i));
panel5.Tag:=i;
trace;
end;
end;
```

```
procedure TForm_trace.Volume1Click(Sender: TObject);
var y0,i,j,yhaut,ybas:integer;
   vol:real;
begin
y0:=strtoint(vertical.Caption);
if nf>0 then
begin
i:=0;
repeat
 i:=i+1;
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;
 until ((y0>=ybas) and (y0<=yhaut)) or (i=nf);

j:=strtoint(labx0.Caption);
vol:=tt[i][j]-tt[i][0];
j:=j+1;
 repeat
 vol:=vol+2*(tt[i][j]-tt[i][0]);
 j:=j+1;
 until j>=strtoint(labx1.Caption);
 vol:=echellex*echelley[i]*(vol+tt[i][j]-tt[i][0])/(2*(limi[i][2]-limi[i][1]));
label6.Caption:='Volumr='+copy(floattostr(vol),1,6);
end;
end;


procedure TForm_trace.Button1Click(Sender: TObject);
var i,code:integer;
   s:string;
   j:real;
begin
```

```
i:=panel3.tag;
s:=edit1.Text;
val(s,j,code);
if code=0 then
  begin
  echelley[i]:=j;
  memo1.Lines.Add(floattostr(j));
  end;
trace;
panel3.Visible:=false;
panel4.Visible:=false;
panel5.Visible:=false;
end;


procedure TForm_trace.echelleY1Click(Sender: TObject);
var y0,i,yhaut,ybas:integer;
begin
y0:=strtoint(vertical.Caption);
if nf>0 then
begin
panel3.Left:=paintbox1.Width div 2-140;panel3.top:=paintbox1.Height div 2-75;

panel3.Visible:=true;
i:=0;
repeat
 i:=i+1;
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;
 until ((y0>=ybas) and (y0<=yhaut)) or (i=nf);
edit1.Text:=floattostr(echelley[i]);
panel3.tag:=i;
memo1.Lines.Add(inttostr(i));
```

```
end;
end;

procedure TForm_trace.effacertout1Click(Sender: TObject);
begin
nf:=0;
nmax:=0;
calculer;
trace;
end;

procedure TForm_trace.enlever1Click(Sender: TObject);
begin
end;

procedure TForm_trace.enlevertout1Click(Sender: TObject);
begin
nf:=0;
nmax:=0;
calculer;
trace;
end;

procedure TForm_trace.PaintBox1MouseUp(Sender: TObject;Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
var i:integer;
begin
i:=(x*10) div scrollbar1.Position+scrollbar2.Position;
label5.Caption:='Point:'+inttostr(i);
vertical.Caption:=inttostr(y);
end;

procedure TForm_trace.PaintBox1MouseMove(Sender: TObject;Shift: TShiftState; X, Y: Integer);
```

```
var t0,t1:integer;
   i:integer;
   s:string;
   tt:trect;
   yhaut,ybas,y0,y1,ymoy:integer;
   dx,dy:real;
begin
labpx.Caption:=inttostr((x*10) div scrollbar1.Position+scrollbar2.Position);
if (shift<>[ssleft]) and (llabel5.caption='') then
begin
str(x,s);llabel10.caption:=s;
str(y,s);llabel11.caption:=s;
end;

if (shift=[ssleft]) and(llabel5.caption='') and (nf>0) then
begin
labx0.Caption:=inttostr((x*10) div scrollbar1.Position+scrollbar2.Position);
llabel5.caption:='*';
str(x,s);llabel10.caption:=s;
str(y,s);llabel11.caption:=s;
llabel12.caption:=llabel10.caption;
llabel13.caption:=llabel11.caption;
end;

if (shift=[ssleft]) and(llabel5.caption<>'') then
begin
labx1.Caption:=inttostr((x*10) div scrollbar1.Position+scrollbar2.Position);
str(x,s);llabel12.caption:=s;
str(y,s);llabel13.caption:=s;
Canvas.Brush.Color := clwhite;
canvas.Pen.Style:= psDot;
Canvas.Pen.Width := 1;
```

```
if strtoint(llabel11.caption)>= strtoint(llabel13.caption) then
 begin
 tt.top:=strtoint(llabel13.caption);
 tt.bottom:=strtoint(llabel11.caption);
 end else
 begin
 tt.top:=strtoint(llabel11.caption);
 tt.bottom:=strtoint(llabel13.caption);
 end;

if strtoint(llabel10.caption)>= strtoint(llabel12.caption) then
 begin
 tt.Left:=strtoint(llabel12.caption);
 tt.right:=strtoint(llabel10.caption);
 end else
 begin
 tt.left:=strtoint(llabel10.caption);
 tt.right:=strtoint(llabel12.caption);
 end;

trace;
i:=scrollbar1.position+strtoint(llabel10.caption);
label2.caption:=inttostr(i);
canvas.MoveTo(tt.left,tt.Top);
canvas.LineTo(tt.right,tt.Top);canvas.LineTo(tt.right,tt.Bottom);
canvas.LineTo(tt.Left,tt.Bottom);canvas.LineTo(tt.Left,tt.Top);
end;
if (shift<>[ssleft])and (llabel5.caption='*') then
begin
llabel5.caption:='';
if strtoint(labx0.Caption)>strtoint(labx1.Caption) then
   begin
   s:=labx0.Caption;
```

```
   labx0.Caption:=labx1.Caption;
   labx1.Caption:=s;
   end;
y0:=strtoint(llabel11.Caption);
y1:=strtoint(llabel13.Caption);
i:=0;
repeat
 i:=i+1;
 yhaut:=((paintbox1.Height)*i) div nf-2;
 ybas:=((paintbox1.Height)*(i-1)) div nf+2;
 ymoy:=(y1-y0) div 2 +y0;
 until ((ymoy>=ybas) and (ymoy<=yhaut)) or (i=nf);

 t0:=trunc((y0-yhaut)*(limi[i][2]-limi[i][1])/(ybas-yhaut)+limi[i][1]);
 t1:=trunc((y1-yhaut)*(limi[i][2]-limi[i][1])/(ybas-yhaut)+limi[i][1]);
laby0.caption:=floattostr(t0);
laby1.caption:=floattostr(t1);
dy:=echelley[i]*(t0-t1)/(limi[i][2]-limi[i][1]);
dx:=(strtofloat(labx1.caption)-strtofloat(labx0.caption))*echellex;
s:='T='+copy(floattostr(dx),1,4)+'s'+'  dy='+copy(floattostr(dy),1,4);
label10.Caption:=s;
end;
end;

procedure TForm_trace.test1Click(Sender: TObject);
var tt:trect;
   Points: array of TPoint;
begin
paintbox1.Canvas.Brush.Color := clred;
paintbox1.Canvas.Pen.Width := 1;
paintbox1.canvas.Pen.Style:= psDot;
tt.Left:=10;tt.Top:=10;tt.Right:=100;tt.Bottom:=100;
paintbox1.Canvas.FrameRect(tt);
```

```
 SetLength(Points, 3);
 Points[0] := Point(5, 5);
 Points[1] := Point(55, 5);
 Points[2] := Point(30, 30);
 Canvas.Pen.Width := 2;
 Canvas.Pen.Color := clyellow;
 Canvas.Brush.Color := clYellow;
 Canvas.Polygon(Points);
end;

procedure TForm_trace.Button7Click(Sender: TObject);
begin
panel2.Visible:=false;
trace;
end;

procedure TForm_trace.ScrollBar4Change(Sender: TObject);
begin
trace;
traceseul(panel2.Tag);
end;

procedure TForm_trace.ScrollBar5Change(Sender: TObject);
begin
trace;
traceseul(panel2.Tag);
end;

procedure TForm_trace.Memo1Click(Sender: TObject);
begin
memo1.Lines.Add('point labx0='+labx0.Caption);
memo1.Lines.Add('point labx1='+labx1.Caption);
memo1.Lines.Add('point vertical='+vertical.Caption);
```

```
memo1.Lines.Add('point y0='+laby0.Caption);
memo1.Lines.Add('point y1='+laby1.Caption);
memo1.Lines.Add('echellex= '+floattostr(echellex));
memo1.Lines.Add('echelley= '+floattostr(echelley[1]/(limi[1][2]-limi[1][1])));
end;

end.
```

Listing assembleur sur Microcontrôleur 18F255

```
      LIST      p=18F2550
     #include <p18F2550.inc>
#DEFINE clk PORTB,1
#DEFINE scx PORTB,0

        cblock 0
        acc00
        acc01
        acc02
        acc03
        acc04
        acc05
        acc06
        acc07
        acc08
        acc09
        acc0A
        acc0B
        acc0C
        acc0D
        acc0E
        acc0F
        acc10
        sauv1
        sauv2
        emiss,recep
        endc
            org 0
              goto    p1

              return  0
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0xFFFF
              Data    0x0001
              decf    0xF2 , F , BANKED
              nop
              Data    0x00BB
              nop
```

