Universitat de Vic
Escola Politècnica Superior

Treball Final de Carrera

Projecte P.I.T.O.S

(Projecte d'informació telefònica d'orientació al S.A.T.)

David Ruiz Solà

Enginyeria Tècnica de Telecomunicacions especialitzat en Sistemes de Telecomunicació

Tutor: Salvador Farràs Clusella

Avaladora: Bet Faja Grau

Vic, setembre de 2011

En agraïment...

A tot JCM, en especial al departament de R+D
per donar-me l'oportunitat de desenvolupar
aquest projecte.

A Salvador Farràs Clusella per acceptar ser el meu tutor
i la seva ajuda durant el desenvolupament
del projecte.

A Bet Faja Grau per acceptar ser l'avaladora
d'aquest projecte i la seva predisposició d'ajudar
en qualsevol moment.

Gràcies sobretot a tu.
Gràcies a tots els meus amics per fer veure que
saben de que parlo en converses de 1's i 0's. Menció a la Marta

A tots els lectors pel seu interès.

1 Índex

1.1 Índex de continguts

1.2 Índex d'imatges

1.3 Índex de Figures

1.4 Índex de Taules

Universitat de Vic
Escola Politècnica Superior

RESUM DE TREBALL DE FINAL DE CARRERA

ENGINYERIA TÈCNICA DE TELECOMUNICACIONS ESPECIALITZAT EN SISTEMES DE TELECOMUNICACIÓ

Títol: Projecte P.I.T.O.S. (Projecte d'informació telefònica d'orientació al SAT)

Paraules clau: Telecomunicacions, electrònica, informàtica, programació i disseny de hardware

Autor: David Ruiz Solà

Tutor: Salvador Farràs Clusella

Avaladora: Bet Faja Grau

RESUM

JCM és una empresa dedicada al disseny de sistemes de control d'accés. Disposa d'uns equips elèctrics amb molts paràmetres configurables, així es poden utilitzar en molts tipus d'instal·lació. Aquets paràmetres són configurables pels clients. JCM, disposa d'un servei d'atenció telefònica (SAT), que intenta donar solucions a tots els problemes que puguin sorgir als clients. Sovint, no poden donar suport als dubtes dels clients per culpa de la poca informació que reben a través del client.

L'objectiu del projecte és resoldre el problema de falta d'informació i mala comunicació per millorar la qualitat del servei que ofereix el SAT. La solució, no ha d'alterar el cost de producció del producte. S'ha de tenir en compte també que els equips poden estar instal·lats en qualsevol part del món i que només es poden utilitzar recursos de fàcil accés per tots els clients.

Per complir amb aquests objectius, emetrem un missatge, a través del brunzidor del equip, amb la informació de configuració de l'equip. Aquest missatge viatjarà a través de la xarxa telefònica fins al SAT. Un cop allà, el descodificarem i n'enviarem les dades a un PC perquè pugui presentar les dades sobre la configuració de una forma clara pel SAT.

FINAL THESIS REPORT SUMMARY

TELECOMMUNICATIONS ENGINEERING SPECIALIZED IN TELECOMMUNICATIONS SYSTEMS

Title: PITOS Project

Key words: Telecommunications, electronics, computer science, programming and hardware design

Author: David Ruiz Solà

Tutor: Salvador Farràs Clusella

Guarantor: Bet Faja Grau

OVERVIEW

JCM is a company dedicated to the design of access control systems. JCM has designed electrical equipment, which has lots of configurable parameters, so it can be used in many different installation. Almost all settings can be modified by customers. JCM also owns a telephone assistance service, which tries to solve customer's problems. However, they are often unable to support customer's doubts due to the lack of information received from the client.

The project aim is to solve the lack of information problem in order to improve the quality of the service offered by the telephone assistance service. The solution must not increase the production cost of the product. It should also be taken under consideration, that the JCM electric equipment can be installed anywhere worldwide. Therefore, we are forced only to use resources that are easily available to all customers.

To meet these goals, we will send a message, using the buzzer attached to the equipment. The message will contain the information about how the electric equipment it is configured. The message will travel through the telephone network to the telephone assistance service. Once there, the data will be decoded and send to a PC so, the telephone assistance service will get clear information about the equipment.

2 Introducció

2.1 Descripció del projecte

2.1.1 Presentació

JCM és una empresa dedicada al disseny de sistemes de control d'accés.

JCM disposa d'uns equips elèctrics que tenen la particularitat de tenir molts paràmetres configurables, per tant, es poden utilitzar en molts tipus d'instal·lació. Així s'aconsegueix reduir els costos de producció.

Al mateix temps, JCM disposa d'un servei d'atenció telefònica (SAT). El SAT gestiona i intenta donar solucions a tots els problemes que puguin sorgir als clients sobre el funcionament dels productes. La gran varietat de possibles configuracions dels equips, dificulta la feina del SAT. No poden donar suport als dubtes dels clients si l'única informació que reben és sobre el model d'equip elèctric servit. A més, s'ha de tenir en compte que els usuaris poden configurar els paràmetres segons les seves necessitats. Resoldre aquest problema és de vital importància per donar el suport adequat a tots els clients i millorar la qualitat del servei que ofereix el SAT.

La solució al problema, no ha d'alterar el cost de producció del producte, per tant, s'ha de trobar alguna solució que impliqui només un canvi de software, però en cap cas canvi de hardware. S'ha de tenir en compte també, que els equips poden estar instal·lats en qualsevol part del món i que només es poden utilitzar recursos de fàcil accés per tots els clients.

La solució passa per utilitzar els brunzidors, dels quals disposen els equips. El brunzidor s'utilitza actualment per indicar algunes de les funcions dels equips, per exemple, quan entra en mode programació.
El brunzidor emetria un senyal sonora codificada amb la informació sobre la configuració del equips. Aquest senyal sonora es recolliria a través del micròfon d'un telèfon i viatjaria a través de la xarxa telefònica fins arribar al departament del SAT a JCM. Un cop allà, la senyal sonora s'ha de descodificar i mostrar la informació d'una forma entenedora.
D'aquesta manera el SAT rebria totes les dades sobre la configuració de la instal·lació i així podria donar una resposta de manera més eficaç, ràpida i de qualitat.

2.1.2Objectius

Crear un software per codificar la informació sobre la configuració de l'equip en una senyal sonora.
Transmetre aquest senyal a través d'un canal d'àudio.
Crear un sistema que sigui capaç de rebre aquest missatge i descodificar el missatge de forma que sigui de fàcil comprensió pel SAT.
Aquets són els objectius del projecte.

En cada apartat es definiran els objectius parcials que corresponen a cada apartat. Els objectius com més petits, més fàcils d'aconseguir, per això s'ha intentat dividir el projecte amb petits objectius.

2.1.3Metodologia

El treball estarà estructurat amb les següents fases:
- Estudi sobre zumbadors, tipus de zumbadors disponibles a JCM i freqüències a les quals emeten.
- Estudi sobre la codificació adequada per poder-la transmetre pel canal d'àudio escollit.
- Crear un sistema perquè sigui capaç d'emetre un senyal codificada a través del zumbador.
- Crear un sistema que sigui capaç de rebre i descodificar la senyal.

2.1.4Organització de la memòria

Aquesta memòria està organitzada de la forma següent:

- **Introducció.** Introduir els punts bàsics del projecte.
- **Estudi de viabilitat.** Estudiar si la solució proposada és viable.
- **Emissor.** Desenvolupar un codi per aconseguir que l'emissor emeti un missatge.
- **Receptor (Hardware).** Desenvolupar un equip capaç de tractar la senyal.
- **Receptor (Software).** Desenvolupar un codi capaç de gestionar la informació.
- **Conclusions.**
- **Bibliografia.**

3 Estudi de viabilitat

3.1 Introducció

És possible transmetre informació d'un quadre? És possible fer-ho amb els mitjans i recursos disponibles? Com i amb que ho hem de fer?
Ens fem totes les preguntes per arribar a una conclusió sobre la viabilitat del projecte.

Abans de començar, hauríem de comprovar si es compleixen dos objectius essencials, per tal de veure, si es pot seguir desenvolupant el projecte:

1. Comprovar si podem generar una senyal amb un equip de JCM des d'un lloc remot i que la informació sigui capaç de viatjar i arribar de forma íntegre el SAT (servei d'atenció Telefònica).
2. Trobar un solució que no faci pujar el cost dels equips.

Algunes de les primeres solucions que ens venen el cap, passen per incloure mòduls GSM, Bluetooth, Wifi,... dins l'equip. Això implicaria pujar el cost dels equips, perquè actualment no hi ha cap equip que disposi d'aquestes parts de Hardware en la seva configuració de fàbrica. Per tant, tots aquests tipus de solucions, que impliquen afegir hardware, queden descartades.

Si fem una ullada els equips de JCM, podem observar que tots els equips disposen d'un brunzidor. Els brunzidors s'utilitzen per indicar alguns dels estats dels equips.
Per exemple: quan un equip fa un reset a la memòria, el brunzidor emet sons molt curts i molt seguits durant un segons.
Aquesta tipus de senyals sonores ajuden al usuari a interactuar amb l'equip.

A priori sembla una bona idea utilitzar aquest dispositius per generar la senyal. Decidim continuar desenvolupant el projecte en aquesta direcció. Com a mínim, sabem que compliríem amb una de les condicions. La solució no faria pujar el cost del producte.

3.2 El brunzidor

En aquest apartat, farem una breu descripció sobre què és i com funciona un brunzidor.

Un brunzidor, és un dispositiu electrònic que produeix un so o brunzit, continu o intermitent d'un mateix to. Serveix com a mecanisme de senyalització o avís, i són utilitzats en múltiples sistemes com en automòbils o en electrodomèstics.

La seva construcció consta de dos elements, un electroimant i una làmina metàl·lica d'acer. El brunzidor pot ser connectat a circuits integrats especials per així aconseguir diferents tons.
Quan s'acciona, el corrent passa per la bobina de l'electroimant i produeix un camp magnètic variable que fa vibrar la làmina d'acer sobre l'armadura.

Podem classificar els brunzidors en dos tipus: el que són autoscil·lants i els que no ho són.
Els autoscil·lants, no necessiten que la senyal d'alimentació oscil·li, per tant funcionen simplement si els connectem a una font d'alimentació. Ara bé, un dels inconvenients que ens podem trobar, és que tenen una resposta poc ràpida als canvis d'estat en l'alimentació i per tant, és més difícil de controlar quan s'engeguen i quan es paren
Els brunzidors que no són autoscil·lants necessiten, que la senyal que els alimenta, estigui oscil·lant a la freqüència que indica el fabricant. Aquest tipus en canvi, tenen una capacitat de reacció molt més ràpida.

Per el nostre projecte, hem de tenir en compte que els equips de JCM disposen de brunzidors no autoscil·lants.

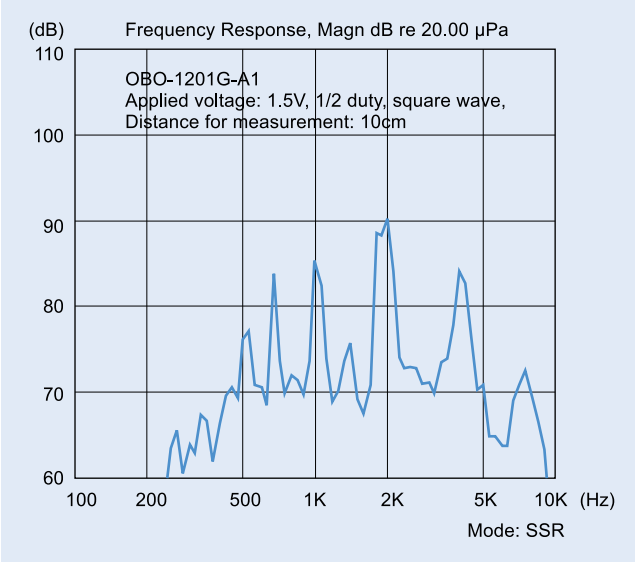
3.3 Brunzidors OBO-1201G-A1

En aquest apartat parlarem dels brunzidors OBO-1201G-A, que són els que disposen tots els equips de JCM.

Segons el Datasheet (s'inclou el datasheet l'annexa 1), que adjunta el fabricant, les característiques més importants són:
- Voltatge de funcionament entre 1 i 2V
- Consum màxim de 55mA

- Nivells de so mínim de 85dB
- Llindar de freqüències a la qual poden oscil·lar els brunzidors es troba entre 2000Hz i 150KHz

El Datasheet inclou una gràfica (consultar la imatge 1), on es pot veure la potència en decibels que emet el brunzidor, en funció de la freqüència d'oscil·lació que els alimenta. S'ha de tenir en compte que per realitzar aquesta gràfica, s'han aplicat 1.5V amb una senyal amb un cicle de treball del 50% i s'han fet les mesures a 10cm de distància



Imatge 1 Potència en funció de freqüència d'oscil·lació

3.4 Experiment: fer sonar un brunzidor

3.4.1 Introducció

En aquest apartat, portarem a terme un petit experiment perquè creiem que abans de continuar, és necessari tenir la certesa que entenem el funcionament dels brunzidors no autoscil·lants.

Un altre qüestió que ens preocupa és comprovar si els brunzidors, que han de ser els responsables de crear la senyal, sonen prou fort i tenen un so net per tal de ser els responsables de la creació del missatge.

Els objectius que ens plantegem en aquest apartat són:

- Comprovar que el fabricant ofereix un resultats fiables en el datasheet
- Descobrir a quina freqüència d'oscil·lació sonen més fort

3.4.2Metodologia

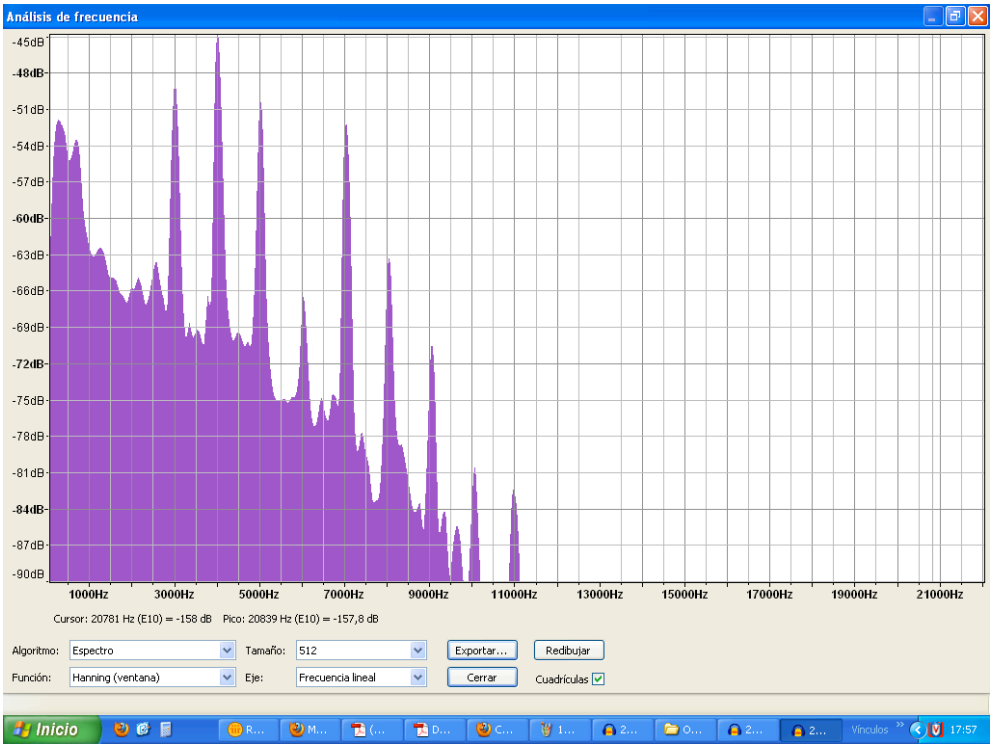
Connectem el brunzidor a un generador de funcions. Ajustem el valors del generador de funcions perquè siguin el més proper possible als valors de funcionament indicats en el datasheet.