```
              Data    0x0002
              nop

;====================
p4            movlb   0x04
              movf    0x10 , W , BANKED
              andlw   0x60
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              iorlw   0x00
              bz      p21
              bra     p22
p21           movf    0x11 , W , BANKED
              xorlw   0x07
              bz      p23
              xorlw   0x0B
              bz      p23
              xorlw   0x07
              bz      p24
              xorlw   0x01
              bz      p25
              xorlw   0x0B
              bz      p26
              xorlw   0x02
              bz      p26
              xorlw   0x03
              bz      p27
              xorlw   0x08
              bz      p28
              xorlw   0x01
              bz      p29
              xorlw   0x0F
              bz      p30
              xorlw   0x03
              bz      p31
              bra     p23
p31           movlb   0x00
              movlw   0x01
              movwf   0xAB     , BANKED
              movlb   0x00
              movlw   0x04
              movwf   0xB2     , BANKED
              bra     p22
p30           rcall   p32
              bra     p22
p29           rcall   p33
              bra     p22
p28           movlb   0x00
              movlw   0x01
              movwf   0xAB     , BANKED
              movlb   0x00
              movlw   0xB4
              movwf   0xAC     , BANKED
              movlw   0x00
              movwf   0xAD     , BANKED
```

```
              movlb   0x00
              bcf     0xB3 , 1 , BANKED
              movlb   0x00
              movlw   0x01
              movwf   0xB0     , BANKED
              bra     p22
p27           rcall   p34
              bra     p22
p26           rcall   p35
              bra     p22
p25           movlb   0x00
              movlw   0x01
              movwf   0xAB     , BANKED
              movlb   0x04
              movf    0x14 , W , BANKED
              movlb   0x00
              movwf   0xAC     , BANKED
              clrf    0xAD     , BANKED
              movlw   0xB5
              addwf   0xAC , F , BANKED
              movlw   0x00
              addwfc  0xAD , F , BANKED
              movlb   0x00
              bcf     0xB3 , 1 , BANKED
              movlb   0x00
              movlw   0x01
              movwf   0xB0     , BANKED
              bra     p22
p24           movlb   0x00
              movlw   0x01
              movwf   0xAB     , BANKED
              movlb   0x04
              movf    0x14 , W , BANKED
              clrf    FSR0H         , ACCESS
              addlw   0xB5
              movwf   FSR0L         , ACCESS
              movlw   0x00
              addwfc  FSR0H     , F , ACCESS
              movff   0412 , INDF0

              bra     p22
p23           bra     p22
p22           return  0

;===================================
p32           movlw   0x80
              movlb   0x04
              subwf   0x10 , W , BANKED
              bnz     p36
              movf    0x13 , W , BANKED
              xorlw   0x03
              bz      p37
              xorlw   0x01
              bz      p38
              xorlw   0x03
              bz      p39
              bra     p40
p39           movlb   0x00
```

```
                movlw   0x01
                movwf   0xAB     , BANKED
                movlb   0x00
                movlw   0xAE
                movwf   0xAC     , BANKED
                movlw   0x0B
                movwf   0xAD     , BANKED
                movlw   0x12
                movlb   0x00
                movwf   0xB0     , BANKED
                clrf    0xB1     , BANKED
                bra     p40
p38             movlb   0x00
                movlw   0x01
                movwf   0xAB     , BANKED
                movlb   0x04
                movf    0x12 , W , BANKED
                movwf   TBLPTRL       , ACCESS
                clrf    TBLPTRH       , ACCESS
                bcf     STATUS    , 0 , ACCESS
                rlcf    TBLPTRL   , F , ACCESS
                rlcf    TBLPTRH   , F , ACCESS
                movlw   0xB6
                addwf   TBLPTRL   , F , ACCESS
                movlw   0x0C
                addwfc  TBLPTRH   , F , ACCESS
                TBLRD*+
                movff   TABLAT    , 00AC

                TBLRD*-
                movff   TABLAT    , 00AD

                movlw   0x02
                movlb   0x00
                clrf    TBLPTRH       , ACCESS
                addwf   0xAC , W , BANKED
                movwf   TBLPTRL       , ACCESS
                movf    0xAD , W , BANKED
                addwfc  TBLPTRH   , F , ACCESS
                TBLRD*+
                movff   TABLAT    , 00B0

                TBLRD*-
                movff   TABLAT    , 00B1

                bra     p40
p37             movlb   0x00
                movlw   0x01
                movwf   0xAB     , BANKED
                movlb   0x04
                movf    0x12 , W , BANKED
                movwf   TBLPTRL       , ACCESS
                clrf    TBLPTRH       , ACCESS
                bcf     STATUS    , 0 , ACCESS
                rlcf    TBLPTRL   , F , ACCESS
                rlcf    TBLPTRH   , F , ACCESS
                movlw   0xBA
                addwf   TBLPTRL   , F , ACCESS
```

```
                movlw   0x0C
                addwfc  TBLPTRH   , F , ACCESS
                TBLRD*+
                movff   TABLAT    , 00AC

                TBLRD*-
                movff   TABLAT    , 00AD

                movff   00AC , TBLPTRL

                movff   00AD , TBLPTRH

                TBLRD*
                movf    TABLAT    , W , ACCESS
                movlb   0x00
                movwf   0xB0      , BANKED
                clrf    0xB1      , BANKED
                bra     p40
p40             movlb   0x00
                bsf     0xB3 , 1 , BANKED
p36             return  0

;===================================
p33             movlb   0x00
                movlw   0x01
                movwf   0xAB      , BANKED
                movlw   0x0F
                movwf   POSTINC1      , ACCESS
                movlw   0x71
                movwf   POSTINC1      , ACCESS
                movlw   0x0F
                movwf   POSTINC1      , ACCESS
                call    p2        , 0

                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movlw   0x01
                movwf   POSTINC1      , ACCESS
                movlw   0xB5
                movwf   POSTINC1      , ACCESS
                movlw   0x00
                movwf   POSTINC1      , ACCESS
                call    p2        , 0

                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movff   0412 , 00B4

                movlb   0x04
                movf    0x12 , W , BANKED
                bnz     p41
                movlb   0x00
                movlw   0x05
                movwf   0xB2      , BANKED
                bra     p42
p41             movlb   0x00
```

```
              movlw   0x06
              movwf   0xB2     , BANKED
              call    p3       , 0

p42           return  0

;==================================
p34           movlb   0x04
              clrf    0x30     , BANKED
              clrf    0x31     , BANKED
              movlb   0x04
              movf    0x10 , W , BANKED
              andlw   0x1F
              xorlw   0x02
              bz      p43
              xorlw   0x03
              bz      p44
              xorlw   0x01
              bz      p45
              bra     p46
p45           movlb   0x00
              movlw   0x01
              movwf   0xAB     , BANKED
              movlb   0x04
              bsf     0x30 , 0 , BANKED
              movlb   0x00
              btfss   0xB3 , 0 , BANKED
              bra     p47
              movlb   0x04
              bsf     0x30 , 1 , BANKED
p47           bra     p46
p44           movlb   0x00
              movlw   0x01
              movwf   0xAB     , BANKED
              bra     p46
p43           movlb   0x00
              movlw   0x01
              movwf   0xAB     , BANKED
              movlb   0x04
              movf    0x14 , W , BANKED
              andlw   0x0F
              mullw   0x08
              movf    PRODL    , W , ACCESS
              clrf    acc0C    , ACCESS
              addlw   0x00
              movwf   acc0B    , ACCESS
              movlw   0x04
              addwfc  acc0C , F , ACCESS
              movf    0x14 , W , BANKED
              andlw   0x80
              bz      p48
              movlw   0x01
p48           mullw   0x04
              movf    PRODL    , W , ACCESS
              movlb   0x00
              addwf   acc0B , W , ACCESS
              movwf   0xAE     , BANKED
              movlw   0x00
```

```
                addwfc  acc0C , W , ACCESS
                movwf   0xAF     , BANKED
                movff   00AE , FSR0L

                movff   00AF , FSR0H

                movf    INDF0     , W , ACCESS
                andlw   0x04
                bz      p49
                movlb   0x04
                movlw   0x01
                movwf   0x30     , BANKED
p49             bra     p46
p46             movlb   0x00
                decf    0xAB , W , BANKED
                bnz     p50
                movlb   0x00
                movlw   0x30
                movwf   0xAC     , BANKED
                movlw   0x04
                movwf   0xAD     , BANKED
                movlb   0x00
                bcf     0xB3 , 1 , BANKED
                movlb   0x00
                movlw   0x02
                movwf   0xB0     , BANKED
p50             return  0