Repetim l'experiment ajustant el generador de funcions en les freqüències d'oscil·lació que hi ha continuació:
- 1.000Hz
- 2.000Hz
- 5.000Hz
- 6.500Hz
-10.000Hz

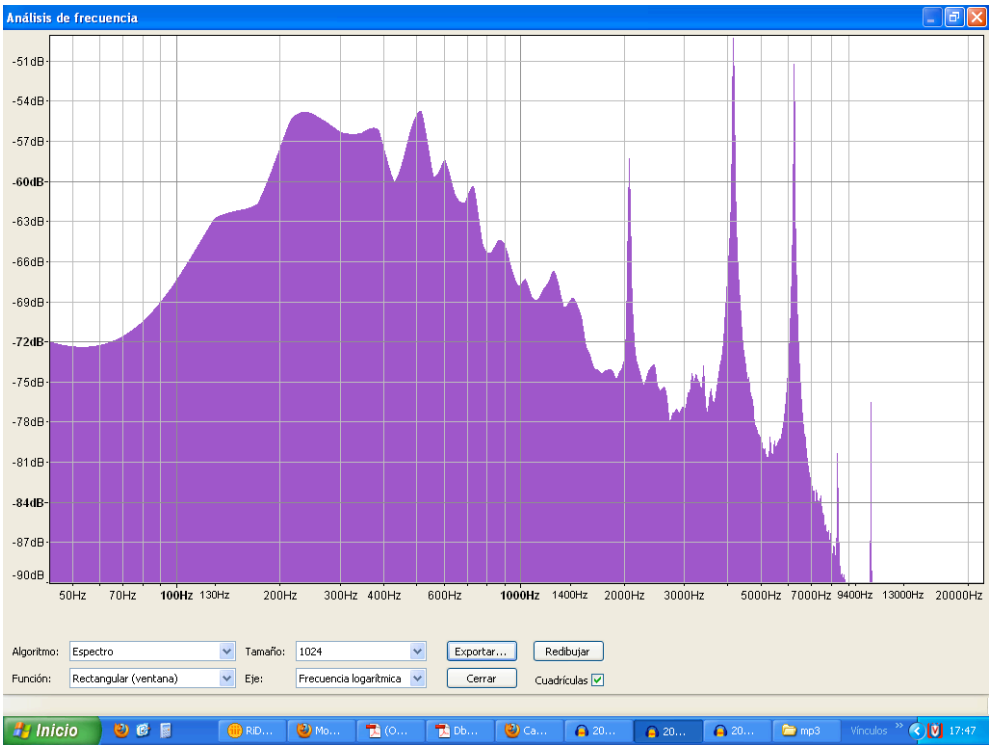
Enregistrem el so que emet el brunzidor en cada una de les freqüències en un laboratori on hi hagi el màxim silenci possible per obtenir resultats fiables.
Una vegada tenim enregistrats les gravacions les analitzem amb un programa informàtic (audacity) que ens mostra quin nivell de potència, en decibels, hem obtingut en funció de cada una de les freqüències.

3.4.3Resultats

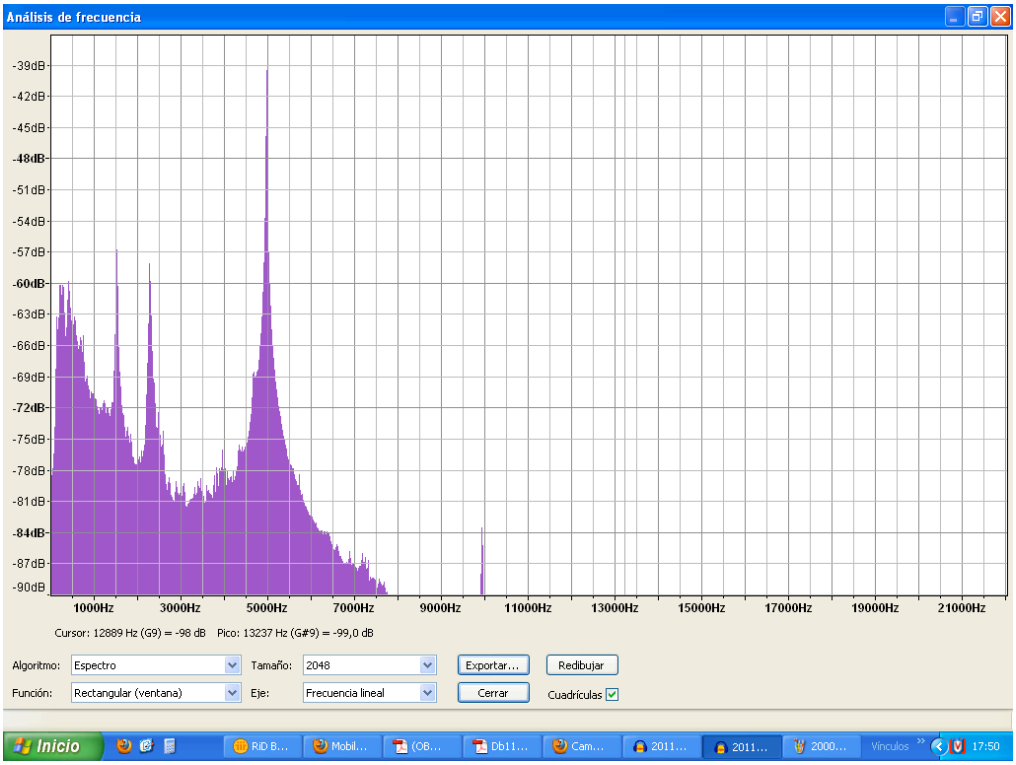
Es poden veure els resultats a 1.000Hz en la imatge 2, els de 2.000Hz en la imatge 3, els de 5.000Hz en la imatge 4, els de 6.500Hz en la imatge 5 i els de 10.000Hz en la imatge 6.



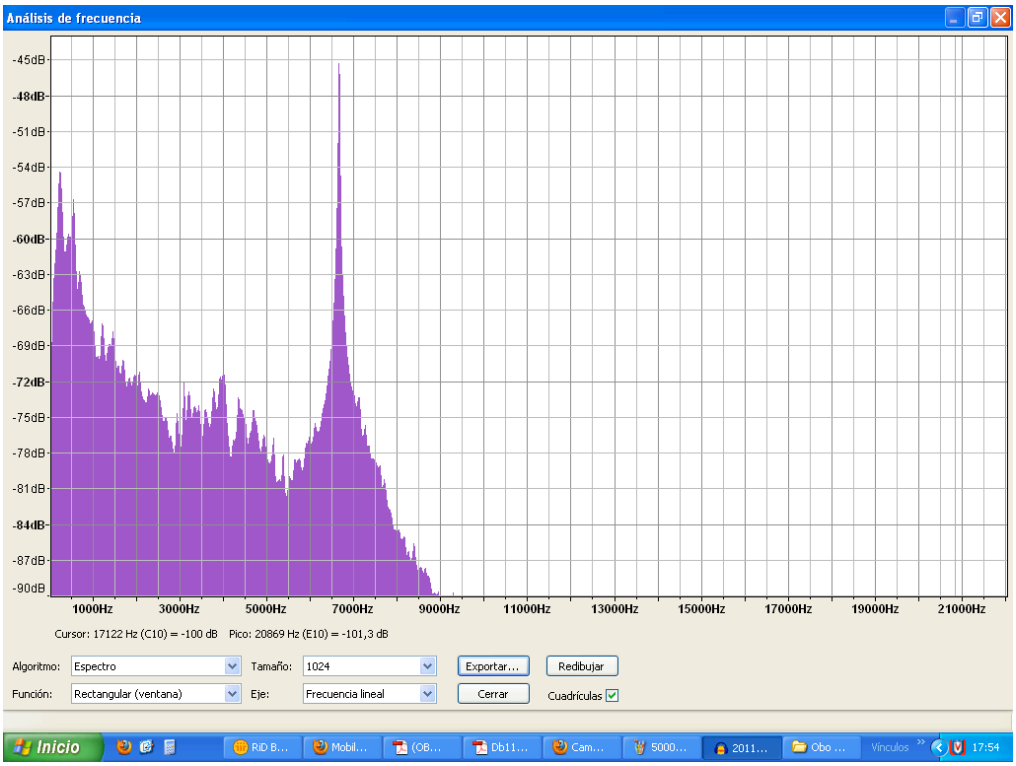
Imatge 2 Nivells de potència a 1.000Hz



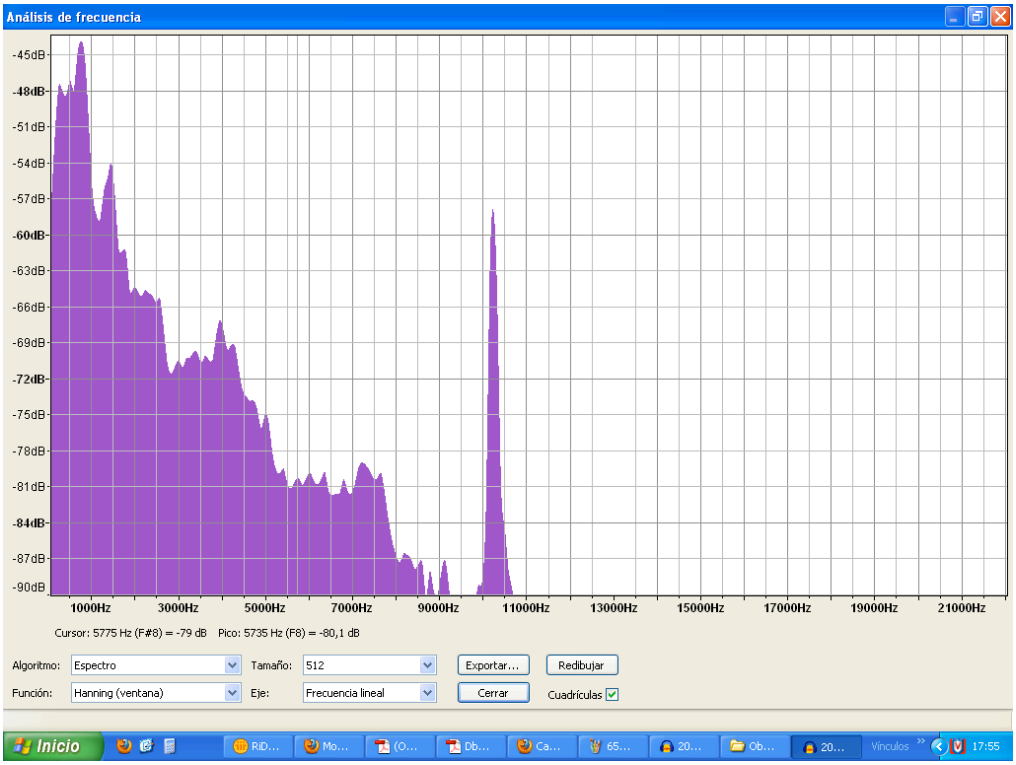
Imatge 3 Nivells de potència a 2.000Hz



Imatge 4 Nivells de potència a 5.000Hz



Imatge 5 Nivells de potència a 6.500Hz



Imatge 6 Nivells de potència a 10.000Hz

3.4.4Conclusions

Els resultats que hem obtingut, coincideixen amb els resultats del datasheet. Tot i així, tenen uns nivells més baixos que els indicats pel fabricant. Atribuïm aquesta diferència a que no disposem de cap sala insonoritzada per portar l'experiment a terme.
Per tant, podem concloure que el fabricant aporta unes especificacions correctes.

Amb aquest experiment també arribem a la conclusió, que si volem fer sonar l'oscil·lador a la màxima potència el farem oscil·lar a 1.000Hz, 2.000Hz o 6.500Hz. De les tres possibles freqüències, obtenim un so més potent quan l'alimentació oscil·la a 2000Hz.

3.5 La xarxa telefònica

3.5.1 Introducció

En aquest apartat, parlarem del canal que utilitzarem per enviar la nostra senyal: la xarxa telefònica.

L'últim pas per prendre la decisió sobre, si és viable o no el projecte, és comprovar que el so sigui capaç de viatjar del emissor fins al receptor a través de la xarxa telefònica

Gràcies a la recerca d'informació, sabem que la xarxa telefònica només és capaç de transmetre sons compresos entre els 300Hz i els 3.400Hz (tot i que la veu humana pugui arribar fins els 10.000Hz). El proper pas en el nostre estudi de viabilitat, és comprovar que realment podem passar un so emès pel brunzidor per la xarxa

3.6 Experiment: transmissió de so a través de la xarxa telefònica

3.6.1 Introducció

Volem comprovar que el so d'un brunzidor és capaç de viatjar per la xarxa telefònica

3.6.2 Objectius

- Comprovar que la xarxa telefònica és capaç de transmetre sons entre 300 i 3400Hz
- Decidir quina és la freqüència idònia per fer oscil·lar els brunzidors i transmetre el so per la línia telefònica

3.6.3 Metodologia

Connectem el brunzidor a un generador de funcions. Ajustem el valors del generador de funcions perquè siguin el més propers possibles als valors indicats al datasheet.

Comprovarem els decibels que genera el brunzidor quan ajustem la freqüència d'oscil·lació de la senyal en els valors els quals, en l'apartat anterior, hem comprovat que són els idonis:

- 1000Hz
- 2000Hz
- 6500Hz

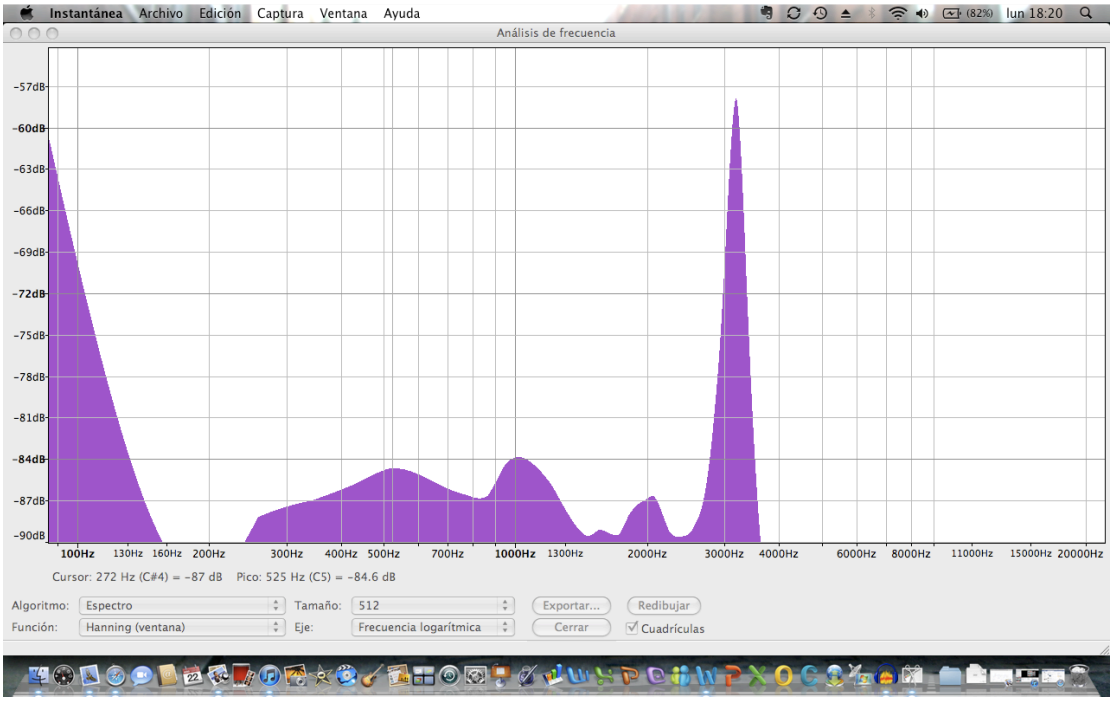
Fem un pont en el telèfon que utilitzarem per rebre la trucada, per tal que el so no passi cap l'altaveu del terminal sinó que passi per l'aparell que utilitzem per registrar les trucades (en aquest cas fem servir un iPhone 3Gs i l'aplicació "Notes de veu")

Realitzem una trucada des d'un terminal, utilitzant un terminal sense modificar. És important que el telèfon que emet la trucada i el que la rep, estiguin prou separats per tal que només pugui arribar el so per la xarxa i no directament per l'aire.

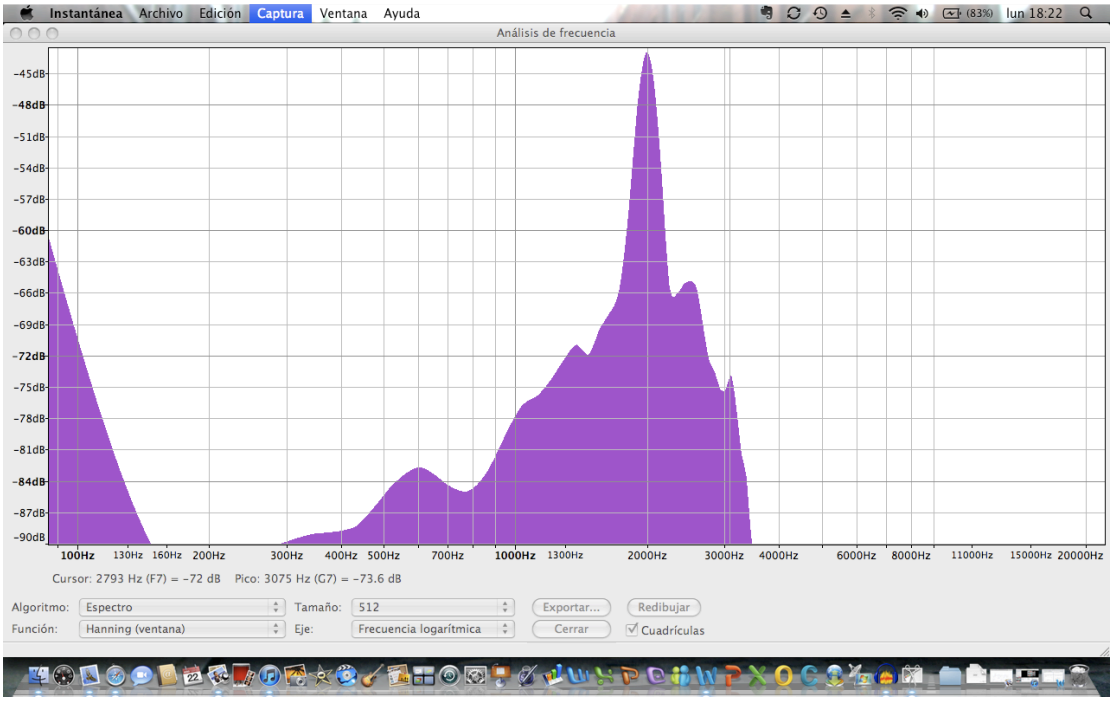
Una vegada tenim aquesta gravació, l'analitzem amb un programa informàtic (audacity). Ens mostra quin és el nivell de decibels que ha arribat després de la transmissió en funció de la freqüència.

3.6.4Resultats

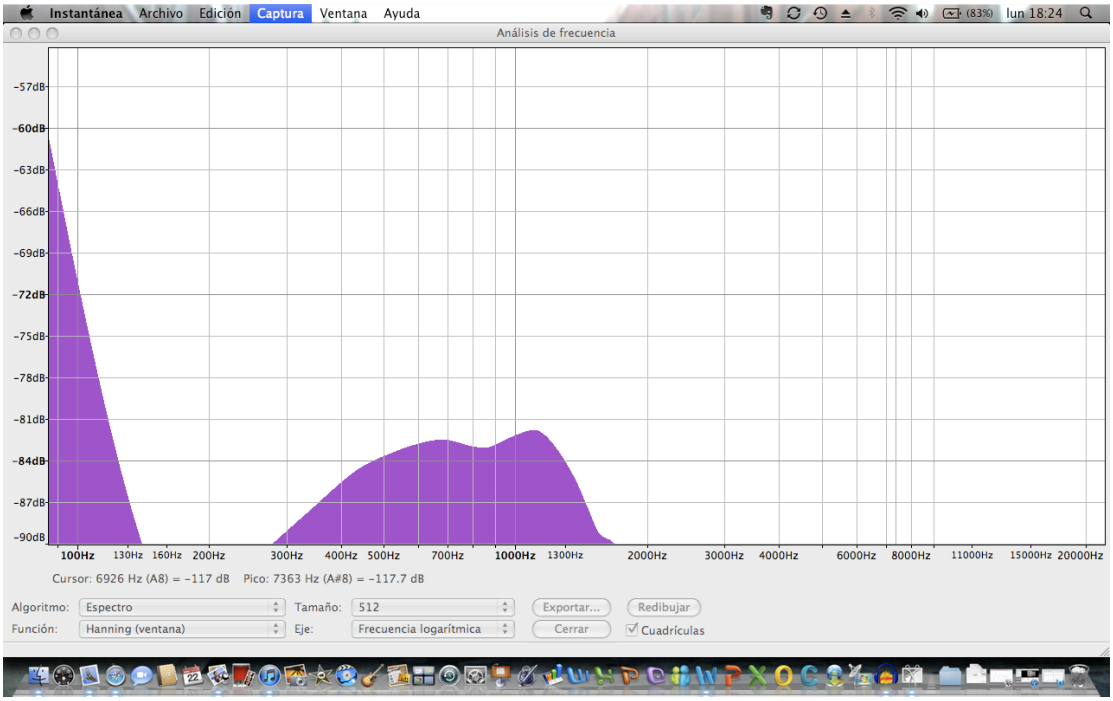
Podem observar els resultat dels nivells de senyal després de la transmissió en 1.000Hz en la imatge 7, de 2.000Hz en la imatge 8 i de 6.500Hz en la imatge 9.



Imatge 7 Nivells de potència a 1.000Hz



Imatge 8 Nivells de potència a 2.000Hz



Imatge 9 Nivells de potència a 6.500Hz

3.6.5Conclusions

Després del assaig, podem afirmar que, en la pràctica, es compleix la predicció teòrica. El telèfon, actua de filtre i només transmet els sons que es troben entre 300Hz i 3400Hz
Arribem a la conclusió que de les tres freqüències d'oscil·lació que feien sonar el brunzidor més fort (1000Hz, 2000Hz, 6500Hz), només aconseguim transmetre per la xarxa telefònica els sons de 1000Hz i 2000Hz.
De les dues freqüències restants, la més idònia és 2000Hz perquè ocuparem la meitat del ample de banda que si utilitzéssim 1.000Hz. Els harmònics de 2.000Hz queden filtrats pel propi telèfon i és la que aconsegueix crear un so més potent

3.7 Conclusions finals

Després de portar a terme l'estudi de viabilitat podem concloure que el projecte, és viable.

Els brunzidors són capaços d'emetre sons a diferents freqüències d'oscil·lació. La més adient a 2000Hz perquè és quan emet un so amb una potencia més alta

La xarxa telefònica, tot i tenir un ampla de banda molt petita, és capaç de transmetre sons a 2.000Hz.

Per tant, podem continuar desenvolupant el projecte, ja que hem pogut transmetre una senyal acústica creada per un brunzidor mitjançant una trucada telefònica.

4 Emissor

4.1 Introducció

Ara que sabem que el projecte és viable, comencem a desenvolupar tota la part que fa referència a la part emissora.

L'objectiu principal que hem d'assolir en aquest apartat, és aconseguir que un equip de JCM sigui capaç d'emetre un missatge a través del brunzidor i que el missatge porti la informació sobre la configuració del equip. És important que organitzem la informació d'una manera prou clara perquè el receptor no tingui problemes per descodificar-la i tractar-la.

Per aconseguir arribar a complir aquest objectiu principal, ens hem marcat uns objectius parcials més petits. Si aconseguim complir els objectius parcials respectivament, arribarem al objectiu final.

El camí d'objectius parcials que ens hem marcat és:
- Fer sonar el brunzidor de manera constant
- Emetre el missatge "HELLO WORLD" a través del brunzidor
- Discutir les possibles codificacions de la senyal i escollir-ne una
- Discutir sobre la organització de la informació
- Emetre un missatge amb el brunzidor amb una configuració d'equip exemple

4.1.1 Material de desenvolupament

En aquest apartat, farem una petita menció sobre el material que utilitzarem per desenvolupar el projecte.

Desenvoluparem el projecte sobre d'una placa d'avaluació de la marca Microchip, la placa "44-Pin Demo board".
Aquesta placa, funciona amb el PIC18F45K20. En l'annex 2 s'inclou el datasheet amb tota la informació detallada sobre el PIC i en l'annex 3 la informació sobre placa Demo.
Hem escollit utilitzar aquesta Demo board pel microcontrolador que utilitza, el PIC18F45K20. Aquest PIC és el que utilitzen alguns equips de JCM. La Demo board ens facilita la feina de programar perquè podem treballar en un entorn similar al dels equips, però sense haver de perdre temps amb les configuracions específiques de cada equip. Ens resulta més simple desenvolupar la nostra idea en un espai més abstracte allunyat del producte. A més,

d'aquesta manera implementar el projecte a qualsevol producte hauria de ser més fàcil. També ens dona un altre avantatge, ja que tenim més flexibilitat alhora de fer proves amb el nostre codi.
La placa disposa de petits dispositius de hardware, com per exemple: LEDs, interruptor, etc. Podem utilitzar aquets dispositius per desenvolupar la nostra aplicació, per exemple, ajudar-nos en la detecció d'errors.

Per gravar el codi dins la "44-Pin Demo board", utilitzem el Pickit 3. El Pickit 3 és una eina de Microchip que es connecta mitjançant uns pins a la Demo board i a través d'un cable USB, es connecta a l'ordinador. D'aquesta manera, podem gravar el programa de l'ordinador al PIC. Aquesta eina també ens permet utilitzar el "debugger" en l'editor de codi, per exemple si volem detectar errors.
Podem veure la placa demo i el gravador en la imatge 10.

Imatge 10 Demo Board "44-pin Demo board" i gravador Pickit 3

Desenvoluparem el software del nostre projecte utilitzant llenguatge C.
Per editar el codi utilitzarem l'editor MPLAB. MPLAB, és un programa gratuït de la marca Microchip. Aquest editor ens permet seleccionar i desenvolupar en tots els diferents microcontroladors de Microchip, a més a més, ens ajuda a gravar el codi dins dels microcontroladors amb l'ajuda del gravador (en el nostre cas el Pickit 3). MPLAB disposa d'eines per començar un projecte desde 0, editar text, compilar, simular el codi, etc.

4.2 Brunzidor a la Demo board

4.2.1 Introducció

Intentarem complir el primer dels objectius parcials, fer sonar un brunzidor sobre la placa d'avaluació o Demo board.
Per arribar a complir l'objectiu hem dividit aquest apartat en dues parts. En la primera part parlarem del hardware i en la segona part parlarem del software

4.2.2 Hardware

La placa d'avaluació de Microchip no disposa de brunzidor, per tant, com és obvi, no el podrem fer sonar sinó en posem cap connectat a la Demo board.
En aquest apartat es fa un breu descripció sobre tot el procés que s'ha seguit per decidir on col·locar el brunzidor.

Com hem vist en l'experiment anterior, per fer sonar el brunzidor només ens fa falta connectar el brunzidor a una font d'alimentació. La font d'alimentació haurà de tenir una senyal d'oscil·lació de 2KHz.
Per tant, hem de connectar el pin negatiu del brunzidor en un port negatiu de la placa d'avaluació, que està marcat com a "GND". El pin positiu del brunzidor, l'hem de connectar a una sortida del microcontrolador.

Per decidir quins dels 44 ports del microcontrolador utilitzarem com a sortida, consultem el datasheet del PIC (imatge 11). Hem de tenir en compte la distribució dels ports en la placa. Creiem que seria aconsellable trobar un port de sortida adient, prop d'un port GND així no caldrà calbejar el brunzidor perquè funcioni i podrem connectar els pins del brunzidor directament a la Demo board.

TABLE 1-3: PIC18F4XK20 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTD is a bidirectional I/O port or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
RD0/PSP0	19	38	38			
RD0				I/O	ST	Digital I/O
PSP0				I/O	TTL	Parallel Slave Port data
RD1/PSP1	20	39	39			
RD1				I/O	ST	Digital I/O
PSP1				I/O	TTL	Parallel Slave Port data
RD2/PSP2	21	40	40			
RD2				I/O	ST	Digital I/O
PSP2				I/O	TTL	Parallel Slave Port data
RD3/PSP3	22	41	41			
RD3				I/O	ST	Digital I/O
PSP3				I/O	TTL	Parallel Slave Port data
RD4/PSP4	27	2	2			
RD4				I/O	ST	Digital I/O
PSP4				I/O	TTL	Parallel Slave Port data
RD5/PSP5/P1B	28	3	3			
RD5				I/O	ST	Digital I/O
PSP5				I/O	TTL	Parallel Slave Port data
P1B				O	—	Enhanced CCP1 output
RD6/PSP6/P1C	29	4	4			
RD6				I/O	ST	Digital I/O
PSP6				I/O	TTL	Parallel Slave Port data
P1C				O	—	Enhanced CCP1 output
RD7/PSP7/P1D	30	5	5			
RD7				I/O	ST	Digital I/O
PSP7				I/O	TTL	Parallel Slave Port data
P1D				O	—	Enhanced CCP1 output

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

Note 1: Default assignment for CCP2 when Configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when Configuration bit CCP2MX is cleared.

Imatge 11 PIC 18 Descripció entrades i sortides

Després de consultar el datasheet, observem que el port RD5 (pin 28), satisfà les condicions que creiem necessàries. És un port que es pot configurar com una sortida digital i a més es troba prop d'un port GND.
Amb tot això, decidim connectar un brunzidor entre aquets dos ports

Per fer sonar el brunzidor només falta connectar la placa a una font d'alimentació i fer que la senyal que surti del port RD5 sigui oscil·lant.

4.2.3Software

En aquest apartat, es discuteix el desenvolupament de totes les funcions necessàries per tal d'aconseguir que pel port RD5 surti una senyal d'oscil·lació que faci vibrar el brunzidor a la freqüència de 2KHz

Per tal d'ordenar les idees hem creat la figura 1

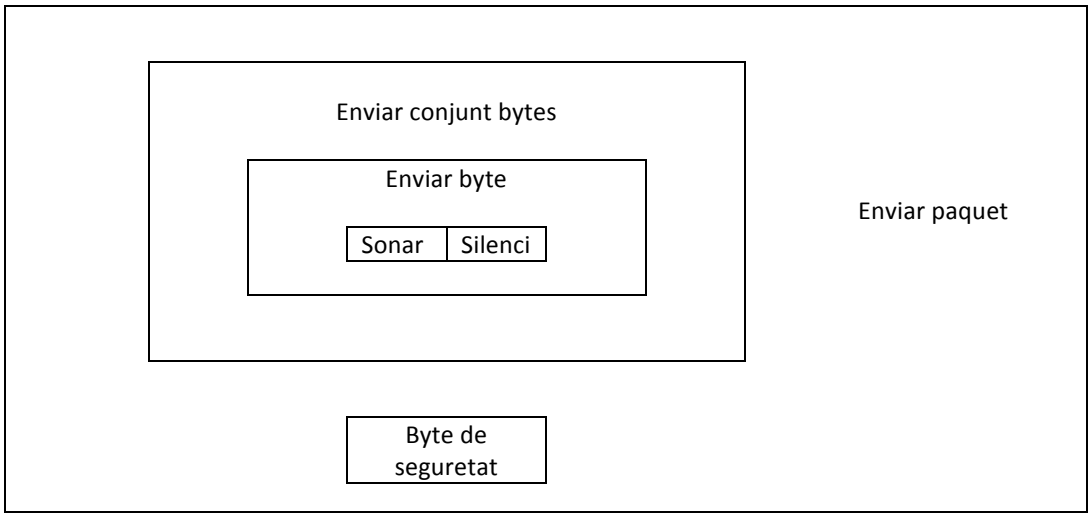


Figura 1 Organització funcions emissió

4.2.3.1Configuració del microcontrolador

Abans de començar a editar el software, cal que configurem alguns dels paràmetres del PIC En aquest apartat, descriurem com configurem, amb l'ajuda del datasheet, el microcontrolador per aconseguir que arribi al brunzidor una senyal oscil·lant a 2000hz i com a conseqüència emeti un so constant. Les parts del PIC que creiem necessari configurar són:

- El port RD4 perquè actuí com un port de sortida
- La freqüència d'oscil·lació del microcontrolador
- El Timer0 del microcontrolador

PORT RD4

Consultem el datasheet, imatge 12, per veure com es configuren els ports D

10.5 PORTD, TRISD and LATD Registers

Note: PORTD is only available on 40/44-pin devices.

PORTD is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISD. Setting a TRISD bit (= 1) will make the corresponding PORTD pin an input (i.e., disable the output driver). Clearing a TRISD bit (= 0) will make the corresponding PORTD pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin).

The Data Latch register (LATD) is also memory mapped. Read-modify-write operations on the LATD register read and write the latched output value for PORTD.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Three of the PORTD pins are multiplexed with outputs P1B, P1C and P1D of the enhanced CCP module. The operation of these additional PWM output pins is covered in greater detail in **Section 16.0 "Enhanced Capture/Compare/PWM (ECCP) Module"**.

Note: On a Power-on Reset, these pins are configured as digital inputs.

PORTD can also be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. See **Section 10.9 "Parallel Slave Port"** for additional information on the Parallel Slave Port (PSP).

Note: When the enhanced PWM mode is used with either dual or quad outputs, the PSP functions of PORTD are automatically disabled.

EXAMPLE 10-4: INITIALIZING PORTD

```
CLRF    PORTD   ; Initialize PORTD by
                ; clearing output
                ; data latches
CLRF    LATD    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISD   ; Set RD<3:0> as inputs
                ; RD<5:4> as outputs
                ; RD<7:6> as inputs
```

Imatge 12 PIC 18 Descripció PORTD

A partir d'aquest fragment, arribem a la conclusió que per configurar el port RD5 com una sortida, cal escriure aquesta sentència

```
TRISD = 0b00000000;
```

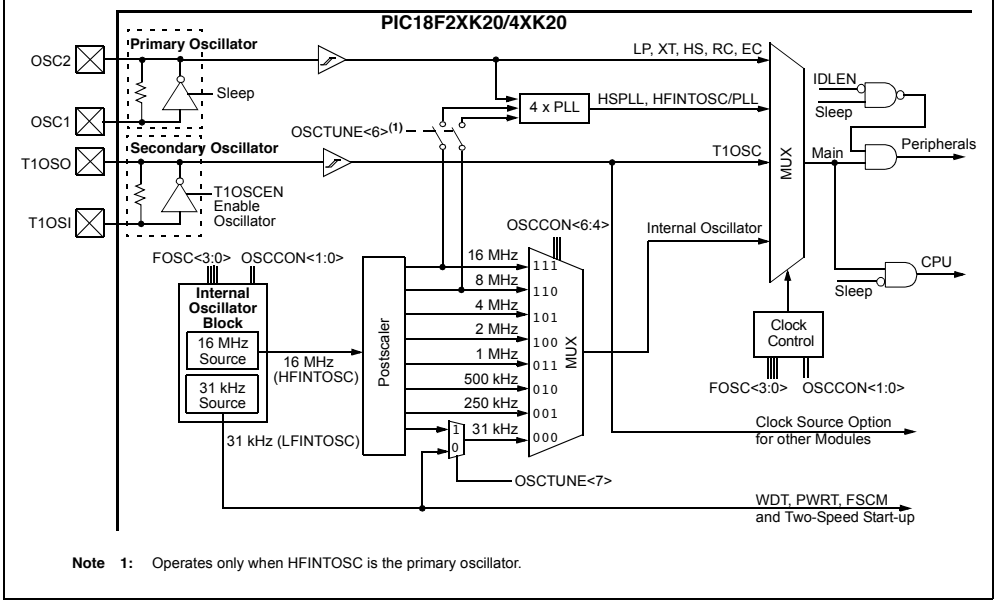
El registre TRISD és el que controla els ports D. Controla si actuen com una entrada o com una sortida. Cada bit del registre TRISD representa un port D del microcontrolador. Per exemple, el bit 5 representa el port RD5. Si el valor de bit5 és 1, el microcontrolador actuarà com una entrada. Si el valor és 0, el port actuarà com una sortida.