;================================
p35             movlb   0x04
                decf    0x12 , W , BANKED
                bnz     p51
                movf    0x10 , W , BANKED
                andlw   0x1F
                iorlw   0x00
                bnz     p51
                movlb   0x00
                movlw   0x01
                movwf   0xAB     , BANKED
                movlw   0x03
                movlb   0x04
                subwf   0x11 , W , BANKED
                bnz     p52
                movlb   0x00
                bsf     0xB3 , 0 , BANKED
                bra     p51
p52             movlb   0x00
                bcf     0xB3 , 0 , BANKED
p51             movlb   0x04
                movf    0x12 , W , BANKED
                bnz     p53
                movf    0x10 , W , BANKED
                andlw   0x1F
                sublw   0x02
                bnz     p53
                movf    0x14 , W , BANKED
                andlw   0x0F
                iorlw   0x00
```

```
                bz      p53
                movlb   0x00
                movlw   0x01
                movwf   0xAB     , BANKED
                movlb   0x04
                movf    0x14 , W , BANKED
                andlw   0x0F
                mullw   0x08
                movf    PRODL    , W , ACCESS
                clrf    acc0C    , ACCESS
                addlw   0x00
                movwf   acc0B    , ACCESS
                movlw   0x04
                addwfc  acc0C , F , ACCESS
                movf    0x14 , W , BANKED
                andlw   0x80
                bz      p54
                movlw   0x01
p54             mullw   0x04
                movf    PRODL    , W , ACCESS
                movlb   0x00
                addwf   acc0B , W , ACCESS
                movwf   0xAE     , BANKED
                movlw   0x00
                addwfc  acc0C , W , ACCESS
                movwf   0xAF     , BANKED
                movlw   0x03
                movlb   0x04
                subwf   0x11 , W , BANKED
                bnz     p55
                movlw   0x84
                movff   00AE , FSR0L

                movff   00AF , FSR0H

                movwf   INDF0         , ACCESS
                bra     p53
p55             btfss   0x14 , 7 , BANKED
                bra     p56
                movff   00AE , FSR0L

                movff   00AF , FSR0H

                clrf    INDF0         , ACCESS
                bra     p53
p56             movlw   0x88
                movff   00AE , FSR0L

                movff   00AF , FSR0H

                movwf   INDF0         , ACCESS
p53             return  0

;================================
p9              movf    USTAT    , W , ACCESS
                bnz     p57
                movlb   0x04
                movf    0x00 , W , BANKED
```

```
              andlw   0x3C
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              sublw   0x0D
              bnz     p58
              rcall   p59
              bra     p60
p58           rcall   p61
p60           bra     p62
p57           movlw   0x04
              subwf   USTAT     , W , ACCESS
              bnz     p62
              rcall   p63
p62           return  0

;==================================
p59           movff   FSR2L     , POSTINC1

              movff   FSR1L     , FSR2L

              movf    POSTINC1  , F , ACCESS
              movlb   0x00
              clrf    0xAA      , BANKED
              clrf    0xAB      , BANKED
              clrf    0xB0      , BANKED
              clrf    0xB1      , BANKED
              call    p4        , 0

              clrf    INDF2           , ACCESS
p68           movf    INDF2     , W , ACCESS
              movwf   acc0B     , ACCESS
              clrf    acc0C     , ACCESS
              movlw   0x01
              subwf   acc0B , W , ACCESS
              movlw   0x00
              subwfb  acc0C , W , ACCESS
              bc      p64
              movlb   0x00
              movf    0xAB , W , BANKED
              bz      p65
              bra     p64
p65           clrf    TBLPTRH         , ACCESS
              rlcf    INDF2     , W , ACCESS
              andlw   0xFE
              rlcf    TBLPTRH   , F , ACCESS
              movwf   TBLPTRL         , ACCESS
              movlw   0xC0
              addwf   TBLPTRL   , F , ACCESS
              movlw   0x0C
              addwfc  TBLPTRH   , F , ACCESS
              TBLRD*+
              movff   TABLAT    , 000B

              TBLRD*
              movff   TABLAT    , 000C

              bra     p66
p67           movff   000C , PCLATH
```

```
              movf    acc0B , W , ACCESS
              movwf   PCL           , ACCESS
p66           rcall   p67
              incf    INDF2     , F , ACCESS
              bra     p68
p64           rcall   p69
              movf    POSTDEC1  , F , ACCESS
              movf    POSTDEC1  , F , ACCESS
              movff   INDF1     , FSR2L

              return  0

;================================
p61           movlw   0x02
              movlb   0x00
              subwf   0xAA , W , BANKED
              bnz     p70
              rcall   p71
              movlb   0x04
              btfsc   0x00 , 6 , BANKED
              bra     p72
              movlw   0xC8
              movwf   0x00     , BANKED
              bra     p73
p72           movlw   0x88
              movwf   0x00     , BANKED
p73           bra     p74
p70           rcall   uu10
p74           return  0

;====================================
p63           movlw   0x04
              movlb   0x00
              subwf   0xB2 , W , BANKED
              bnz     p75
              movff   0412 , UADDR

              movf    UADDR     , W , ACCESS
              sublw   0x00
              bc      p76
              movlw   0x05
              movwf   0xB2     , BANKED
              bra     p75
p76           movlw   0x03
              movwf   0xB2     , BANKED
p75           movlb   0x00
              decf    0xAA , W , BANKED
              bnz     p77
              rcall   p78
              movlb   0x04
              btfsc   0x04 , 6 , BANKED
              bra     p79
              movlw   0xC8
              movwf   0x04     , BANKED
              bra     p80
p79           movlw   0x88
              movwf   0x04     , BANKED
```

```
p80           bra     p81
p77           rcall   uu10
p81           return  0

;============================
p78           movff   FSR2L     , POSTINC1

              movff   FSR1L     , FSR2L

              movlw   0x02
              addwf   FSR1L     , F , ACCESS
              movlb   0x00
              movlw   0x20
              subwf   0xB0 , W , BANKED
              movlw   0x00
              subwfb  0xB1 , W , BANKED
              bc      p82
              movff   00B0 , POSTINC2

              movff   00B1 , POSTDEC2

              bra     p83
p82           movlw   0x20
              movwf   POSTINC2      , ACCESS
              clrf    POSTDEC2      , ACCESS
p83           movlb   0x04
              bcf     0x04 , 1 , BANKED
              bcf     0x04 , 0 , BANKED
              movlw   0x01
              movf    PLUSW2    , W , ACCESS
              iorwf   0x04 , F , BANKED
              movff   INDF2     , 0405

              movlw   0x00
              movf    PLUSW2    , W , ACCESS
              movlb   0x00
              subwf   0xB0 , F , BANKED
              movlw   0x01
              movf    PLUSW2    , W , ACCESS
              subwfb  0xB1 , F , BANKED
              movlw   0x30
              movwf   0xAE     , BANKED
              movlw   0x04
              movwf   0xAF     , BANKED
              movlb   0x00
              btfss   0xB3 , 1 , BANKED
              bra     p84
p87           movff   FSR2L     , FSR0L

              movff   FSR2H     , FSR0H

              movf    POSTINC0  , W , ACCESS
              iorwf   POSTDEC0  , W , ACCESS
              bz      p85
              movff   00AC , TBLPTRL

              movff   00AD , TBLPTRH
```

```
                TBLRD*
                movf    TABLAT    , W , ACCESS
                movff   00AE , FSR0L

                movff   00AF , FSR0H

                movwf   INDF0         , ACCESS
                movlb   0x00
                incf    0xAE , F , BANKED
                movlw   0x00
                addwfc  0xAF , F , BANKED
                incf    0xAC , F , BANKED
                addwfc  0xAD , F , BANKED
                decf    INDF2     , F , ACCESS
                movlw   0x01
                bc      p86
                decf    PLUSW2    , F , ACCESS
p86             bra     p87
p85             bra     p88
p84             movff   FSR2L     , FSR0L

                movff   FSR2H     , FSR0H

                movf    POSTINC0  , W , ACCESS
                iorwf   POSTDEC0  , W , ACCESS
                bz      p88
                movff   00AC , FSR0L

                movff   00AD , FSR0H

                movf    INDF0     , W , ACCESS
                movff   00AE , FSR0L

                movff   00AF , FSR0H

                movwf   INDF0         , ACCESS
                movlb   0x00
                incf    0xAE , F , BANKED
                movlw   0x00
                addwfc  0xAF , F , BANKED
                incf    0xAC , F , BANKED
                addwfc  0xAD , F , BANKED
                decf    INDF2     , F , ACCESS
                movlw   0x01
                bc      p89
                decf    PLUSW2    , F , ACCESS
p89             bra     p84
p88             movwf   TABLAT        , ACCESS
                movlw   0x02
                subwf   FSR1L     , W , ACCESS
                bc      p90
                clrf    FSR1L         , ACCESS
                movf    POSTDEC1  , F , ACCESS
p90             movwf   FSR1L         , ACCESS
                movf    TABLAT    , W , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movff   INDF1     , FSR2L
```