Amb la nostra sentencia acabem de configurar el port RD5 (i la resta de ports D) com una sortida.

FREQÜÈNCIA D'OSCIL·LACIÓ

Per poder generar una senyal de 2KHz abans, ens hem d'assegurar quina és la freqüència del microcontrolador. Per això, tornem a consultar el datasheet (imatge 13 i imatge 14).

FIGURE 2-1: PIC® MCU CLOCK SOURCE BLOCK DIAGRAM



Imatge 13 PIC 18 Descripció clock intern

REGISTER 2-1: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-0	R/W-1	R/W-1	R-q	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS(1)	IOFS	SCS1	SCS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	q = depends on condition
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **IDLEN:** Idle Enable bit
1 = Device enters Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF<2:0>:** Internal Oscillator Frequency Select bits
111 = 16 MHz (HFINTOSC drives clock directly)
110 = 8 MHz
101 = 4 MHz
100 = 2 MHz
011 = 1 MHz(3)
010 = 500 kHz
001 = 250 kHz
000 = 31 kHz (from either HFINTOSC/512 or LFINTOSC directly)(2)

bit 3 **OSTS:** Oscillator Start-up Time-out Status bit(1)
1 = Device is running from the clock defined by FOSC<2:0> of the CONFIG1 register
0 = Device is running from the internal oscillator (HFINTOSC or LFINTOSC)

bit 2 **IOFS:** HFINTOSC Frequency Stable bit
1 = HFINTOSC frequency is stable
0 = HFINTOSC frequency is not stable

bit 1-0 **SCS<1:0>:** System Clock Select bits
1x = Internal oscillator block
01 = Secondary (Timer1) oscillator
00 = Primary clock (determined by CONFIG1H[FOSC<3:0>]).

Note 1: Reset state depends on state of the IESO Configuration bit.
2: Source selected by the INTSRC bit of the OSCTUNE register, see text.
3: Default output frequency of HFINTOSC on Reset.

Imatge 14 PIC 18 Descripció registre del oscil·lador

Si tenim en compte la informació del datasheet, utilitzarem la següent sentència per definir el temps d'oscil·lació del nostre microcontrolador

```
OSCCON=0b01010000
```

Aquesta sentència posa el valor binari 01010000 al registre OSCCON, que és l'encarregat de controlar el temps d'oscil·lació. Si no modifiquéssim aquets registre, aquest prendria el valor de 0 en tots els seus bits.

Forçant un '101' en els bits 6-4, aconseguim que la freqüència d'oscil·lació interna del microcontrolador sigui de 4Mhz.

TIMER 0

L'últim paràmetre que configurarem abans de començar a escriure la nostre funció, és el Timer0. Consultem el datasheet per saber quin registre hem de configurar per poder-lo controlar (imatge 15).

12.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 12-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 12-1. Figure 12-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

REGISTER 12-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

Imatge 15 PIC 18 Descripció registre del Timer0

D'aquest fragment, podem extreure que el registre T0CON és l'encarregat de configurar el Timer0. Escrivim la següent sentència

```
TOCON = 0b01001000
```

Utilitzem el Timer0 per generar la senyal que oscil·la a 2KHz

El Timer0, pot estar configurat com un comptador de 8 o de 16 bits. Nosaltres hem forçat el bit6 amb el valor 1, per tan actuarà com un comptador de 8 bits. Això vol dir, que el Timer0 farà increments de 1 cada cicle fins arribar als 255 increments (2 (codi binari)^8 bits=255 increments). Activarà un bit de flag cada vegada que hagi fet els 255 increments.

4.2.3.2 void sonar ( )

Amb aquesta funció, el que volem aconseguir és que la senyal de RD5 canvi d'estat (de 1 a 0) periòdicament per tal d'aconseguir una senyal de 2KHz . D'aquesta manera el brunzidor sonarà d'una manera constant.

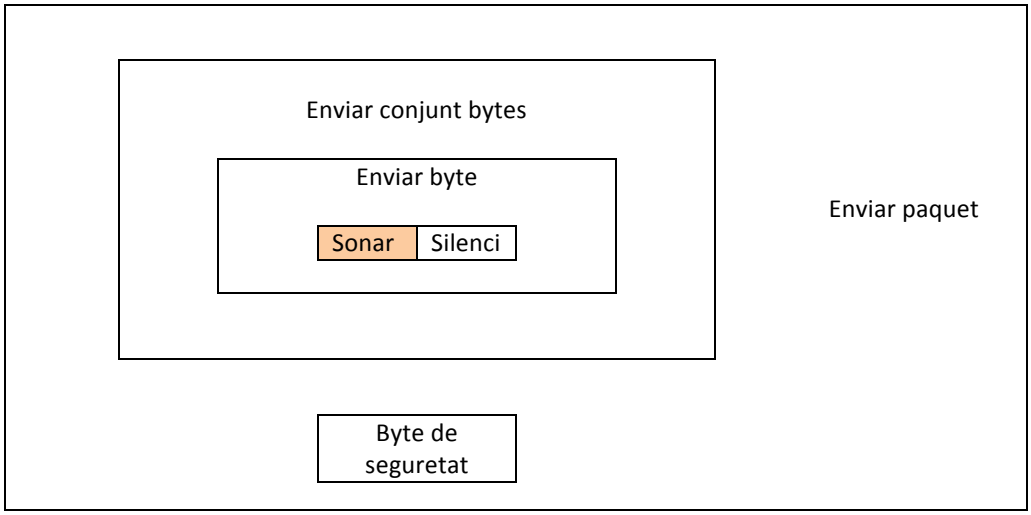


Figura 2 Organització funcions emissió. Desenvolupant sonar

Primer, la funció comprova si el flag que indica que el comptador ha arribat el final, està activat. Per fer-ho, comprova si el valor que hi ha a "INTCONbits.TMR0IF" és 1.
Si no ha arribat el final, el flag és 0 i la funció no fa res
En canvi si el flag és 1, vol dir que ha arribat el final del comptador. Per tant, tornem a posar al flag 0, per poder esperar a la següent activació.
Tot seguit comprovem si l'últim senyal que s'ha enviat al port RD5 (en la nostra funció el port RD5 pren el nom de "a") era un 0, si és així enviem un 1 pel port RD5. Si l'últim senyal que s'ha enviat era un 1, enviem un 0.

El nostre comptador de 8 bits fa els increments a una velocitat de 4MHz. Cada 255 increments enviem un senyal al port RD5 diferent. Aquest canvi d'estat en el port RD5 cada 255 increments, és el que fa que la senyal de la sortida del port oscil·li a 2.000Hz.

A continuació, es detallen els càlculs que ens han ajudat a prendre la decisió per utilitzar un comptador de 8 bits (255 increments) i configurar el microcontrolador a una freqüència de treball de 4MHz

4.2.4 Conclusions

En aquest punt, disposem del hardware i del software que creiem que ens ajudaran a complir el nostre objectiu.
Per comprovar que hem aconseguit l'objectiu, col·loquem la funció *void sonar ( )* dins d'un bucle infinit. Col·loquem el brunzidor sobre la Demo board, tal i com hem descrit en l'apartat de hardware
Alimentem la Demo board utilitzant el port USB i el Pickit 3. Efectivament obtenim un so constant a 2.000Hz.
Això doncs s'afirma que hem aconseguit el primer dels nostres objectius parcials d'aquesta part del projecte.

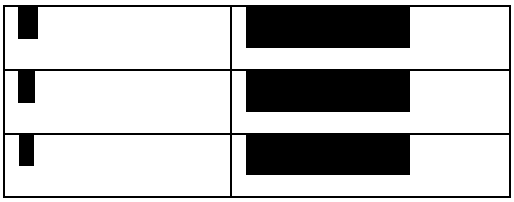
4.3 Hello world

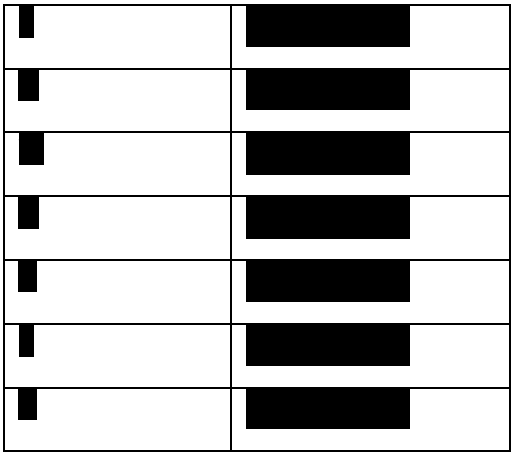
4.3.1 Introducció

Som capaços de fer sonar el brunzidor de manera constant. En aquest apartat, es descriu tot el procés que hem seguit per intentat complir el següent objectiu.
Volem transmetre un missatge senzill a través del brunzidor. Quan es comença l'estudi d'un nou llenguatge de programació el primer missatge que és transmet, és la frase "HELLO WORLD".
Per tant, l'objectiu d'aquest apartat és emetre "HELLO WORLD" a través del brunzidor

Els brunzidors que utilitzem per aquest projecte només tenen dos estats: o emeten so, o no emeten so. Són un sistema binari. El brunzidor pot estar parat (en estat 0) o pot estar funcionant (en estat 1)

Necessitem convertir la frase "HELLO WORLD" en un missatge que tingui una codificació binària.
Per convertir la frase "HELLO WORLD" a una codificació binària, utilitzem una taula de conversió ASCII (s'inclou una taula de conversió completa en l'annex 4)





Taula1 Hello world a codificació binària

En la taula 1 veiem com queda el missatge "HELLO WORLD" amb codificació ASCII

Ara, doncs, sabem quina informació hem d'enviar a través dels brunzidors. Només queda descriure totes les funcions necessàries per poder emetre aquest missatge

4.3.2void SonarTemps (int x)

Com hem vist en l'apartat d'introducció, per poder emetre el missatge "HELLO WORLD" a través del brunzidor, necessitem emetre dues classes de bits, els bits que equivalen al estat 1, farem sonar el brunzidor, i els bits que equivalen al estat 0, farem que brunzidor estigui en silenci.

Actualment, en la configuració dels equips de JCM, el so més ràpid que hi ha, és de 40ms. Necessitem un temps de bit prou llarg perquè tot i que tinguem una interferència en algun instant, el bit no quedi malmès del tot. Amb una durada de bit de 40ms a priori no sembla que tinguem aquest problema. Calculem quants bytes podem enviar amb 60s amb aquest temps de bit.

187 bytes per minut ens sembla un temps acceptable, es per això, que decidim que el temps d'emissió de cada bit sigui de 40ms. El primer que hem de fer, és modificar la funció del apartat anterior que emetia un so constant i infinit. L'hem de modificar perquè emeti un so constant però només durant 40ms.

Aquesta funció té un paràmetre d'entrada

- **int x:** Indica quin serà el temps de bit amb mil·lisegons

Aquesta funció, agafa el temps de bit int x que nosaltres hem escollit i calcula quantes activacions de flag de Timer0 hi haurà durant aquest temps de bit escollit. El número d'activacions es guarda en la variable temps.

Mentre que el número d'activacions no superi la variable temps, aquesta funció farà sonar el brunzidor utilitzant la mateixa tècnica que en la funció *void sonar ( )*.

Un cop el número d'activacions sobrepassi el límit que acabem de calcular, el brunzidor deixarà de sonar

4.3.3 void SilenciTemps (int x)

Com s'ha descrit anteriorment, el nostre missatge és binari. En aquest moment, som capaços d'enviar bits de valor 1. La missió de la funció que es descriu en aquest apartat, és enviar bits de valor 0.

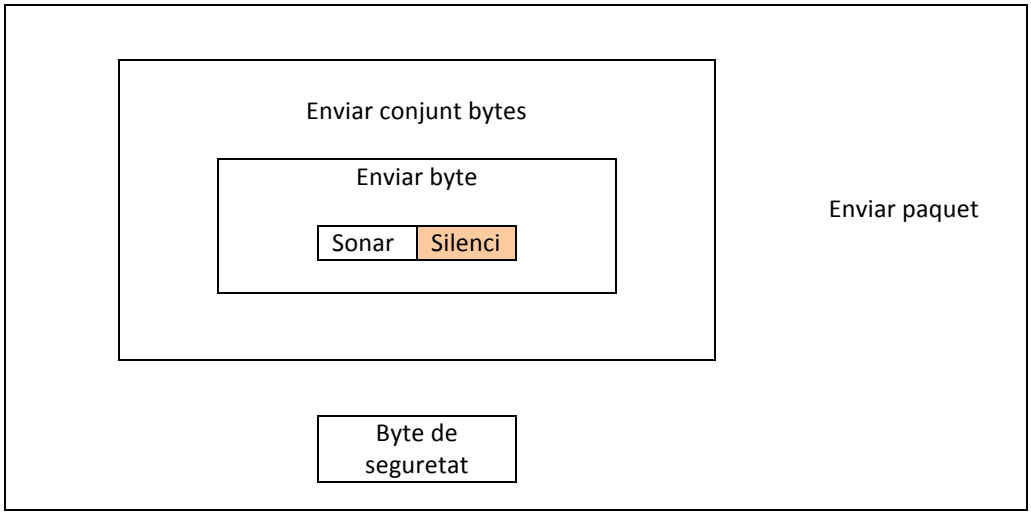


Figura 3 Organització funcions emissió. Desenvolupant silenci

Aquesta funció té un paràmetre d'entrada

- **int x:** Indica quin serà el temps per cada bit amb mil·lisegons

Aquesta funció és molt semblant a la funció *voidSonarTemps( )*. Cal que li passem el temps de bit en mil·lisegons. Després, la funció calcula quantes vegades s'activarà el flag del Timer0 abans no finalitzi el temps de bit.

Com que l'objectiu en aquesta funció és, que el brunzidor no funcioni durant el temps de bit, només hem d'esperar que passi el temps desitjat sense fer res. Aquesta funció, espera el número d'activacions que hem calculat i un cop ha passat, s'acaba la rutina.

4.3.4void EnviaByte (char a)

La funció d'aquest apartat té l'objectiu d'emetre una lletra sencera del nostre missatge. Cada una de les lletres del missatge, està formada per 8 bits (8 bits=1byte). Combinarem les funcions que hem desenvolupat en apartats anteriors. Crearem una funcio nova que sigui capaç d'enviar un byte sencer.

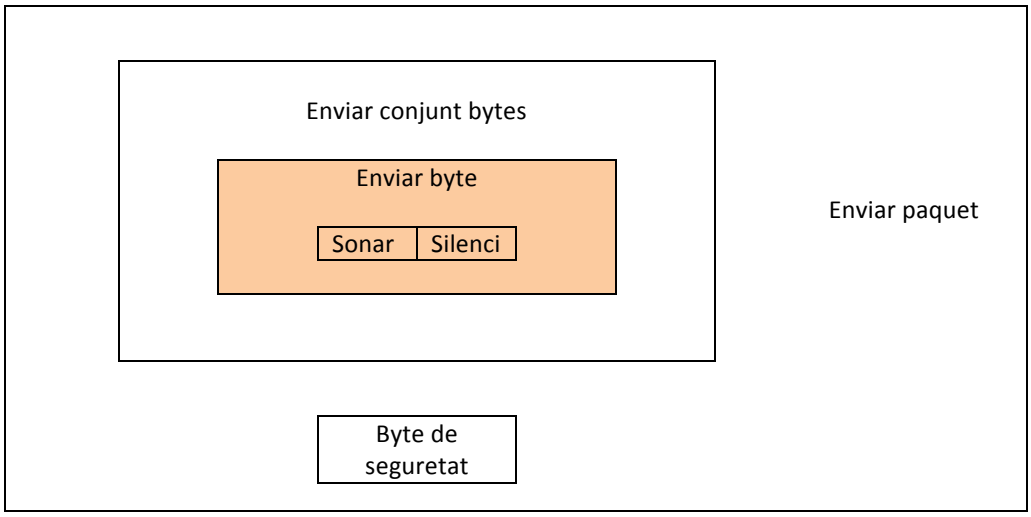


Figura 4 Organització funcions emissió. Desenvolupant enviar byte

Aquesta funció té un paràmetre d'entrada

- **char a:** indica el byte (nosaltres utilitzarem una lletra, char) que volem que s'emeti

Primer, comencem a analitzar el bit8 del byte, el més cap a l'esquerra del nostre paràmetre d'entrada. Comprovarem si el bit8 és un 1 o un 0.

Per decidir quin és el valor del bit8, fem una operació AND lògica bit a bit entre el valor binari 1000000 amb el byte a transmetre.
Si el resultat d'aquesta AND lògica és 10000000, vol dir que el bit8 és un 1. Si el resultat és 00000000 vol dir que el bit8 és un 0.
Es descriu un exemple perquè sigui més entenedor

byte a transmetre --> 'H' = 0100 1000
byte de comparació --> 1000 0000
Apliquem l'operació lògica AND bit a bit
Resultat--> 0000 0000
Per tant, arribem a la conclusió que el bit8 del byte a transmetre és 0

Un cop hem acabat aquesta comparació, emetem el resultat. Després desplacem la posició dels bits del byte a emetre un bit cap a l'esquerra. Per exemple, el bit2 passa a ser el bit3
Apliquem un bucle per repetir aquest procés 8 vegades, així ens assegurem que transmetem els 8 bits del byte

4.3.5void SendBuffer (char buffer[ ], int numchar)

En aquest apartat, fem una descripció sobre l'última funció que ens falta per aconseguir el nostre objectiu: enviar la frase "HELLO WORLD" a través del brunzidor.
Gràcies a la funció que hem desenvolupat, som capaços d'enviar tot un byte (char).
L'objectiu d'aquesta funció és enviar una frase sencera. Per tant, enviar més d'un byte

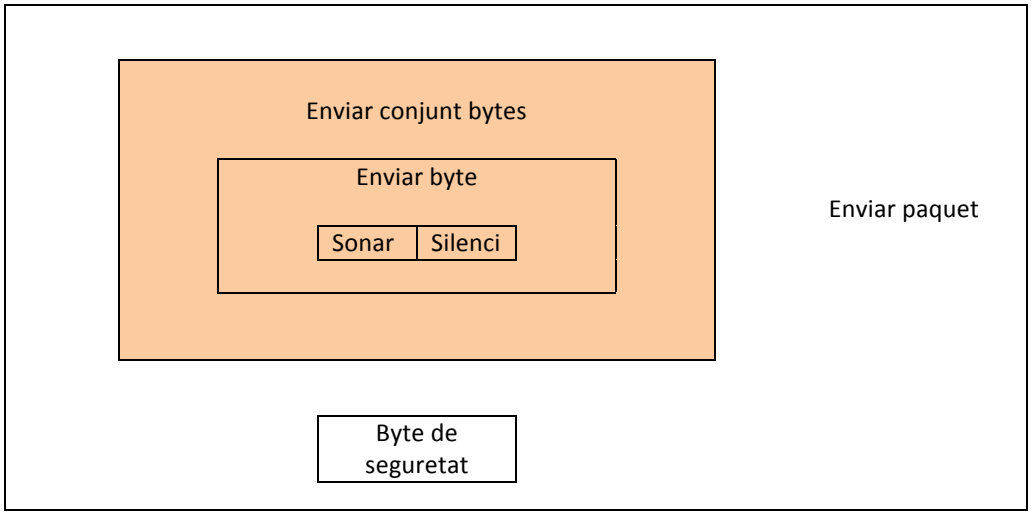


Figura 5 Organització funcions emissió. Desenvolupant enviar conjunt bytes

```
void SendBuffer (char buffer[], int numchar)

{
        int i;
        for (i=0; i<numchar;i++)
        {
                EnviaByte (buffer [i]);
        }
}
```

En aquesta funció se li han de passar dos paràmetres

- Char buffer [ ]: indica quins bytes, en forma de matriu, volem que transmeti

- int numchar: Indica quina és la grandària de la matriu buffer [ ]

Aquesta funció té un funcionament molt simple. És un bucle, que crida la funció *void EnviaByte (char a)*, tantes vegades com indica *numchar*. Cada vegada que crida *void EnviaByte (char a),* el brunzidor emet un byte (char) diferent. La funció acaba quan s'han emés tots els caràcters que hi ha a buffer.

4.3.6Conclusions

Afegim les tres funcions que s'han descrit en aquest apartat dins del nostre codi. Per comprovar si hem complert amb el nostre objectiu, posem dins d'un bucle infinit la funció *void SendBuffer (char buffer[], int numchar).*

Quan alimentem la Demo board amb totes aquestes noves funcions gravades dins del PIC, la placa d'avaluació emet de manera infinita la frase "HELLO WORLD".

Per tant, donem per complert l'objectiu d'aquest apartat

4.4 Codificació de la senyal

4.4.1 Introducció

Discutirem sobre les possibles codificacions que podem utilitzar en el projecte PITOS.
Fins ara, hem desenvolupat el projecte sense tenir em compte la part receptora. Som capaços d'enviar qualsevol missatge fins a l'aparell receptor. En aquest punt creiem necessari començar a pensar en la part receptora. Hem de triar una codificació perquè la informació sigui el més fàcil possible de descodificar i interpretar per la part receptora.
Farem un petit estudi sobre les codificacions més importants. El final prendrem una decisió sobre quina codificació utilitzarem per continuar amb el nostre projecte

4.4.2 Codificació digital unipolar, codificació digital polar i codificació digital bipolar

Analitzarem aquestes 3 famílies de codificació, per després escollir quina de les tres s'adequa més a les nostres necessitats.

La codificació unipolar, és una codificació tipus binari. Només s'assigna polaritat quan el bit té el valor de 1. Normalment s'assigna un voltatge positiu per indicar el 1 binari . Si es desitja, es pot assignar la polaritat negativa per indicar el bit de valor 1. És la codificació més simple i codifica els bits directament com arriben.
Els problema d'aquest tipus de codificació, és que no disposa de sincronisme.

La codificació digital polar, és un tipus de codificació digital de la informació. S'assigna els bits de valor 1 a una polaritat i els bits de valor 0 s'assignen a una altre polaritat (Tot i que de forma natural, aquest tipus de codificacions no tenen cap tipus de resistència a la pèrdua d'informació ni tampoc sincronisme, es poden aplicar tècniques per aconseguir les dues característiques).
Les llargues cadenes de bits continuen essent un problema perquè no causaran cap transició en la senyal i en podem perdre els sincronisme

La codificació digital bipolar, és un tipus de codificació digital de la informació. Pot oposar resistència a determinades formes de pèrdua d'informació durant la transmissió. En aquest tipus de codificació s'utilitzen tres nivells diferents de polarització. El nivell de 0 volts, s'utilitza per representar els bits de valor 0. Els bits 1, es codifiquen com a valors positiu o

negatius de forma alterna. Si el primer bit valor 1 es codifica amb un voltatge positiu, el següent bit de valor 1, es codificarà com a negatiu. Per tant, sempre es produirà una alternança en la polaritat encara que hi hagi una cadena de bits de valor 1 seguits.

Per les característiques del nostre projecte, ens veiem forçats a quedar-nos amb la codificació digital polar. El brunzidors poden prendre dos estats o sonen (els podem assignar la polaritat positiva), o no sonen (els podem assignar la polaritat negativa). Per tant, aquesta sembla a priori, la millor codificació per continuar desenvolupant el projecte

4.4.3Codificació digital polar

En aquest apartat, parlarem dels tipus de codificacions digitals polars. Podem trobar tres codificacions diferents.
- NRZ(no retorn a zero)
- RZ (retorn a zero)
- Bifase (auto sincronitzades)

NRZ (no retorn a zero)

En aquest tipus de codificació, normalment els bits de valor 1 es representen amb la polaritat positiva. Els bits de valor 0, es representen normalment amb un voltatge negatiu. Aquest tipus de codificació, no disposa de cap forma de seguretat per no perdre informació. Es poden aplicar tècniques per millorar-ne la seguretat.
Si s'utilitza aquesta codificació en una sistema de comunicació, és necessari afegir tècniques de sincronisme externes per poder fer possible la comunicació. En la figura 6, podem veure un exemple de codificació NRZ

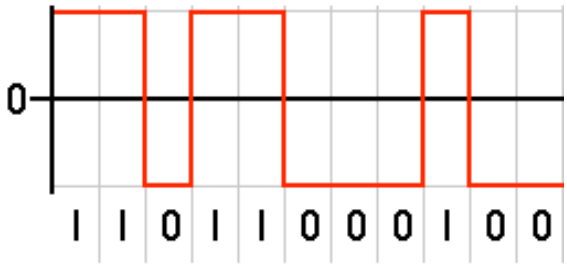


Figura 6 Nivell de senyal en codificació NRZ

RZ (retorn a zero)

En aquest tipus de codificació, els bits de valor 1, es representen normalment amb polaritat positiva. Els bits de valor 0, es representen normalment amb un voltatge negatiu. En els dos casos però, la polaritat torna al valor 0 a la meitat de cada pols. Aquest canvi es produeix encara que hi hagin cadenes de bits de valor 1 o 0 seguides. Aquesta codificació disposa de sincronisme. Això vol dir que, en una comunicació la senyal de sincronisme no s'ha d'enviar per un altre canal. La part negativa d'aquesta codificació és que hem d'enviar el doble de bits que amb la codificació NRZ. En la figura 7, podem veure un exemple de codificació RZ

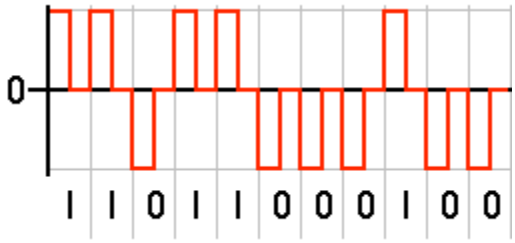


Figura 7 Nivell de senyal en codificació RZ

Bifase (auto sincronitzades) o Manchester

En aquest tipus de codificació, hi ha una transició entre dos nivells a la meitat de cada bit, com en el cas anterior. És una senyal auto sincronitzada, ja que en cada bit, podem obtenir la senyal de sincronisme. Com en el cas anterior, això fa que doblem la quantitat de bits que necessitem enviar

La senyal de sincronisme en aquest cas, queda fusionada amb la senyal de les dades. Una transició de polaritat negativa a positiva, representa un bit de valor 1. Una transició de polaritat positiva a negativa, representa un bit de valor 0. L'avantatge d'aquesta codificació, és que encara que tinguem una cadena de bits de valor 1 o 0, es produirà una transició com a mínim a la meitat de cada bit. En la figura 8 podem veure un exemple de codificació Manchester

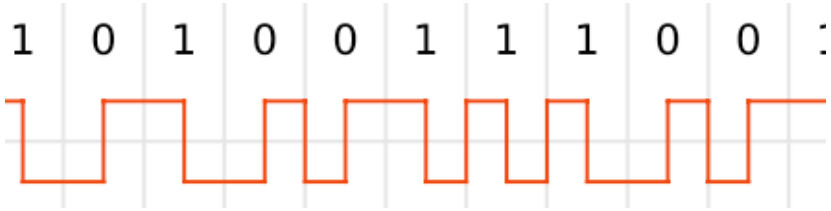


Figura 8 Nivell de senyal en codificació Manchester

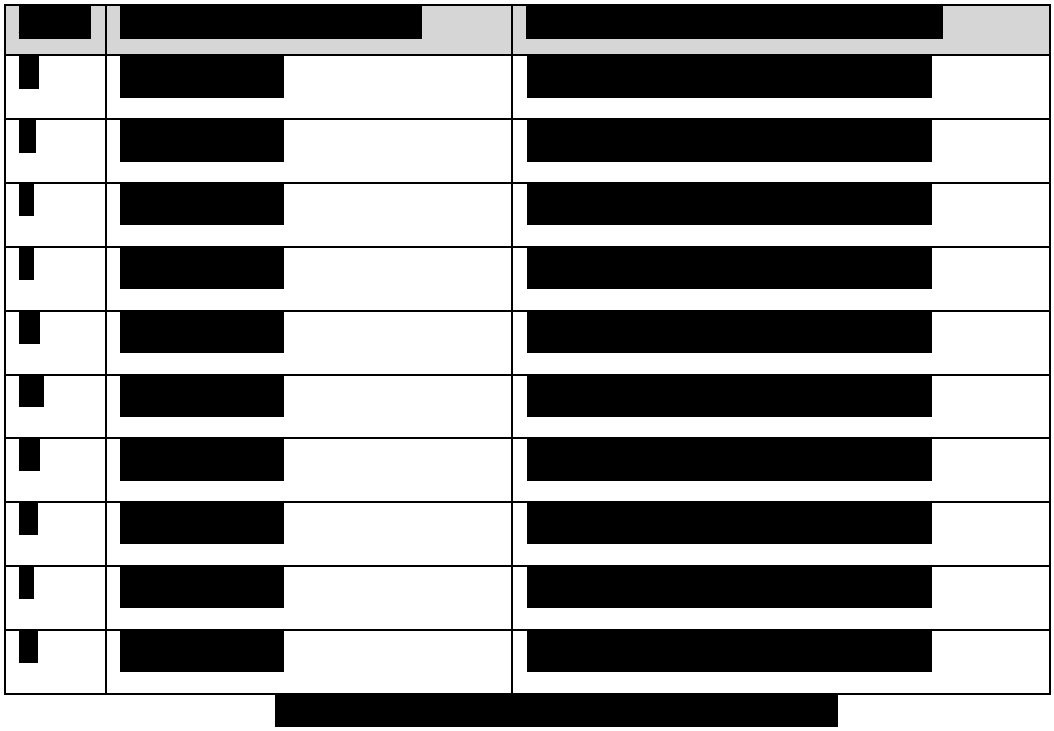
4.4.4Conclusions

Un cop hem vist els tipus de codificacions, creiem necessari escollir una codificació que disposi de sincronisme, encara que això signifiqui que multipliquem per 2 la quantitat de bits que hem d'enviar. Aquest tipus de codificació facilitarà molt la nostra feina. Podrem detectar cadenes de 0 i 1's seguits.

Els brunzidors només disposen de dos nivells de senyal.

Per tot això, hem decidit que el nostre missatge estarà codificat amb la codificació Manchester.

Per tant, el missatge “HELLO WORLD” queda com es mostra en la taula 2:



4.4.5Modificació software

Un cop hem pres la decisió d’utilitzar la codificació Manchester, hem d’aplicar unes petites modificacions en el software, perquè cada vegada que haguem de transmetre un bit de valor 1, s'emeti un bit de valor 0 després. Cada vegada que haguem d'emetre un bit de valor 0, s'emeti també un bit de valor 1 després.

Per adaptar el codi a la nova codificació, fem un canvi a la funció *void EnviaByte (char a).*

Com es pot observar, després d'emetre un bit de valor 0 el PIC emet un bit de valor 0 i al inrevés.

4.5 Organització de la informació

4.5.1 Introducció

Amb el nostre projecte, som capaços d'enviar missatges amb codificació Manchester. Fins ara, només hem provat d'enviar missatges artificials com "HELLO WORLD". En aquest apartat, observarem com s'organitzen les dades gràcies a una configuració exemple (real). L'arxiu s'inclou a l'annex 5. Prendrem una decisió sobre com volem organitzar la informació que volem enviar i adaptarem el codi al format.

L'objectiu d'aquest apartat és decidir com s'organitza la informació que volem enviar en la transmissió del missatge i fer les modificacions pertinents.

4.5.2 Organització de la informació

Disposem d'un arxiu exemple on podem veure com es guarda la informació sobre la configuració d'un equip JCM.
Abans de continuar prenent decisions, observarem la primera línia del codi per veure com està organitzada.

```
{{0x00,0x00,0xA0,0x00,0x05,0x05,0x00,0x05},8}
```

Veiem, que la informació que hem d'enviar està organitzada en paquets de bytes. Els bytes estan separats entre ells per comes. El final de cada paquet hi ha un valor enter, ens indica el número de bytes que conté el paquet.
Si fem una ullada a tot el document, veiem que no tots els paquets tenen la mateixa llargada. En un arxiu de configuració podem trobar paquets de 8, 12 ,14 i 20 bytes de llargada.

La decisió sobre quina informació volem enviar i com l'organitzem, la prendrem per convenció.
Volem que el missatge es transmeti de manera correcta, per tant, cal que tan la part emissora com la receptora, sàpiguen quina és la convenció a la qual arribem.
Les úniques condicions sobre quina informació hem d'enviar són:
- Necessitem un byte que indiqui que comença la transmissió d'un paquet.
- Necessitem un byte que indiqui que finalitza la transmissió d'un paquet, ja que la llargada dels paquets, és variable.
- Necessitem un byte de seguretat, per saber si tots els bytes s'han enviat de forma correcta.
- Com a última condició i la més bàsica, també hem d'enviar la informació de configuració del equip JCM

4.5.3Conclusions

Segons les condicions imposades per l'apartat anterior, hem arribat a l'acord que l'organització de la informació serà com s'indica en la taula 3:

Tipus de byte	Descripció
0b1111 1111	Aquest byte, es posa el principi de la transmissió, per indicar que comença una transmissió
(bytes informació)	Tot seguit, s'envien els bytes més importants. Són els que porten la informació sobre la configuració del equip. Aquesta part pot variar la seva llargada perquè ja hem vist que els paquets poden tenir varies dimensions
(Byte seguretat)	Tot seguit, s'inclou un byte de seguretat. Per obtenir el byte de seguretat farem un càlcul. Farem un XOR binaria entre tots els bytes

	d'informació del paquet. La part receptora farà el mateix càlcul. Quan rebi un paquet, si el byte de seguretat rebut, coincideix amb el byte de seguretat que ha calculat, vol dir que la transmissió de dades, s'ha fet de forma correcta.
0b0000 0000	El final s'inclou aquest byte que indica que la transmissió del paquet ha finalitzat

Taula3 Descripció format paquet

4.5.4Modificació software (I)

A partir de les conclusions a les quals hem arribat, veiem la necessitat de crear una petita funció i modificar una petita part del software, per tal que es puguin complir les especificacions descrites en l'apartat de conclusions.

En aquesta modificació crearem una funció nova. La seva missió, serà la creació del byte de seguretat. El byte de seguretat es crea a mitjançant una operació XOR binaria entre tots els bytes d'informació del paquet.