```
              return  0

;================================
p71           movff   FSR2L     , POSTINC1

              movff   FSR1L     , FSR2L

              movlw   0x02
              addwf   FSR1L     , F , ACCESS
              movlw   0x03
              movlb   0x04
              andwf   0x00 , W , BANKED
              movwf   INDF1         , ACCESS
              movlw   0x01
              movff   INDF1     , PLUSW2

              movff   0401 , INDF2

              movf    POSTINC2  , W , ACCESS
              movlb   0x00
              addwf   0xB0 , F , BANKED
              movf    POSTDEC2  , W , ACCESS
              addwfc  0xB1 , F , BANKED
              movlw   0x30
              movwf   0xAC      , BANKED
              movlw   0x04
              movwf   0xAD      , BANKED
p93           movff   FSR2L     , FSR0L

              movff   FSR2H     , FSR0H

              movf    POSTINC0  , W , ACCESS
              iorwf   POSTDEC0  , W , ACCESS
              bz      p91
              movff   00AC , FSR0L

              movff   00AD , FSR0H

              movf    INDF0     , W , ACCESS
              movff   00AE , FSR0L

              movff   00AF , FSR0H

              movwf   INDF0         , ACCESS
              incf    0xAE , F , BANKED
              movlw   0x00
              addwfc  0xAF , F , BANKED
              incf    0xAC , F , BANKED
              addwfc  0xAD , F , BANKED
              decf    INDF2     , F , ACCESS
              movlw   0x01
              bc      p92
              decf    PLUSW2    , F , ACCESS
p92           bra     p93
p91           movwf   TABLAT        , ACCESS
              movlw   0x02
              subwf   FSR1L     , W , ACCESS
              bc      p94
```

```
              clrf    FSR1L          , ACCESS
              movf    POSTDEC1  , F , ACCESS
p94           movwf   FSR1L          , ACCESS
              movf    TABLAT    , W , ACCESS
              movf    POSTDEC1  , F , ACCESS
              movff   INDF1     , FSR2L

              return  0

;=================================
p69           movlb   0x00
              movf    0xAB , W , BANKED
              bnz     p95
              movlb   0x04
              movlw   0x20
              movwf   0x01     , BANKED
              movlw   0x10
              movwf   0x02     , BANKED
              movlw   0x04
              movwf   0x03     , BANKED
              movlw   0x84
              movwf   0x00     , BANKED
              movlb   0x04
              movwf   0x04     , BANKED
              bra     p96
p95           movlb   0x04
              btfss   0x10 , 7 , BANKED
              bra     p97
              movlb   0x00
              movf    0xB0 , W , BANKED
              movlb   0x04
              subwf   0x16 , W , BANKED
              movlb   0x00
              movf    0xB1 , W , BANKED
              movlb   0x04
              subwfb  0x17 , W , BANKED
              bc      p98
              movff   0416 , 00B0

              movff   0417 , 00B1

p98           rcall   p78
              movlb   0x00
              movlw   0x01
              movwf   0xAA     , BANKED
              movlb   0x04
              movlw   0x20
              movwf   0x01     , BANKED
              movlw   0x10
              movwf   0x02     , BANKED
              movlw   0x04
              movwf   0x03     , BANKED
              movlw   0x80
              movwf   0x00     , BANKED
              movlb   0x04
              movlw   0x30
              movwf   0x06     , BANKED
              movlw   0x04
```

```
              movwf   0x07     , BANKED
              movlw   0xC8
              movwf   0x04     , BANKED
              bra     p96
p97           movlb   0x00
              movlw   0x02
              movwf   0xAA     , BANKED
              movlb   0x04
              clrf    0x05     , BANKED
              movlw   0xC8
              movwf   0x04     , BANKED
              movlb   0x04
              movlw   0x20
              movwf   0x01     , BANKED
              movlw   0x30
              movwf   0x02     , BANKED
              movlw   0x04
              movwf   0x03     , BANKED
              movlw   0xC8
              movwf   0x00     , BANKED
p96           bcf     UCON      , 4 , ACCESS
              return  0

;=====================
uu10          movlb   0x00
              clrf    0xAA     , BANKED
              movlb   0x04
              movlw   0x20
              movwf   0x01     , BANKED
              movlw   0x10
              movwf   0x02     , BANKED
              movlw   0x04
              movwf   0x03     , BANKED
              movlw   0x88
              movwf   0x00     , BANKED
              movlb   0x04
              clrf    0x04     , BANKED
              return  0
;=====================================================
p12           movlw   0xFC
              movwf   TRISC         , ACCESS
              clrf    PORTC         , ACCESS
              return  0
;============================================================
p100          movlw   0x2A
              movwf   ADCON2        , ACCESS
              movlw   0x0B
              movwf   ADCON1        , ACCESS
              bsf     ADCON0    , 0 , ACCESS
              bcf     ADCON0    , 5 , ACCESS
              bcf     ADCON0    , 4 , ACCESS
              return  0
;=========================================================
p118          movff   FSR2L     , POSTINC1

              movff   FSR1L     , FSR2L

              movf    FSR2L     , W , ACCESS
```

```
               addlw   0xFB
               movwf   FSR0L         , ACCESS
               movlw   0xFF
               addwfc  FSR2H     , W , ACCESS
               movwf   FSR0H         , ACCESS
               movf    POSTINC0  , W , ACCESS
               iorwf   POSTDEC0  , W , ACCESS
               bz      p99
               rcall   p100
p99            movlw   0xFD
               movff   PLUSW2    , 0006

               movlw   0xFE
               movff   PLUSW2    , 0007

               movlw   0x03
               xorwf   acc06 , W , ACCESS
               bnz     p101
               movf    acc07 , W , ACCESS
p101           bz      p102
               movlw   0x02
               xorwf   acc06 , W , ACCESS
               bnz     p103
               movf    acc07 , W , ACCESS
p103           bz      p104
               movlw   0x01
               xorwf   acc06 , W , ACCESS
               bnz     p105
               movf    acc07 , W , ACCESS
p105           bz      p106
               movf    acc06 , W , ACCESS
               bnz     p107
               movf    acc07 , W , ACCESS
p107           bz      p108
               bra     p109
p108           bcf     ADCON0    , 3 , ACCESS
               bcf     ADCON0    , 2 , ACCESS
               bra     p109
p106           bcf     ADCON0    , 3 , ACCESS
               bsf     ADCON0    , 2 , ACCESS
               bra     p109
p104           bsf     ADCON0    , 3 , ACCESS
               bcf     ADCON0    , 2 , ACCESS
               bra     p109
p102           bsf     ADCON0    , 3 , ACCESS
               bsf     ADCON0    , 2 , ACCESS
               bra     p109
p109           bsf     ADCON0    , 1 , ACCESS
p110           btfsc   ADCON0    , 1 , ACCESS
               bra     p110
               movf    POSTDEC1  , F , ACCESS
               movff   INDF1     , FSR2L

               return  0

;============================
ProcessIO      movff   FSR2L     , POSTINC1
```

```
                movff   FSR1L     , FSR2L

                movlw   0x02
                addwf   FSR1L     , F , ACCESS
                movlw   0x40
                movwf   POSTINC1      , ACCESS
                movlw   0x60
                movwf   POSTINC1      , ACCESS
                movlw   0x00
                movwf   POSTINC1      , ACCESS
                call    HIDRxReport, 0

                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                sublw   0x00
                bc      p111
                bsf     PORTC     , 0 , ACCESS
                movlb   0x00
                decf    0x60 , W , BANKED
                bnz     p112
                clrf    POSTINC2      , ACCESS
                clrf    POSTDEC2      , ACCESS
p116            movff   POSTINC2  , 000B

                movff   POSTDEC2  , 000C

                movff   0061 , 000D

                clrf    acc0E     , ACCESS
                btfsc   acc0D , 7 , ACCESS
                setf    acc0E     , ACCESS
                movf    acc0C , W , ACCESS
                xorwf   acc0E , W , ACCESS
                btfss   WREG      , 7 , ACCESS
                bra     p113
                rlcf    acc0E , W , ACCESS
                bra     p114
p113            movf    acc0D , W , ACCESS
                subwf   acc0B , W , ACCESS
                movf    acc0E , W , ACCESS
                subwfb  acc0C , W , ACCESS
p114            bc      p112
                bsf     PORTC     , 1 , ACCESS
                movff   0062 , POSTINC1

                call    Delay10KTCYx , 0

                movf    POSTDEC1  , F , ACCESS
                bcf     PORTC     , 1 , ACCESS
                movff   0062 , POSTINC1

                call    Delay10KTCYx , 0

                movf    POSTDEC1  , F , ACCESS
                incf    INDF2     , F , ACCESS
                movlw   0x01
                bnc     p115
```