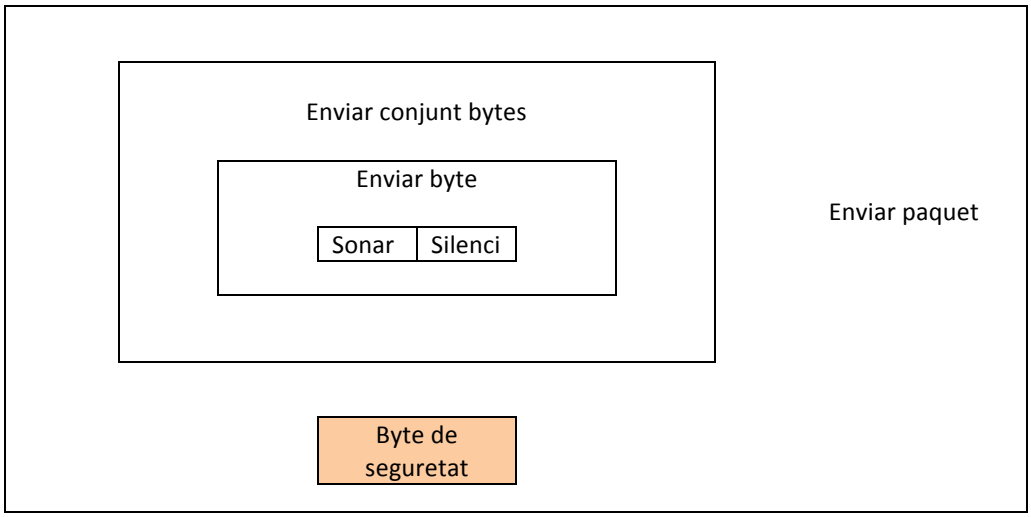


Figura 9 Organització funcions emissió. Desenvolupant byte de seguretat

```
void CrearSeguretat (char info[], int numchar)
{
		int i,j;
		char res=0b00000000;

		for (i=0; i<numchar;i++)
		{
			res=info [i]^res;
		}
		seguretat=res;
}
```

Aquesta funció té dos paràmetres d'entrada:

- Char info [ ]: indica els bytes d'informació de l'equip dels quals es vol crear el byte de seguretat

- int numchar: indica la llargada de Char info[ ]

Aquesta funció agafa el primer i el segon byte de Char info [ ] i els opera amb una XOR binaria. Després, guarda el resultat i fa el mateix procés però aquesta vegada amb el segon i el tercer byte. Aquesta operació es va repetint fins que ha operat tots els valors del paquet entre ells. El resultat es guarda en la variable "seguretat".

1.1.1. Modificació software (II)

Un cop hem calculat el byte de seguretat, ja tenim tots els bytes que ens fan falta per enviar al primer paquet al receptor. Només fa falta adaptar la funció *void SendBuffer (char buffer[ ], int numchar),* perquè sigui capaç d'enviar els paquets tal com hem decidit.

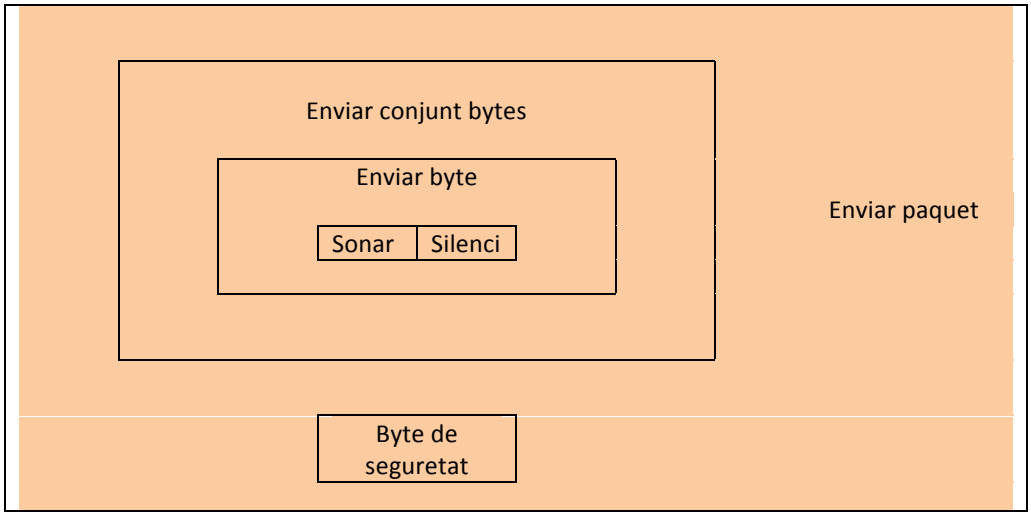


Figura 10 Organització funcions emissió. Desenvolupant enviar paquet

```
void SendBuffer (char buffer[], int numchar)
{
        int i;

        EnviaByte (0b11111111);
        for (i=0; i<numchar;i++)
        {
                EnviaByte (buffer [i]);
        }
        EnviaByte (seguretat);
        EnviaByte (0b00000000);
        SilenciTemps (700);
}
```

Aquesta funció té dos paràmetres d'entrada:

- Char info [ ]: indica els bytes d'informació del equips dels quals es vol crear el byte de seguretat

- int numchar: indica la llargada de Char info[ ]

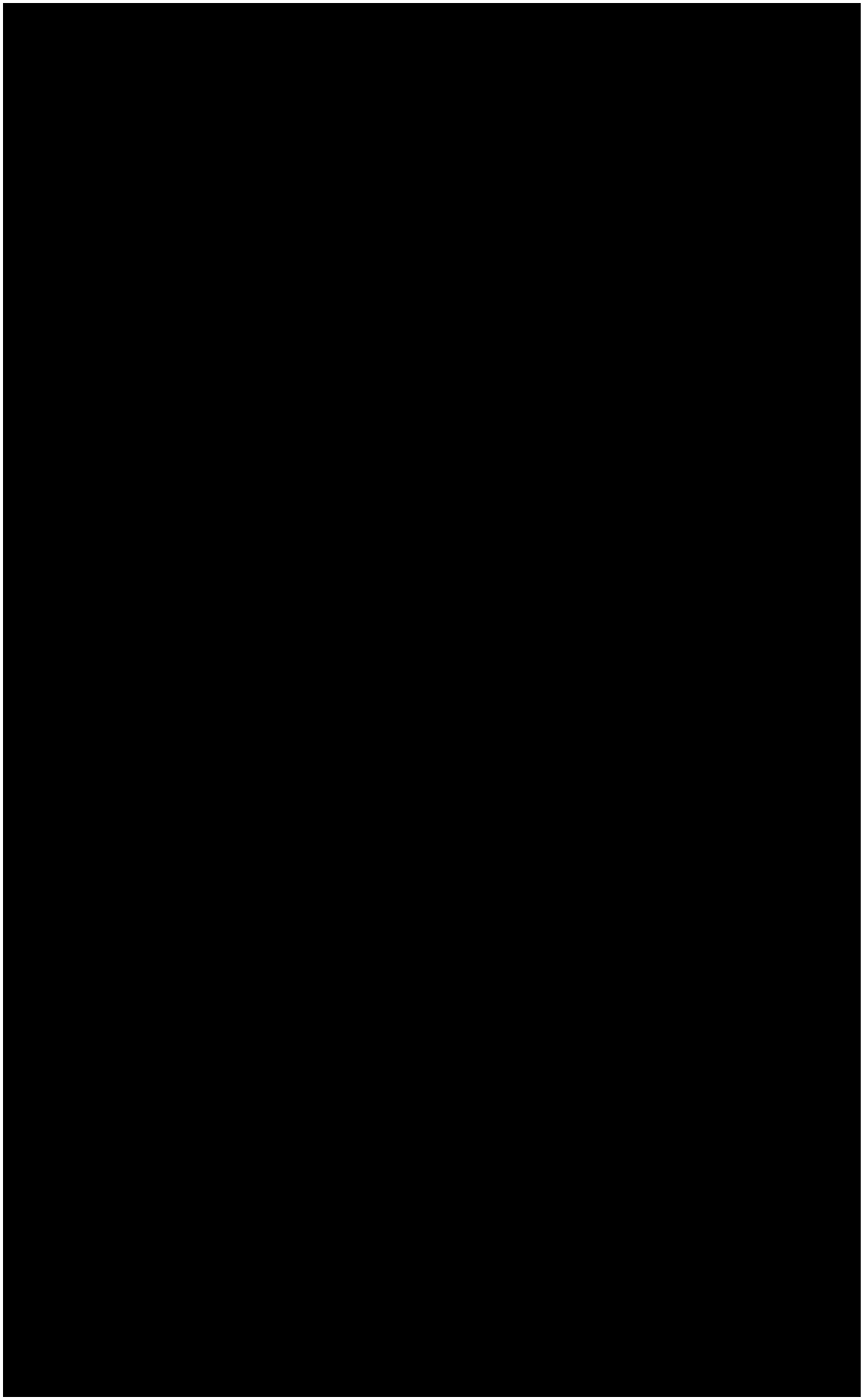
El funcionament és molt simple. Primer envia el byte 1111 1111 , que és el byte que indica que comença la transmissió. Després envia els bytes corresponents a la informació de configuració. Tot seguit, envia el byte de seguretat que hem creat amb la funció descrita anteriorment. Finalment envia l'últim byte, el 0000 0000, que és el que indica la fi de la transmissió.

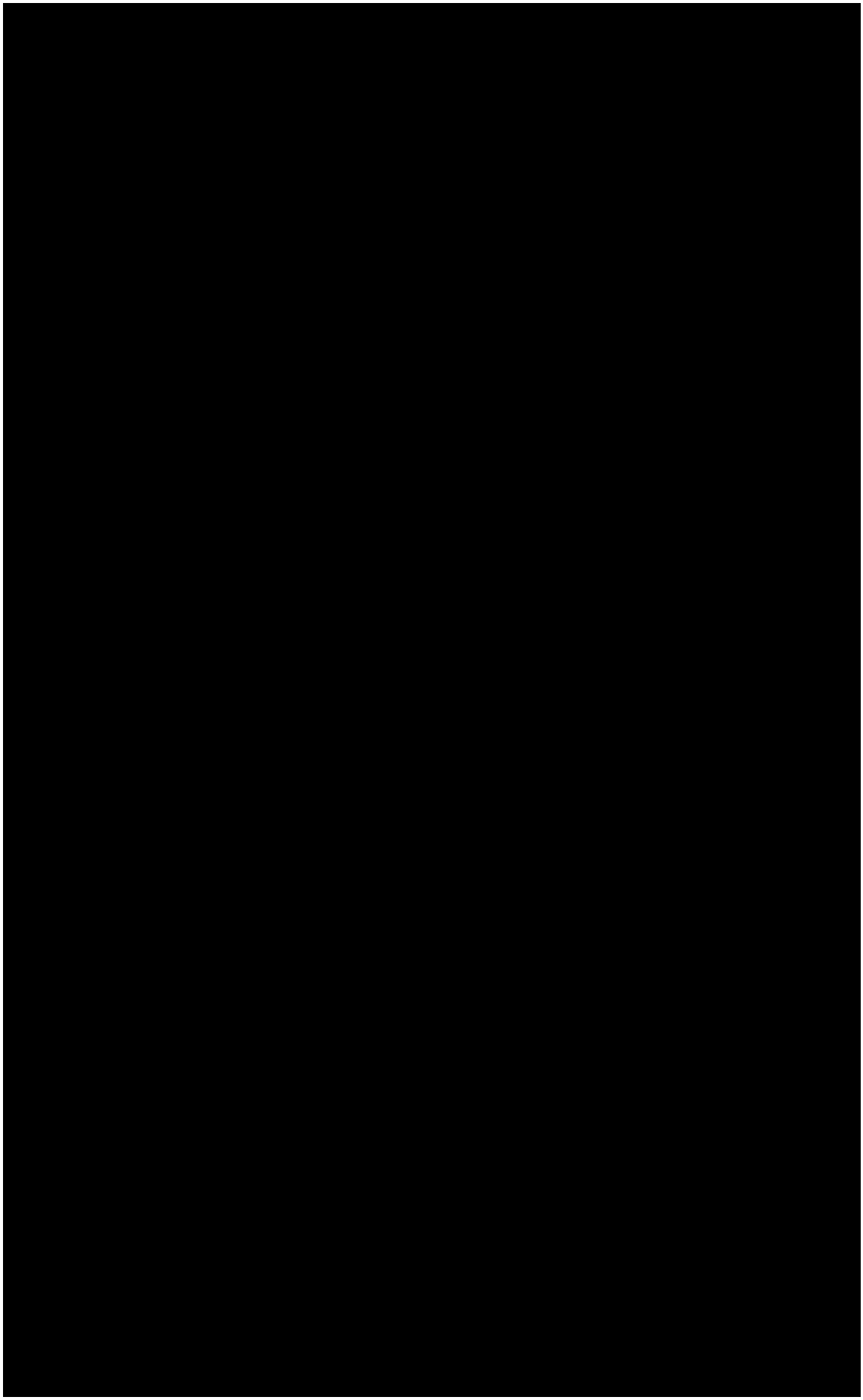
Un cop s'ha enviat tota la informació, es crea un moment de pausa entre paquet i paquet de 700ms. Hem escollit aquest temps perquè la part de recepció ha de processar cada paquet que li arriba durant aquest interval. Un cop ha processat el paquet s'ha de tornar a preparar per rebre el següent paquet, és per això que hem pensat que 700ms és un temps prudent.

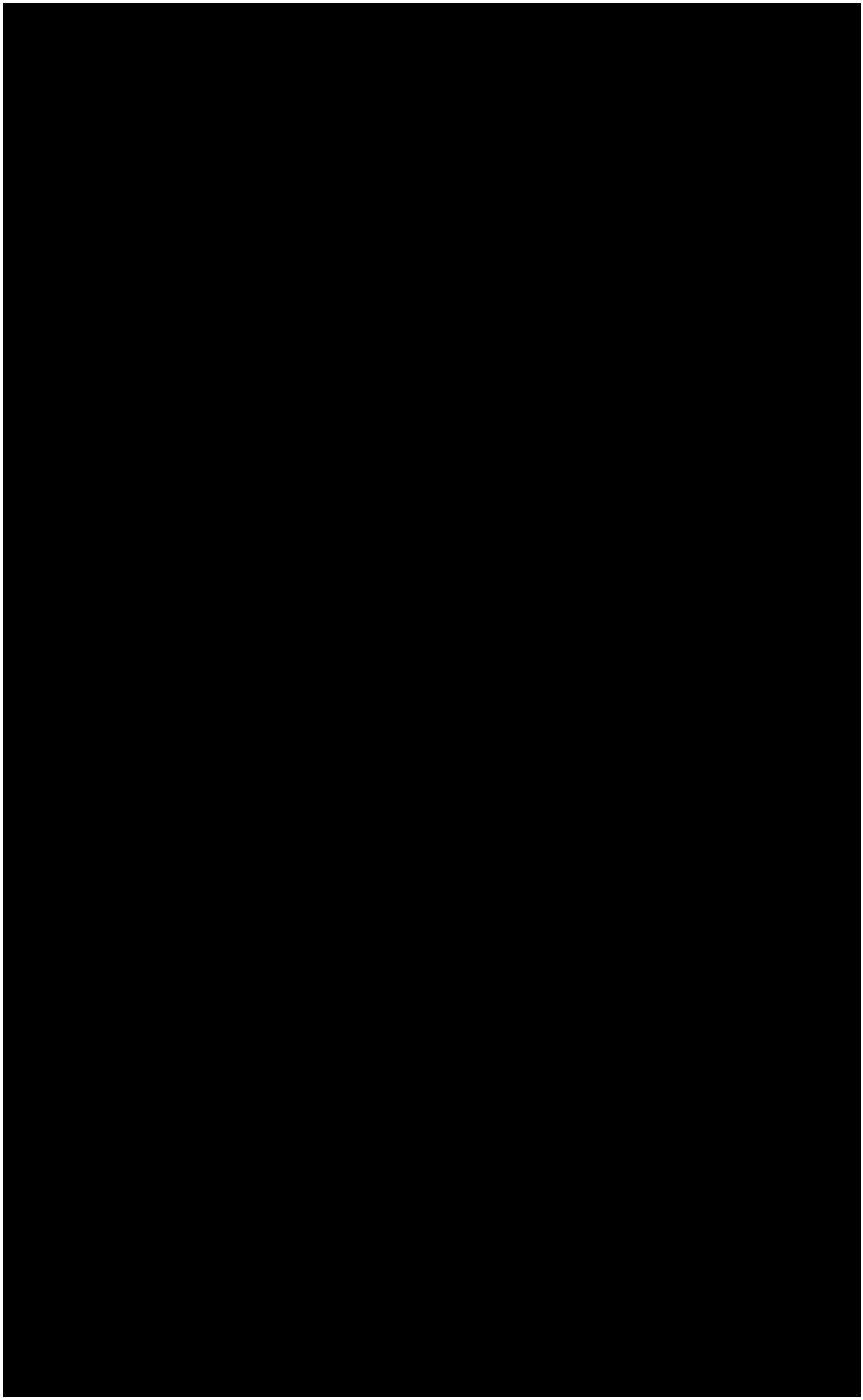
4.6 Emissió codificació exemple

En aquest apartat, es vol dissenyar la versió final del software de la part emissora del projecte. Per simular una situació real, s'ha utilitzat un fitxer exemple amb les dades d'una configuració.

Aquesta és la versió final del software:







En la primera part del codi, es configuren alguns dels aspectes del microcontrolador. Després es declaren totes les funcions de la part emissora.
La transmissió no començarà, fins que no premem l'interruptor que es troba en la Demo board. Un cop es prem el botó, comença la transmissió de totes les dades que hi ha en el fitxer de configuració.
Un cop ha acabat, es torna a preparar per començar la transmissió

4.7 Conclusions

Un cop hem arribat en aquest apartat, vol dir que hem aconseguit complir tots els objectius parcials que ens hem plantejat.
Amb el software que hem creat i la part de hardware que hem assemblat sobre de la placa d'avaluació hem aconseguit assolir l'objectiu principal per aquesta part del projecte.
Hem aconseguit que un equip de JCM emeti la seva configuració a través del brunzidor.

5 Receptor (Hardware)

5.1 Introducció

La part que s'ocupa de l'emissió de la senyal ja està desenvolupada. En aquest punt, només ens queda dissenyar la part que s'ocupa de la recepció de les dades.

El desenvolupament de la recepció serà una mica més llarg que el desenvolupament de l'emissió, és per això, que hem decidit dividir la recepció en dos apartats.
Primer parlarem del Hardware, sobre com arriben les dades a un microcontrolador.
En la segona part parlarem del Software, sobre com s'interpreten les dades i les convertim en útils perquè es puguin utilitzar

Si aconseguim resoldre els problemes i objectius que es plantegen en aquestes dues parts, completarem la part de la recepció del projecte.

En aquesta part del projecte, dissenyarem el hardware. Volem dissenyar un circuit que sigui capaç de rebre la senyal de la trucada amb la informació de l'equip, fer les transformacions que facin falta a la senyal i enviar-la a un pin d'entrada d'un PIC.

5.2 Introducció al Hardware

Quan despengem el telèfon i escoltem una configuració d'equip, l'únic que escoltem, són tot un seguit de tons a una velocitat de 40ms per to. Com es fa evident, és impossible per una persona descodificar aquesta cadena de sons i transformar-la en informació útil. Per tant, hem decidit que aquesta feina la farà un microcontrolador.

El microcontrolador, és l'aparell que s'ocuparà de descodificar totes aquestes dades. Tot i així, no es pot analitzar la senyal directament perquè necessitem que la senyal d'entrada tingui unes condiciones específiques. Necessitem fer uns canvis en la senyal per adaptar-la a les característiques del microcontrolador.

La senyal que arriba via la xarxa telefònica, arriba en dos tipus de intervals:
- Intervals on la senyal és propera els 0 volts
- Intervals on la senyal té una freqüència d'oscil·lació de 2000Hz i una amplitud entre 150mV i 500mV aprox. (Pot variar segons el volum que tinguin els tons)

L'objectiu d'aquesta part és convertir la senyal que rebem via la xarxa telefònica, en una senyal que pugui interpretar un PIC. Per això, hem d'aconseguir que a la sortida de la part receptora hi hagi una senyal de l'estil tren de polsos amb una amplitud de 5 Volts.

Per arribar a complir aquest objectiu, dissenyarem una petita placa. Alhora de dissenyar el circuit ens hem marcat uns objectius parcials. Cada objectiu parcial implica dissenyar una etapa més en el circuit. Cada etapa aporta un petit canvi a la senyal. La suma de tots els petits canvis, és la que farà possible complir amb l'objectiu global. Els objectius parcials són:

- Transportar la senyal de la xarxa telefònica fins el nostre circuit
- Amplificar la senyal.
- Filtrar la senyal de tots els sorolls. Només ens interessen els tons de 2.000Hz.
- Convertir la senyal en un tren de polsos amb una amplitud de 5V i un DC offset de 0 Volts.

5.3 Material de desenvolupament

Abans de començar a dissenyar la part de hardware, parlarem una mica de les eines que utilitzarem en aquesta part

Per desenvolupar la part del hardware, hem decidit que utilitzarem una placa de forats. Els avantatges d'utilitzar aquesta placa és que podem utilitzar components convencionals, que són molt més manejables que els components SMD. A més, per fer les connexions només hem de fer una mica de pressió sobre de la placa perquè el component quedi connectat. La placa de forats també ens permet fer proves amb més flexibilitat i podrem canviar de components i la seva disposició en qualsevol moment, les vegades que vulguem. Aquesta placa també ens permetrà fer les proves de funcionament del nostre disseny.
Un cop haguem validat el disseny de la placa, procedirem a soldar el disseny final del nostre circuit en un placa PCB. Utilitzarem en la mesura del possible components SMD.

Per analitzar la senyal i comprovar si hem complert amb els nostres objectius, utilitzarem un oscil·loscopi.

Un cop haguem complert l'objectiu de connectar la xarxa telefònica i haguem comprovat que funciona correctament, utilitzarem un IPhone per simular la senyal d'entrada. Hem enregistrat el so que emet la part emissora del projecte. El so enregistrat té les mateixes característiques que el so que prové de la xarxa. Per no haver d'establir una trucada cada

vegada que volem fer una prova i fer el procés més àgil, introduirem la senyal en el circuit utilitzant un cable d'àudio connectat a la sortida d'àudio del mòbil.

5.4 Connexió de la xarxa telefònica al circuit

5.4.1 Introducció

Discutirem sobre com podem transportar la senyal de la xarxa telefònica fins el nostre circuit a temps real i sense perdre informació.

Hem decidit, que la millor solució per aconseguir el nostre objectiu, és dissenyar una placa que adapti la senyal a les necessitats del microcontrolador. El primer pas però abans de començar a modificar la senyal és transportar la senyal fins el nostre circuit

L'objectiu d'aquesta part, és per tant, fer que el nostre circuit rebi la mateixa senyal que arriba a l'altaveu del telèfon.

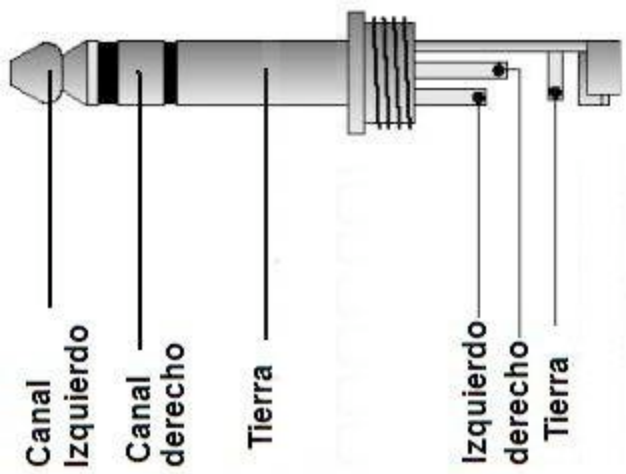
5.4.2 Metodologia

En aquest apartat, expliquem tot el procediment que hem seguit per connectar la xarxa telefònica a l'entrada de la nostra placa.

Podríem trobar moltes maneres diferents de connectar al telèfon al nostre circuit, però creiem que és important que utilitzem els recursos dels que disposem. Així no fem encarir el projecte més del necessari.

El SAT disposa d'un aparell connectat a un terminal telefònic i connectat, a través d'un cable d'àudio, a uns auriculars que també incorporen micròfon. La funció d'aquest aparell, és commutar la trucada de l'altaveu del telèfon a l'altaveu dels auriculars a través d'un interruptor.

La solució que proposem, passa per tallar el cable d'àudio que va a parar a l'auricular i connectar-lo amb la nostra placa.
Després d'una petita recerca per investigar com són els cables d'àudio, descobrim que segueixen l'esquema de connexió de la imatge 16.



Imatge 16 Esquema de connexió d'un Jack

Tallem el cable d'àudio i retirem la part on hi ha connectats els auriculars. A continuació, comencem a identificar on són totes les parts que s'indiquen a la imatge 16. Un cop les tenim identificades, posem una mica d'estany a la punta dels cables perquè siguin de més fàcil connexió a la placa de forats.

El cable que correspon al canal dret i el cable que correspon al canal esquerra, porten exactament la mateixa senyal. Tallem un dels cables perquè no molesti i l'altre el connectem a la placa.
El cable que correspon al terra també el connectem a la placa

Un cop ja tenim totes les connexions fetes ja podem comprovar si funciona la connexió. El sistema queda configurat com s'indica en la imatge 17

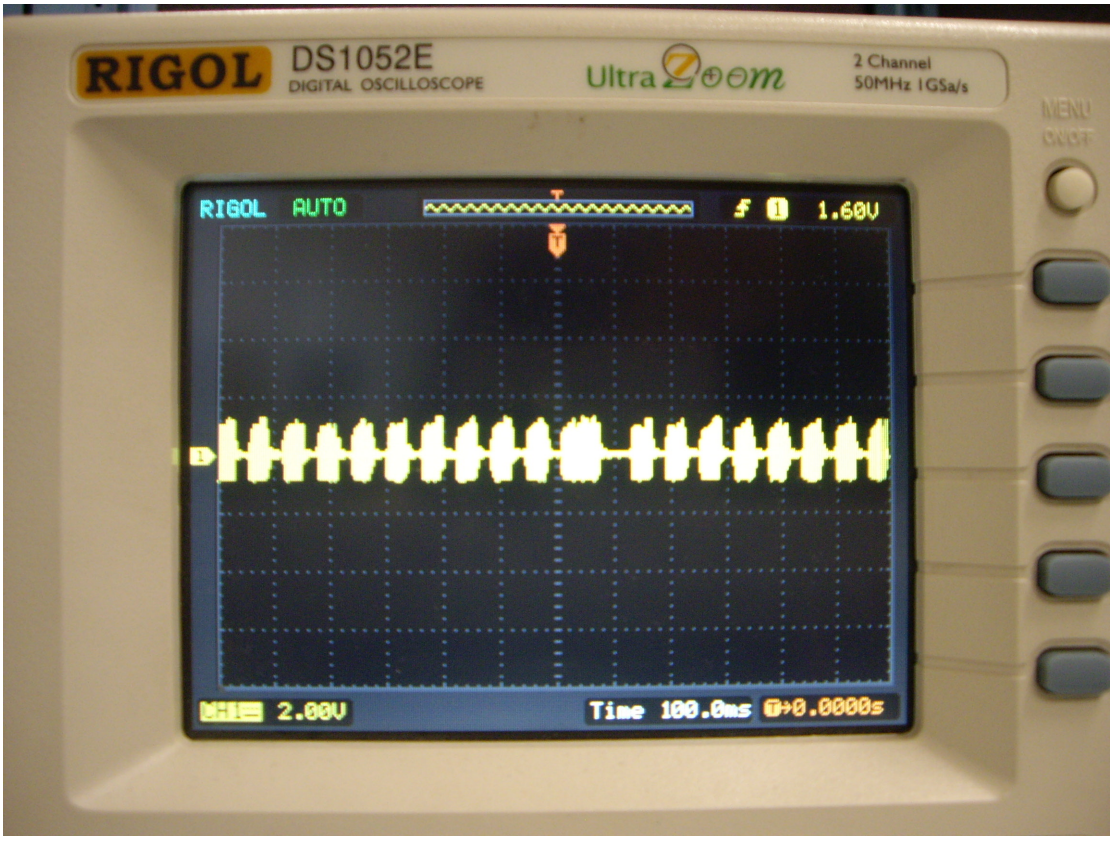
Imatge 17 Connexió de la xarxa telfònica a la placa de recepció

5.4.3Conclusions

Per veure si la connexió compleix amb els objectius, només falta fer una simulació i veure quina és la senyal a la sortida.

Establim un trucada entre la part emissora i la part receptora. Amb la sonda del oscil·loscopi, comprovarem quina senyal arriba a la placa. Configurem l'aparell de telèfon perquè la senyal passi al circuit i no a l'altaveu. Fem que la part emissora comenci a emetre la senyal.

A través del oscil·loscopi observem que la senyal arriba al circuit (imatge 18), per tant podem donar per complert el nostre objectiu. Som capaços de fer arribar la senyal d'àudio a la nostre placa a temps real i sense pèrdua d'informació.



Imatge 18 Senyal d'audio a l'entrada

Recordem que a partir d'aquest apartat, l'establiment de la trucada es simula mitjançant un arxiu d'àudio amb un IPhone

5.5 Amplificador

5.5.1 Introducció

En l'apartat anterior, hem aconseguit que la senyal passi fins la nostre placa. Observem que els nivells de potència poden variar segons el que ens allunyem o ens apropem dels brunzidors, també pot variar segons els models de terminal de telèfon que utilitzem. En tots els casos, la senyal arriba amb uns nivells massa baixos perquè la senyal pugui ser útil pel microcontrolador.

Parlarem sobre com podem aconseguir amplificar la senyal perquè tingui una amplitud de 5 volts aproximadament. Amb una amplitud de 5 volts, la senyal podrà ser tractada pel microcontrolador.

L'objectiu d'aquest apartat serà doncs, dissenyar un circuit el més simple possible per ampliar la senyal fins aproximadament els 5 Volts.

5.5.2Amplificadors operacionals

Per amplificar la senyal, utilitzarem amplificadors operacionals perquè creiem que amb aquest mètode aconseguirem un disseny més senzill.
En aquest apartat descriurem breument què és un amplificador operacional.

Un amplificador operacional (també conegut com A.O. o op-amp), és un circuit electrònic (normalment es presenta com un circuit integrat) que té dues entrades i una sortida. La sortida del amplificador operacional, és la deferència entre les dues entrades multiplicat per un factor , G (Guany). El guany és igual al voltatge de sortida Vout .
$Vout = G \cdot (V+ - V-)$

Els amplificadors operacionals, es poden utilitzar per realitzar operacions matemàtiques (sumar, restar, multiplicar, ...) es per això que reben aquest nom

Els A.O. tenen un guany infinit, una impedància d'entrada infinita, un ample de banda infinit, una impedància de sortida nul·la, el temps de resposta és nul i no té soroll. Com que la impedància d'entrada és infinita, també es diu que les corrents d'entrada són zero

Els amplificadors operacionals es representen com s'indica en la figura 11:

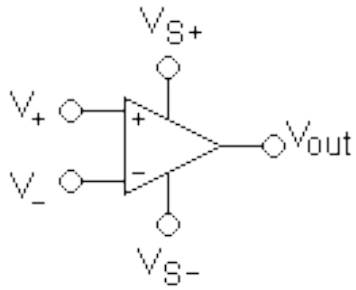


Figura 11 Connexions amplificador operacional

5.5.3Amplificador Operacional, configuració no-inversora

Com hem pogut veure en l'apartat anterior, els amplificadors operacionals són circuits que s'utilitzen per realitzar moltes funcions diferents, per tan existeixen moltes configuracions diferents per poder oferir un funcionament adequat a cada necessitat.

En el nostre cas, necessitem els amplificador operacionals perquè amplifiquin la senyal i no apliquin cap altre canvi. Es per això, que hem decidit que aplicarem una configuració no-inversora per amplificar la senyal a través dels op-amps. Podem veure com són les connexions en aquest tipus de configuració en la figura 12.

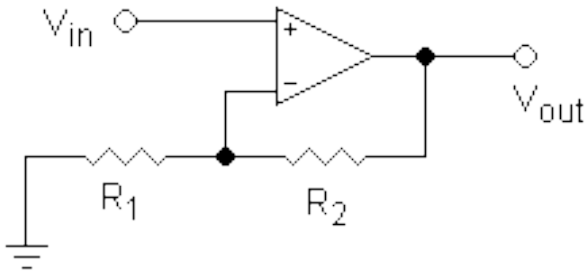


Figura 12 Configuració connexió no-inversora

La senyal d'entrada passa pel port positiu del amplificador operacional. Com hem vist, el guany dels amplificadors és molt gran i el voltatge del pin positiu és igual al voltatge del pol negatiu. Si coneixem el voltatge del pin negatiu, podem calcular la relació que existeix entre el voltatge de sortida i el voltatge d'entrada fent ús d'un petit divisor de tensió
Per aquesta configuració s'estableix aquest relació entre les variables

(Equació 7) $$V_{out} = V_{in}(1 + \frac{R_2}{R_1})$$

Volem que el voltatge de sortida sigui de 5V. El voltatge d'entrada pot variar, per tant, farem el càlcul amb un voltatge d'entrada alt i un voltatge d'entrada baix. Per aplicar aquest disseny al nostre circuit només ens falta trobar el valor de les dues resistències. Com que només tenim una equació, però dues incògnites, decidim fixar R1 amb un valor de 1KΩ. Hem decidit fixar la resistència amb aquest valor perquè és un valor convencional i no ens costarà gens trobar resistències amb aquest valor.

Primer provem de resoldre l'equació amb un valor d'entrada de 0,150V

(Equació 8) $V_{out} = V_{in}(1 + \frac{R_2}{R_1})$

(Equació 9) $5 = 0{,}150(1 + \frac{R_2}{1K})$

(Equació 10) $\frac{5}{0{,}150} = 1 + \frac{R_2}{1K}$

(Equació 11) $33{,}33 - 1 = \frac{R_2}{1K}$

(Equació 12) $R_2 = 32{,}33 \cdot 1K$

(Equació 13) $R_1 = 32{,}33K$

Ara provem de resoldre un altre cop la mateixa equació, però aquesta vegada amb un voltatge d'entrada de 0,5V

(Equació 14) $V_{out} = V_{in}(1 + \frac{R_2}{R_1})$

(Equació 15) $5 = 0{,}5(1 + \frac{R_2}{1K})$

(Equació 16) $\frac{5}{0{,}5} = 1 + \frac{R_2}{1K}$

(Equació 17) $10 - 1 = \frac{R_2}{1K}$

(Equació 18) $R_2 = 9 \cdot 1K$

(Equació 19) $R_1 = 9K$

Amb aquest càlculs podem observar que aquesta configuració, si fixem tots els valor de les resistències, si hi ha un petit canvi en la tensió de la senyal d'entrada, podria fer que deixes de funcionar tal com i nosaltres volem. És per això, que per resoldre aquesta inconvenient, substituirem el valor de R1 per un potenciòmetre de 50K. Amb aquest potenciòmetre ens assegurem que sempre serem capaços d'amplificar la senyal. Si hi ha un canvi en la senyal d'entrada només ens farà falta ajustar el potenciòmetre.