```
                incf    PLUSW2    , F , ACCESS
p115            bra     p116
p112            movlb   0x00
                movf    0x60 , W , BANKED
                bnz     p117
                setf    0x60     , BANKED
                movlw   0x01
                movwf   POSTINC1      , ACCESS
                clrf    POSTINC1      , ACCESS
                movlw   0x00
                movwf   POSTINC1      , ACCESS
                clrf    POSTINC1      , ACCESS
                rcall   p118
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movff   ADRESH    , 0061

                movff   ADRESL    , 0062

p117            movlb   0x04
                btfsc   0x0C , 7 , BANKED
                bra     p119
                movlw   0x40
                movwf   POSTINC1      , ACCESS
                movlw   0x60
                movwf   POSTINC1      , ACCESS
                movlw   0x00
                movwf   POSTINC1      , ACCESS
                call    HIDTxReport, 0

                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
p119            bcf     PORTC     , 0 , ACCESS
p111            btfsc   PORTC     , 2 , ACCESS
                bra     p120
                movlb   0x00
                movf    0xBB , W , BANKED
                iorwf   0xBC , W , BANKED
                bnz     p120
                movlb   0x00
                setf    0x60     , BANKED
                movlw   0x01
                movwf   POSTINC1      , ACCESS
                clrf    POSTINC1      , ACCESS
                movlw   0x00
                movwf   POSTINC1      , ACCESS
                clrf    POSTINC1      , ACCESS
                rcall   p118
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movff   ADRESH    , 0061

                movff   ADRESL    , 0062
```

```
                movlb   0x04
                btfsc   0x0C , 7 , BANKED
                bra     p120
                movlw   0x40
                movwf   POSTINC1      , ACCESS
                movlw   0x60
                movwf   POSTINC1      , ACCESS
                movlw   0x00
                movwf   POSTINC1      , ACCESS
                call    HIDTxReport , 0

                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movlw   0x01
                movlb   0x00
                movwf   0xBB     , BANKED
                clrf    0xBC     , BANKED
p120            btfss   PORTC     , 2 , ACCESS
                bra     p121
                movlb   0x00
                clrf    0xBB     , BANKED
                clrf    0xBC     , BANKED
p121            movwf   TABLAT        , ACCESS
;-----------------------------------------------------
                movlw   0x02
                subwf   FSR1L     , W , ACCESS
                bc      p122
                clrf    FSR1L         , ACCESS
                movf    POSTDEC1  , F , ACCESS
p122            movwf   FSR1L         , ACCESS
                movf    TABLAT    , W , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movff   INDF1     , FSR2L

                return  0
                nop
                Data    0xFFFF
                Data    0xFFFF
                Data    0xFFFF
                Data    0xFFFF
                Data    0xFFFF
                Data    0xFFFF
                goto    p1

                return  0

;============================
                movlb   0x04
                movf    0x10 , W , BANKED
                andlw   0x1F
                sublw   0x01
                bz      p123
                bra     p124
p123            movf    0x14 , W , BANKED
                bz      p125
                bra     p124
```

```
p125          movlw   0x06
              subwf   0x11 , W , BANKED
              bnz     p126
              movf    0x13 , W , BANKED
              xorlw   0x23
              bz      p127
              xorlw   0x01
              bz      p128
              xorlw   0x03
              bz      p129
              bra     p130
p129          movlb   0x00
              movlw   0x02
              movwf   0xAB     , BANKED
              movlb   0x00
              decf    0xB4 , W , BANKED
              bnz     p131
              movlb   0x00
              movlw   0xD2
              movwf   0xAC     , BANKED
              movlw   0x0B
              movwf   0xAD     , BANKED
p131          movlw   0x09
              movlb   0x00
              movwf   0xB0     , BANKED
              clrf    0xB1     , BANKED
              bra     p130
p128          movlb   0x00
              movlw   0x02
              movwf   0xAB     , BANKED
              movlb   0x00
              decf    0xB4 , W , BANKED
              bnz     p132
              movlb   0x00
              movlw   0x87
              movwf   0xAC     , BANKED
              movlw   0x0C
              movwf   0xAD     , BANKED
p132          movlb   0x00
              decf    0xB4 , W , BANKED
              bnz     p133
              movlw   0x2F
              movlb   0x00
              movwf   0xB0     , BANKED
              clrf    0xB1     , BANKED
p133          bra     p130
p127          bra     p130
p130          movlb   0x00
              bsf     0xB3 , 1 , BANKED
p126          movlb   0x04
              movf    0x10 , W , BANKED
              andlw   0x60
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              rrncf   WREG      , F , ACCESS
              sublw   0x01
```

```
            bz      p134
            bra     p124
p134        movf    0x11 , W , BANKED
            xorlw   0x0B
            bz      p135
            xorlw   0x08
            bz      p136
            xorlw   0x09
            bz      p137
            xorlw   0x08
            bz      p138
            xorlw   0x0B
            bz      p139
            xorlw   0x08
            bz      p140
            bra     p124
p140        rcall   p141
            bra     p124
p139        rcall   p142
            bra     p124
p138        movlb   0x00
            movlw   0x02
            movwf   0xAB     , BANKED
            movlb   0x00
            movlw   0xB6
            movwf   0xAC     , BANKED
            movlw   0x00
            movwf   0xAD     , BANKED
            movlb   0x00
            bcf     0xB3 , 1 , BANKED
            movlb   0x00
            movlw   0x01
            movwf   0xB0     , BANKED
            bra     p124
p137        movlb   0x00
            movlw   0x02
            movwf   0xAB     , BANKED
            movff   0413 , 00B6

            bra     p124
p136        movlb   0x00
            movlw   0x02
            movwf   0xAB     , BANKED
            movlb   0x00
            movlw   0xB7
            movwf   0xAC     , BANKED
            movlw   0x00
            movwf   0xAD     , BANKED
            movlb   0x00
            bcf     0xB3 , 1 , BANKED
            movlb   0x00
            movlw   0x01
            movwf   0xB0     , BANKED
            bra     p124
p135        movlb   0x00
            movlw   0x02
            movwf   0xAB     , BANKED
            movff   0412 , 00B7
```

```
              bra     p124
p124          return  0
p141          return  0
p142          return  0

;====================================
p3            movlb   0x00
              clrf    0xB8     , BANKED
              movlw   0x1E
              movwf   UEP1          , ACCESS
              movlb   0x04
              movlw   0x40
              movwf   0x09     , BANKED
              movlw   0x50
              movwf   0x0A     , BANKED
              movlw   0x04
              movwf   0x0B     , BANKED
              movlw   0x88
              movwf   0x08     , BANKED
              movlb   0x04
              movlw   0x90
              movwf   0x0E     , BANKED
              movlw   0x04
              movwf   0x0F     , BANKED
              movlw   0x40
              movwf   0x0C     , BANKED
              return  0

;=================
HIDTxReport   movff   FSR2L     , POSTINC1

              movff   FSR1L     , FSR2L

              movf    POSTINC1  , F , ACCESS
              movlw   0xFC
              movf    PLUSW2    , W , ACCESS
              sublw   0x40
              bc      p143
              movlw   0x40
              movwf   PRODL         , ACCESS
              movlw   0xFC
              movff   PRODL     , PLUSW2

p143          clrf    INDF2         , ACCESS
p145          movlw   0xFC
              movf    PLUSW2    , W , ACCESS
              subwf   INDF2     , W , ACCESS
              bc      p144
              movf    INDF2     , W , ACCESS
              movwf   INDF1         , ACCESS
              movlw   0xFD
              movff   PLUSW2    , FSR0L

              movlw   0xFE
              movff   PLUSW2    , FSR0H

              movf    INDF1     , W , ACCESS
```

```
                addwf   FSR0L     , F , ACCESS
                movlw   0x00
                addwfc  FSR0H     , F , ACCESS
                movf    INDF0     , W , ACCESS
                movwf   POSTINC1      , ACCESS
                movf    INDF2     , W , ACCESS
                clrf    FSR0H         , ACCESS
                addlw   0x90
                movwf   FSR0L         , ACCESS
                movlw   0x04
                addwfc  FSR0H     , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    INDF1     , W , ACCESS
                movwf   INDF0         , ACCESS
                incf    INDF2     , F , ACCESS
                bra     p145
p144            movlw   0xFC
                movff   PLUSW2    , 040D

                movlw   0x40
                movlb   0x04
                andwf   0x0C , F , BANKED
                btg     0x0C , 6 , BANKED
                movlw   0x88
                iorwf   0x0C , F , BANKED
                movf    POSTDEC1  , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movff   INDF1     , FSR2L

                return  0

;==================
HIDRxReport     movff   FSR2L     , POSTINC1

                movff   FSR1L     , FSR2L

                movlb   0x00
                clrf    0xB8      , BANKED
                movlb   0x04
                btfsc   0x08 , 7 , BANKED
                bra     p146
                movlw   0xFC
                movff   PLUSW2    , 000B

                movf    0x09 , W , BANKED
                bsf     STATUS    , 0 , ACCESS
                subfwb  acc0B , W , ACCESS
                bc      p147
                movlw   0xFC
                movff   0409 , PLUSW2

p147            movlb   0x00
                clrf    0xB8      , BANKED
p149            movlw   0xFC
                movf    PLUSW2    , W , ACCESS
                subwf   0xB8 , W , BANKED
                bc      p148
                movf    0xB8 , W , BANKED
```

```
                clrf    FSR0H         , ACCESS
                addlw   0x50
                movwf   FSR0L         , ACCESS
                movlw   0x04
                addwfc  FSR0H     , F , ACCESS
                movf    INDF0     , W , ACCESS
                movwf   POSTINC1      , ACCESS
                movf    0xB8 , W , BANKED
                movwf   INDF1         , ACCESS
                movlw   0xFD
                movff   PLUSW2    , FSR0L

                movlw   0xFE
                movff   PLUSW2    , FSR0H

                movf    INDF1     , W , ACCESS
                addwf   FSR0L     , F , ACCESS
                movlw   0x00
                addwfc  FSR0H     , F , ACCESS
                movf    POSTDEC1  , F , ACCESS
                movf    INDF1     , W , ACCESS
                movwf   INDF0         , ACCESS
                incf    0xB8 , F , BANKED
                bra     p149
p148            movlb   0x04
                movlw   0x40
                movwf   0x09      , BANKED
                andwf   0x08 , F , BANKED
                btg     0x08 , 6 , BANKED
                movlw   0x88
                iorwf   0x08 , F , BANKED
p146            movlb   0x00
                movf    0xB8 , W , BANKED
                bra     p150
p150            movf    POSTDEC1  , F , ACCESS
                movff   INDF1     , FSR2L

                return  0

;==============================
p13             btfss   UCON      , 3 , ACCESS
                rcall   p151
                bra     p152
                btfsc   UCON      , 3 , ACCESS
                rcall   p8
p152            movlb   0x00
                decf    0xB2 , W , BANKED
                bnz     p153
                btfsc   UCON      , 5 , ACCESS
                bra     p153
                clrf    UIR           , ACCESS
                clrf    UIE           , ACCESS
                bsf     UIE       , 0 , ACCESS
                bsf     UIE       , 4 , ACCESS
                movlw   0x02
                movwf   0xB2      , BANKED
p153            return  0
```

```
;=========================
p151          clrf    UCON          , ACCESS
              clrf    UIE           , ACCESS
              bsf     UCON      , 3 , ACCESS
              movlb   0x00
              movlw   0x01
              movwf   0xB2     , BANKED
              return  0

;=========================
p8            clrf    UCON          , ACCESS
              clrf    UIE           , ACCESS
              movlb   0x00
              clrf    0xB2     , BANKED
              return  0

;================================
              goto    p8


;============================
uu14          movlb   0x00
              movf    0xB2 , W , BANKED
              bnz     uu154
              bra     uu155
uu154         movf    UIR       , W , ACCESS
              andlw   0x04
              bz      uu156
              movf    UIE       , W , ACCESS
              andlw   0x04
              bz      uu156
              rcall   uu157
uu156         btfsc   UCON      , 1 , ACCESS
              bra     uu155
              movf    UIR       , W , ACCESS
              andlw   0x01
              bz      uu158
              movf    UIE       , W , ACCESS
              andlw   0x01
              bz      uu158
              rcall   uu159
uu158         movf    UIR       , W , ACCESS
              andlw   0x10
              bz      uu160
              movf    UIE       , W , ACCESS
              andlw   0x10
              bz      uu160
              rcall   uu161
uu160         movf    UIR       , W , ACCESS
              andlw   0x40
              bz      uu162
              movf    UIE       , W , ACCESS
              andlw   0x40
              bz      uu162
              rcall   uu163
uu162         movf    UIR       , W , ACCESS
              andlw   0x20
              bz      uu164
```

```
              movf    UIE       , W , ACCESS
              andlw   0x20
              bz      uu164
              rcall   uu165
uu164         movf    UIR       , W , ACCESS
              andlw   0x02
              bz      uu166
              movf    UIE       , W , ACCESS
              andlw   0x02
              bz      uu166
              rcall   uu167
uu166         movlw   0x03
              movlb   0x00
              subwf   0xB2 , W , BANKED
              bc      uu168
              bra     uu155
uu168         movf    UIR       , W , ACCESS
              andlw   0x08
              bz      uu155
              movf    UIE       , W , ACCESS
              andlw   0x08
              bz      uu155
              call    p9       , 0

              bcf     UIR       , 3 , ACCESS
uu155         return  0

;==================================
uu161          bsf     UIE       , 2 , ACCESS
              bcf     UIR       , 4 , ACCESS
              bsf     UCON      , 1 , ACCESS
              bcf     PIR2      , 5 , ACCESS
              bcf     INTCON    , 0 , ACCESS
              bsf     PIE2      , 5 , ACCESS
              bsf     INTCON    , 3 , ACCESS
              sleep
              btfsc   INTCON    , 0 , ACCESS
              rcall   uu169
              bcf     PIE2      , 5 , ACCESS
              bcf     INTCON    , 3 , ACCESS
              return  0

;==========================
uu157         bcf     UCON      , 1 , ACCESS
              bcf     UIE       , 2 , ACCESS
              bcf     UIR       , 2 , ACCESS
              return  0

;============================
uu169         movlb   0x00
              btfss   0xB3 , 0 , BANKED
              bra     uu170
              rcall   uu157
              bsf     UCON      , 2 , ACCESS
              movlb   0x00
              movlw   0x08
              movwf   0xB9     , BANKED
              movlw   0x07
```

```
             movwf   0xBA     , BANKED
uu171        decf    0xB9 , F , BANKED
             movlw   0x00
             subwfb  0xBA , F , BANKED
             movf    0xB9 , W , BANKED
             iorwf   0xBA , W , BANKED
             bnz     uu171
             bcf     UCON     , 2 , ACCESS
uu170        return  0

;=====================================
uu163        bcf     UIR      , 6 , ACCESS
             return  0

;=====================================
uu165        btfss   UEP0     , 0 , ACCESS
             bra     uu172
             call    uu10     , 0

             bcf     UEP0     , 0 , ACCESS
uu172        bcf     UIR      , 5 , ACCESS
             return  0

;=====================================
uu167        bcf     UIR      , 1 , ACCESS
             return  0

;==============================
uu159        clrf    UEIR         , ACCESS
             clrf    UIR          , ACCESS
             movlw   0x9F
             movwf   UEIE         , ACCESS
             movlw   0x7B
             movwf   UIE          , ACCESS
             clrf    UADDR        , ACCESS
             movlw   0x0F
             movwf   POSTINC1     , ACCESS
             movlw   0x71
             movwf   POSTINC1     , ACCESS
             movlw   0x0F
             movwf   POSTINC1     , ACCESS
             rcall   p2
             movf    POSTDEC1 , F , ACCESS
             movf    POSTDEC1 , F , ACCESS
             movf    POSTDEC1 , F , ACCESS
             movlw   0x16
             movwf   UEP0         , ACCESS
uu174        btfss   UIR      , 3 , ACCESS
             bra     uu173
             bcf     UIR      , 3 , ACCESS
             bra     uu174
uu173        bcf     UCON     , 4 , ACCESS
             call    uu10     , 0

             movlb   0x00
             bcf     0xB3 , 0 , BANKED
             movlb   0x00
             clrf    0xB4     , BANKED
```

```
               movlb   0x00
               movlw   0x03
               movwf   0xB2     , BANKED
               return  0
;=====================================================
p2             movff   FSR2L     , POSTINC1

               movff   FSR1L     , FSR2L

               movlw   0xFD
               movff   PLUSW2    , FSR0L

               movlw   0xFE
               movff   PLUSW2    , FSR0H

uu176          movlw   0xFC
               movf    PLUSW2    , W , ACCESS
               bz      uu175
               clrf    POSTINC0      , ACCESS
               movlw   0xFC
               decf    PLUSW2    , F , ACCESS
               bra     uu176
uu175          movf    POSTDEC1  , F , ACCESS
               movff   INDF1     , FSR2L

               return  0

;.................. Descriptor.................................
              data 0x0112
              data 0x0200
              data 0x0000
              data 0x2000
              data 0x14D8
              data 0x0015
              data 0x0001
              data 0x0201
              data 0x0100

              data 0x0209
              data 0x0029
              data 0x0101
              data 0x0A000
              data 0x0932
              data 0x0004
              data 0x0200
              data 0x0003
              data 0x0000
              data 0x2109
              data 0x0101
              data 0x0100
              data 0x2F22
              data 0x0700
              data 0x8105
              data 0x4003
              data 0x0100
              data 0x0507
              data 0x0301
              data 0x0040
```

```
data 0x0401
data 0x0903
data 0x3204

data 0x6803    ;1
data 0x6800    ;2
data 0x6800    ;3
data 0x7000    ;4
data 0x7000    ;5
data 0x7000    ;6
data 0x7000    ;7
data 0x7000    ;8
data 0x7000    ;9
data 0x7000    ;10
data 0x7000    ;11
data 0x7000    ;12
data 0x7000    ;13
data 0x7000    ;14
data 0x7000    ;15
data 0x7000    ;16
data 0x7000    ;17
data 0x7000    ;18
data 0x7000    ;19
data 0x7000    ;20
data 0x7000    ;21
data 0x7000    ;22
data 0x7000    ;23
data 0x7000    ;24
data 0x6800
data 0x5503      ; 1
data 0x5300      ; 2
data 0x5300      ; 3
data 0x5300      ; 4
data 0x5300      ; 5
data 0x5300      ; 6
data 0x5300      ; 7
data 0x5300      ; 8
data 0x5300      ; 9
data 0x5300      ;10
data 0x5300      ;11
data 0x5300      ;12
data 0x5300      ;13
data 0x5300      ;14
data 0x5300      ;15
data 0x5300      ;16
data 0x5300      ;17
data 0x5300      ;18
data 0x5300      ;19
data 0x5300      ;20
data 0x5300      ;21
data 0x5300      ;22
data 0x5300      ;23
data 0x5300      ;24
data 0x5300      ;25
data 0x5300      ;26
data 0x5300      ;27
data 0x5300      ;28
data 0x5300      ;29
```

```
              data 0x5300      ;30
              data 0x5300      ;31
              data 0x5300      ;32
              data 0x5300      ;33
              data 0x5300      ;34
              data 0x5300      ;35
              data 0x5300      ;36
              data 0x5300      ;37
              data 0x5300      ;38
              data 0x5300      ;39
              data 0x5300      ;40
              data 0x5300      ;41
              data 0x5300      ;42
              data 0x5300      ;43
              data 0x5300      ;44
              data 0x5300      ;45
              data 0x5300      ;46
              data 0x5300      ;47
              data 0x5300      ;48
              data 0x5300      ;49
              data 0x5300      ;50
              data 0x5300      ;51

              data 0x0600
              data 0xFFA0
              data 0x0109
              data 0x01A1
              data 0x0309
              data 0x0015
              data 0xFF26
              data 0x7500
              data 0x9540
              data 0x8108
              data 0x0902
              data 0x1504
              data 0x2600
              data 0x00FF
              data 0x4075
              data 0x0895
              data 0x0291
              data 0x0509
              data 0x0015
              data 0xFF26
              data 0x7500
              data 0x9540
              data 0xB108
              data 0xC002
              data 0x0BC0
              data 0x0BC0
              data 0x0BE9
              data 0x0BED
              data 0x0C1F
              data 0x0806
;=================================cinit================
cinit         movlw   0x2A
              movwf   TBLPTRL       , ACCESS
              movlw   0x00
              movwf   TBLPTRH       , ACCESS
```

```
                movlw   0x00
                movwf   TBLPTRU       , ACCESS
                movlb   0x00
                TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   0xA5     , BANKED
                TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   0xA6     , BANKED
aa1             bnz     aa2
                tstfsz  0xA5     , BANKED
                bra     aa2
                bra     aa6
aa2             TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   0xA0     , BANKED
                TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   0xA1     , BANKED
                TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   0xA2     , BANKED
                TBLRD*+
                TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   FSR0L          , ACCESS
                TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   FSR0H          , ACCESS
                TBLRD*+
                TBLRD*+
                TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   0xA3     , BANKED
                TBLRD*+
                movf    TABLAT    , W , ACCESS
                movwf   0xA4     , BANKED
                TBLRD*+
                TBLRD*+
                movff   TBLPTRL   , 00A7

                movff   TBLPTRH   , 00A8

                movff   TBLPTRU   , 00A9

                movff   00A0 , TBLPTRL

                movff   00A1 , TBLPTRH

                movff   00A2 , TBLPTRU

                movlb   0x00
                movf    0xA3 , F , BANKED
aa3             bnz     aa4
                movf    0xA4 , F , BANKED
                bz      aa5
aa4             TBLRD*+
                movf    TABLAT    , W , ACCESS
```

```
              movwf   POSTINC0      , ACCESS
              decf    0xA3 , F , BANKED
              bc      aa3
              decf    0xA4 , F , BANKED
              bra     aa4
aa5           movff   00A7 , TBLPTRL

              movff   00A8 , TBLPTRH

              movff   00A9 , TBLPTRU

              movlb   0x00
              decf    0xA5 , F , BANKED
              movlw   0x00
              subwfb  0xA6 , F , BANKED
              bra     aa1
aa6          return  0

;====================================
main          rcall   IniSyst
p185          rcall   USBTasks
              call    ProcessADC, 0

              bra     p185
              return  0

;============================
IniSyst       movlw   0x0F
              iorwf   ADCON1    , F , ACCESS
              movlw   0x14
              movwf   UCFG          , ACCESS
              movlb   0x00
              clrf    0xB2     , BANKED
              movlb   0x00
              clrf    0xB3     , BANKED
              movlb   0x00
              clrf    0xB4     , BANKED
              goto    p12                ;config PORTC


;==========================================
USBTasks      call    p13      , 0

              btfsc   UCFG      , 7 , ACCESS
              bra     p186
              call    uu14      , 0

p186          return  0

;================================================
Delay10KTCYx  movlw   0xFF
              movf    PLUSW1    , W , ACCESS
              movwf   acc10     , ACCESS
              movlw   0xEF
              bra     p187
p18           movlw   0xF3
p187          movwf   INDF1         , ACCESS
p15           decfsz  INDF1     , F , ACCESS
```

```
              goto    p15

              movlw   0x0C
              movwf   acc0F     , ACCESS
p17           clrf    INDF1         , ACCESS
p16           decfsz  INDF1     , F , ACCESS
              goto    p16

              decfsz  acc0F , F , ACCESS
              goto    p17

              decfsz  acc10 , F , ACCESS
              goto    p18

              return  0
;=============================================================
p1            call clignot
              movlw b'10111100'
              movwf TRISB
               movlw d'15'          ;entree digitales
              movwf ADCON1

              lfsr    0x1 , 0x300

              lfsr    0x2 , 0x300

              clrf    TBLPTRU       , ACCESS
              bcf     acc0A , 6 , ACCESS
              call    cinit    , 0

p188          call    main     , 0

              bra     p188
              return  0
;=============================================================
clignot          bcf TRISB,6
                 movlw 3
                  movwf 0x61
                 movlw d'200'
                 movwf 0x62
cl1              movf 0x61,F
                 btfsc STATUS,Z
                 return
                 decf 0x61
                 bsf PORTB,6
                 call dell
                 bcf PORTB,6
                 call dell
                 bra cl1

dell             movf 0x62,W
                 movwf 0x63
ap2_3            movlw d'20'
                 movwf 0x64
ap2_2            movlw 0x31
                 movwf 0x65
ap2_1            decf 0x65,F
                 bnz ap2_1
```

```
                decf 0x64,F
                bnz ap2_2
                decf 0x63
                bnz ap2_3
                return

;=========================================================
ProcessADC    movff   FSR2L     , POSTINC1
              movff   FSR1L     , FSR2L
              movlw   0x02
              addwf   FSR1L     , F , ACCESS
;-----emission possible ? ----------------------
              movlb   0x04
              btfsc   0x0C , 7 , BANKED
              bra     pp120
;-------emission autorsée ? --------------------
              movlb 0x00
               btfss emiss,0
               bra pp120
;-------debut emission---------------------------
              movlw   0x40
              movwf   POSTINC1      , ACCESS
              movlw   0x60
              movwf   POSTINC1      , ACCESS
              movlw   0x00
              movwf   POSTINC1      , ACCESS
              call    HIDTxReport , 0

              movf    POSTDEC1  , F , ACCESS
              movf    POSTDEC1  , F , ACCESS
              movf    POSTDEC1  , F , ACCESS

             movlb 0x00
             clrf emiss
;----------fin emission--------------------------
;-------test si reception prete------------------
pp120          movlb 0x00
               clrf recep
               movlw   0x40
               movwf   POSTINC1      , ACCESS
               movlw   0x60
               movwf   POSTINC1      , ACCESS
               movlw   0x00
               movwf   POSTINC1      , ACCESS
               call    HIDRxReport, 0
               movf    POSTDEC1  , F , ACCESS
               movf    POSTDEC1  , F , ACCESS
               movf    POSTDEC1  , F , ACCESS
               sublw   0x00
               bc      pp002
              bra pp003
pp002          goto pp200
;----reception faite----------------------------
pp003          setf recep
               movlw 0x15
               subwf 0x60,W
               btfsc STATUS,Z
               bra appl1
```

```
               movlw 0x05
               subwf 0x60,W
                  btfsc STATUS,Z
                  bra appl2
                  movlw 0x10
                  subwf 0x60,W
                  btfsc STATUS,Z
                  bra appl3
                  clrf emiss
                  bra pp200

appl2             movf 0x61,W
                  addwf 0x62,F
                  addwf 0x63,F
                  addwf 0x64,F
                  addwf 0x65,F
                  addwf 0x66,F
                  setf emiss
                  bra pp200

appl3             movf 0x61,W
                  subwf 0x62,F
                  subwf 0x63,F
                  subwf 0x64,F
                  subwf 0x65,F
                  subwf 0x66,F
                  setf emiss
                  bra pp200

appl1
                   bsf scx        ;! ! ! ! !
                  bcf clk
                  bcf scx         ;chip select
                  clrf 0x66
                   clrf 0x67
                 clrf 0x68
                 clrf 0x69
                 clrf 0x6a

                  bsf clk
                  nop
                  nop
                  bcf clk         ;start conversion
                  nop
                  nop
                  nop
                  nop
                  nop
                  nop

                  bsf clk
                  nop
                  bcf clk
                  nop
                  btfsc PORTB,7
                  bsf 0x6a,7
                  btfsc PORTB,2
                  bsf 0x66,7
```

```
btfsc PORTB,3
bsf 0x67,7
btfsc PORTB,4
bsf 0x68,7
btfsc PORTB,5
bsf 0x69,7
bsf clk
nop
bcf clk
nop
btfsc PORTB,7
bsf 0x6a,6
btfsc PORTB,2
bsf 0x66,6
btfsc PORTB,3
bsf 0x67,6
btfsc PORTB,4
bsf 0x68,6
btfsc PORTB,5
bsf 0x69,6

bsf clk
nop
bcf clk
nop
btfsc PORTB,7
bsf 0x6a,5
btfsc PORTB,2
bsf 0x66,5
btfsc PORTB,3
bsf 0x67,5
btfsc PORTB,4
bsf 0x68,5
btfsc PORTB,5
bsf 0x69,5

bsf clk
nop
bcf clk
nop
btfsc PORTB,7
bsf 0x6a,4
btfsc PORTB,2
bsf 0x66,4
btfsc PORTB,3
bsf 0x67,4
btfsc PORTB,4 ;!!!!!!!!!!!!!!!!!!
bsf 0x68,4
btfsc PORTB,5
bsf 0x69,4

bsf clk
nop
bcf clk
nop
btfsc PORTB,7
bsf 0x6a,3
btfsc PORTB,2
```

```
                bsf 0x66,3
                btfsc PORTB,3
                bsf 0x67,3
                btfsc PORTB,4
                bsf 0x68,3
                btfsc PORTB,5
                bsf 0x69,3

                bsf clk
                nop
                bcf clk
                nop
                btfsc PORTB,7
                bsf 0x6a,2
                btfsc PORTB,2
                bsf 0x66,2
                btfsc PORTB,3
                bsf 0x67,2
                btfsc PORTB,4
                bsf 0x68,2
                btfsc PORTB,5
                bsf 0x69,2

                bsf clk
                nop
                bcf clk
                nop
                btfsc PORTB,7
                bsf 0x6a,1
                btfsc PORTB,2
                bsf 0x66,1
                btfsc PORTB,3
                bsf 0x67,1
                btfsc PORTB,4
                bsf 0x68,1
                btfsc PORTB,5
                bsf 0x69,1

                bsf clk
                nop
                bcf clk
                nop
                btfsc PORTB,7
                bsf 0x6a,0
                btfsc PORTB,2
                bsf 0x66,0
                btfsc PORTB,3
                bsf 0x67,0
                btfsc PORTB,4
                bsf 0x68,0
                btfsc PORTB,5
                bsf 0x69,0

                bsf scx
                setf emiss

;----fin reception-----------------------------
pp200         movwf   TABLAT        , ACCESS
```

```
;-----------------------------------------------
             movlw   0x02
             subwf   FSR1L     , W , ACCESS
             bc      pp122
             clrf    FSR1L       , ACCESS
             movf    POSTDEC1  , F , ACCESS
pp122        movwf   FSR1L       , ACCESS
             movf    TABLAT    , W , ACCESS
             movf    POSTDEC1  , F , ACCESS
             movff   INDF1     , FSR2L

             return  0
;===============================================
       ORG 0x300000
       Data   0x0C22
       Data   0x0037
       Data   0x8000
       Data   0x0080
       Data   0xC00F
       Data   0xE00F
       Data   0x400F

       end
```

ملخص

في هذه الأطروحة نقدم تطوير واجهة الإنسان ـ آلة للمعلومات الطبية والإتصالات ـ عن طريق USB- HID من جهاز المراقبة إلى جهاز الكمبيوتر وكذلك الإتصالات عن بعد بواسطة TCP IP . استعملت لغة البرمجة على الحاسوب لغة delphi

لمعالجة المعطيات على الحاسوب مثل التمثيل ، التخزين ، انقلاب ، الإزاحة وحساب البيانات الطبية Visual Basic التداخل البيني و كذلك لغة ASSEMBLEUR .

كلمات مفتاحية :

PPG– PTG الطب عن بعد USB HID TCP HP vr fc .

Résumé

La pratique télé médicale consiste à prélever sur le patient des informations multidimensionnelles et multi médias représentatives de son état physiopathologique, de les faire parvenir à un terminal informatique local dans un premier temps puis à un terminal informatique distant dans un deuxième temps.

Nous nous proposons dans le cadre de ce travail de mettre en œuvre :

4) La réalisation d'une plateforme télé médicale non invasive et non intrusive dédiée à l'acquisition simultanée et en temps réel de jusqu'à 16 signaux physiologiques unidimensionnels.
5) La réalisation d'une interface hardware construite autour des C.A.N. séries AD0830 d'Analog Device assurant la numérisation des signaux et du microcontrôleur 18F2550 de Microchip assurant leurs transferts du patient vers la poste local sous protocole USB avec les spécifications HID.
6) La réalisation d'une interface graphique sous environnement Visual Basic mettant à profit le composant Winsock compatible avec le système d'exploitation Windows et l'architecture client-serveur permettant la connexion des différents terminaux informatiques pour le transfert des données à travers les réseaux télé médicaux sous protocole TCP-IP.

Mots clés— Télémédecine, USB, HID, mcHID.dll, TCP-IP, Microcontrôleur, firmware, VB6, socket, Photopléthysmographie, Electrocardiographie, Spectrophotométries, Télé surveillance.

Abstract

The telemedical practice consists in taking on patient of multidimensional information and multi media representative of its physiopathological state, to do them to arrive to a local computer terminal initially then with a distant computer terminal in the second time.

We propose within the framework of this article to put in work:

4) The realization of a telemedical platform noninvasive and nonintrusive dedicated to simultaneous acquisition and in real-time of to 16 one-dimensional physiological signals.

5) The realization of an interface hardware built around the C.A.N series AD0830 of Analog Device ensuring the digitalization of the signals and the microcontroller 18F2550 of Microchip ensuring their transfers of the patient towards the local post office under protocol USB with specifications HID.

6) The realization of an graphical interface under environment Visual BASIC making profitable the component Winsock compatible with the operating system Windows and architecture customer-server allowing the connection of the various computer terminals for the transfer of the data through the telemedical networks under protocol TCP-IP.

Key words Télémédecine, USB, HID, mcHID.dll, TCP-IP, Microcontroller, firmware, VB6, socket, Photopléthysmographie, Electrocardiographie, Spectrophotométries.