Per tant, el disseny d'aquesta etapa de hardware queda com es mostra en la figura 6:

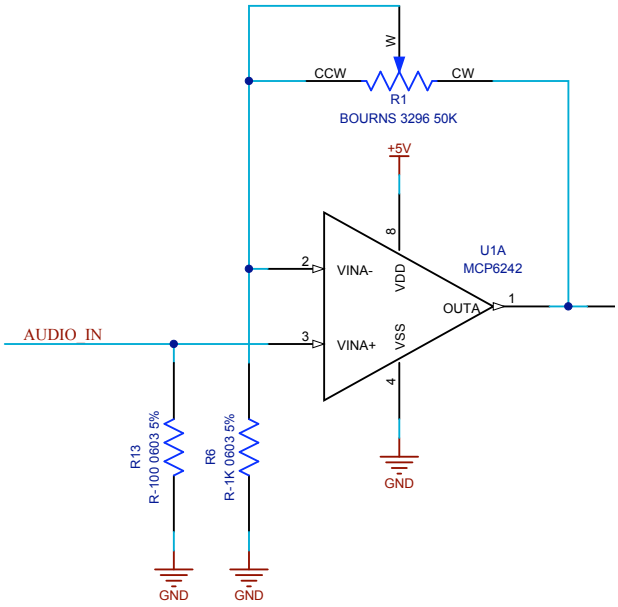


Figura 13 Esquemàtic etapa amplificadora

5.5.4Amplificador operacional MCP6242

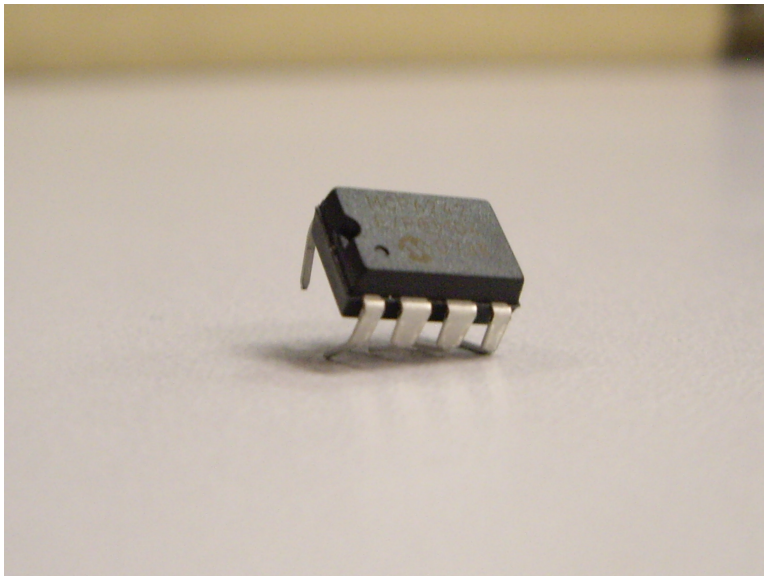
Un cop hem decidit quin serà el disseny de l'etapa que amplifica la senyal, només ens queda escollir quins amplificadors operacionals utilitzarem. Consultarem els datasheets dels proveïdors d'amplificadors operacionals. Escollirem un model perquè formi part del nostre disseny

Després de consultar els datasheets dels proveïdors usuals de JCM, hem decidit escollir els amplificadors operacionals MCP6242.

Són del tipus rail-to-rail, significa que per funcionar només els fa falta alimentació positiva. Els amplificadors que no són d'aquest tipus necessiten un port amb alimentació positiva i un amb alimentació negativa

Els MCP6242 suporten un ample de banda de 55Khz, que és molt superior a les nostres necessitats, recordem que la senyal que podem rebre a través de la xarxa telefònica va dels 300Hz als 3400Hz

El preu també és un motiu alhora d'escollir aquest model, aquest és el més econòmic que hem trobat. El podem veure en la imatge 19



Imatge 19 Amplificador operacional MCP6242

En l'annex 6 s'inclou el datasheet dels amplificadors operacionals MCP6242.

5.5.5Conclusions

Ara que tenim tots els components del nostre disseny, només falta comprovar si el circuit té el mateix comportament que en la teoria.

Agafem tots els components i els posem en la placa de forats tal i com indica el nostre disseny. Injectem la senyal dels brunzidors a l'entrada del circuit. Observem com és la senyal a la sortida del circuit, amb un oscil·loscopi. En podem veure el resultat en la imatge 20

Imatge 20 Senyal amplificada

Després d'ajustar el potenciòmetre, observem que la senyal a la sortida del amplificador té una amplitud aproximadament de 5 Volts. Per tan, hem complert amb l'objectiu d'aquest apartat.

5.6 Filtre

5.6.1 Introducció

Disposem d'un sistema que fa arribar la senyal sonora al nostre circuit. Una part del circuit actua com un amplificador que amplia la senyal fins a fer-la útil pel microcontrolador. El problema, és que del conjunt de senyal que arriba fins aquest punt del circuit, només ens interessen els sons de 2.000Hz que corresponen els que han estat creats pel brunzidor. La resta de sons de la senyal, els considerem soroll i ens podrien fer interpretar el missatge de forma errònia. És per aquest motiu, que hem decidit aplicar un filtre que ens permeti descartar tot allò que no és útil per complir amb l'objectiu del projecte.

El nostre objectiu és dissenyar un filtre que elimini tot el soroll de la senyal i que només deixi passar els tons emesos pel brunzidor de la part emissora.

Abans de dissenyar un filtre, farem un petita recerca per veure quins tipus de filtre hi ha, i n'escollirem un.

5.6.2Descripció de filtre electrònic

Un filtre electrònic, és un circuit que discrimina una determinada freqüència o un conjunt de freqüències d'una senyal elèctrica quan passa per ell. En modifica l'amplitud i la fase.

Existeixen diferents tipus de filtres, en funció de les freqüències que deixa passar. Filtre passabaix, passabanda o passaalt

També hi ha altres tipus de filtres depenent de quina sigui la seva funció de transferència, filtre de Butterworth, filtre de Txebixev , filtre de Cauer i filtre de Bessel

Els filtres electrònics, es formem amb interconnexions de components passius: resistències, condensadors, bobines i components actius: transistors, amplificador operacionals,...

5.6.3Tipus de filtres en funció de la freqüència

Segons la seva resposta en freqüència, podem classificar els filtres d'aquesta manera:

Filtre passabaix: Correspon a un filtre, caracteritzat per permetre el pas de les freqüències més baixes i atenuar les freqüències més altes. El filtre requereix de dos terminals d'entrada, dos de sortida i d'una caixa negra, també anomenada quadripol. Podem tenir qualsevol valor de freqüència a l'entrada, però a la sortida, només seran presents les freqüències que permeti passar el filtre.

Filtre passaalt: Un filtre passaalt, és un circuit format per una resistència i un condensador connectats en sèrie, de manera que aquest permet només el pas de freqüències per sobre d'una freqüència en particular. Aquesta freqüència, és la freqüència de tall (Fc) i atenua les freqüències per sota d'aquesta freqüència. Aquest filtres RC, no són perfectes, per la qual cosa es fa l'anàlisi en el cas ideal i en el cas real.

Filtre passabanda: És aquell, que permet el pas de les components de freqüència contingudes en un determinat rang de freqüències, entre una freqüència de tall superior FH i una d'inferior FL

5.6.4 Tipus de filtre en funció de la seva funció de transferència

Els filtres, també els podem classificar segons la seva funció de transferència. La funció de transferència descriu la forma en què la senyal aplicada canvia en amplitud i en fase en travessar el filtre. La funció de transferència escollida caracteritza el filtre.

Filtre de Butterworth: Caracteritzat per tenir una banda de pas suau i un tall agut

Filtre de Txebixev: Caracteritzat per tenir un tall agut però amb una banda de pas amb ondulacions

Filtre de Cauer: Caracteritzat per aconseguir una zona de transició més abrupta que els anteriors, a canvi d'oscil·lacions en totes de les seves bandes

Filtre de Bessel: Caracteritzat perquè en el cas de ser analògic, assegura una variació de fase constant.

5.6.5 Disseny del filtre

Dissenyarem el filtre que implementarem en el nostre circuit i que ens servirà per filtrar tot el soroll que rebem del telèfon.

Un cop hem estudiat tots els tipus de filtres, prenem les següents decisions:

- Utilitzarem un filtre tipus passabanda. Arriben freqüències entre 300Hz i 3400Hz, només ens interessa rebre les senyals a 2000Hz, és per això, que el nostre filtre només deixarà passar senyals compreses entre els 1500 i els 2500Hz
- El filtre que dissenyarem, serà del tipus Txebixev, perquè tot i tenir una banda de pas amb ondulacions, obtindrem un tall agut.

En el disseny del filtres, utilitzarem amplificadors operacionals, utilitzarem els mateixos que hem utilitzat en l'apartat anterior.

Per crear el disseny del filtre utilitzarem "FilterLab".

FilterLab és una eina de software creada per Microchip, que ajuda en el disseny de filtres actius. Aquesta eina, crea un esquema del circuit amb els seus components i mostra per pantalla la resposta del filtre.

Introduïm quines són les característiques que volem pel filtre dins del software FilterLab.
Volem un filtre passabanda, amb un tall inferior de 1500hz, un tall superior de 2500hz i volem que el filtre sigui del tipus Txebixev. Per no complicar molt el nostre disseny i no tenir excessives etapes en el filtre, decidim que volem un filtre d'ordre 4.

En la figura 14 podem veure quin és el filtre proposat per FilterLab:

Podem veure la resposta i comprovar que el filtre, almenys en la versió teòrica, compleix amb el requisits de disseny en la figura 15

5.6.6El divisor de tensió

Com es pot veure en l'esquemàtic del nostre filtre, el port positiu dels amplificadors operacionals que utilitzem, necessitem una senyal que sigui la meitat de la senyal d'alimentació del amplificador rail-to-rail. Els amplificadors tenen una alimentació de 5V, per tant, a l'entrada del terminal positiu necessitem una senyal de 2,5V.

Per aconseguir aquesta alimentació, aplicarem un divisor de tensió a la senyal que utilitzen els op-amps per alimentar-se.

Un divisor de tensió, és una configuració de circuit elèctric que reparteix la tensió d'una font entre dues o més impedàncies connectades en sèrie, tal i com es mostra en la figura 16

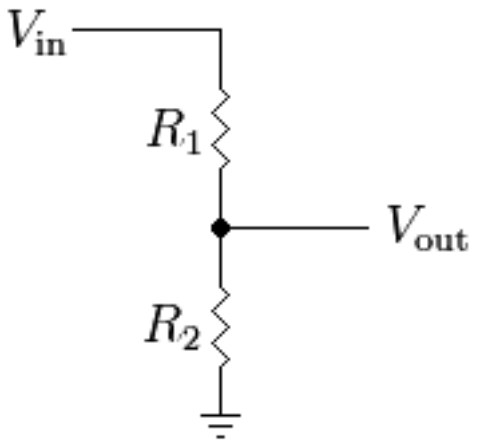


Figura 16 Connexió divisor de tensió

Aquesta configuració compleix amb la següent fórmula

(Equació 20) $$V_{out} = \frac{R_2}{R_1+R_2} \cdot V_{in}$$

Coneixem el voltatge d'entrada, 5V, i volem que a la sortida hi hagi una alimentació de 2,5 volts. Fixem el valor de R2 en 100KΩ i ara ja podem resoldre aquesta equació.

(Equació 21) $$V_{out} = \frac{R_2}{R_1+R_2} \cdot V_{in}$$

(Equació 22) $$2{,}5 = \frac{1K}{R_1+1K} \cdot 5$$

(Equació 23) $$\frac{2{,}5}{5} \cdot (R_1 + 1K) = 1K$$

(Equació 24) $$(R_1 + 1K) = \frac{1K}{0{,}5}$$

(Equació 25) $$R_1 = 2K - 1K = 1K$$

Per tan el nostre divisor de tensió queda configurat com en la figura 17. Fem arribar la senyal del divisor de tensió a tots els punts del circuit, on es faci necessari un voltatge de 2,5V

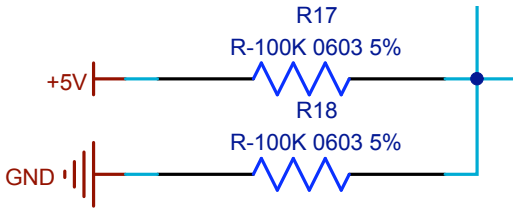


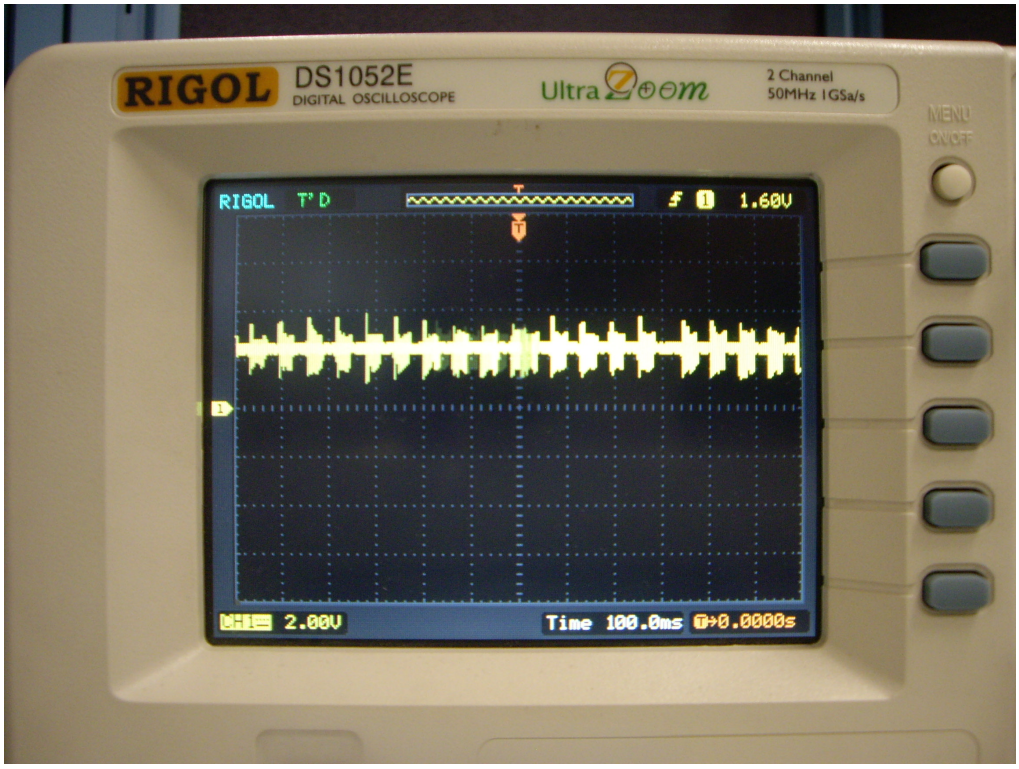
Figura 17 Esquemàtic divisor de tensió

5.6.7Conclusions

Abans de validar el disseny del filtre, hem de comprovar que en la pràctica el filtre es comporta igual que en la teoria.

Col·loquem tots els components per crear el filtre tal i com s'indica en l'esquemàtic. Connectem el filtre al circuit que tenim dissenyat fins aquest punt i connectem el divisor de tensió perquè les entrades positives dels amplificadors de tensió tinguin una alimentació de 2,5V.

Per comprovar si el filtre funciona, connectarem a l'entrada del circuit un generador de funcions. El generador de funcions el configurem perquè introdueixi una senyal amb les mateixes característiques que quan arriba per la xarxa telefònica. A la sortida del circuit connectem l'oscil·loscopi per comprovar com és la senyal a la sortida del equip.
Començarem injectant una senyal a una freqüència de 0Hz i anirem pujant la freqüència 100Hz cada 5 segons fins arribar als 4000Hz. Durant aquest procés, observarem com és la senyal que veiem a través del oscil·loscopi.



Imatge 21 Senyal a la sortida del filtre

Observem que quan injectem una senyal amb un freqüència d'entre 0 i 1250Hz a la sortida del nostre filtre detectem una senyal molt baixa. A partir de 1250Hz fins 1500Hz la senyal que veiem a la sortida va augmentat progressivament fins els 2V. Quan passem per les freqüències entre 1500Hz i 2500Hz, veiem que el valor d'energia es manté, tot i que al mig d'aquest període podem observar una pèrdua d'uns 0,2V. La imatge 21 mostra l'instant en que observem la sortida a 1700Hz.
Dels 2500Hz als 2750Hz podem observar com l'energia torna a baixar progressivament. A partir dels 2750Hz no s'observa energia a la sortida del filtre

Com hem pogut comprovar, el filtre teòric compleix a la pràctica amb les especificacions que necessitem. Per tant, podem concloure que hem complert amb l'objectiu. La senyal que tenim a la sortida del filtre, és una senyal que només aporta informació emesa pels brunzidors.

5.7 Conversor de senayl AC a DC

5.7.1 Introducció

Connectem totes les parts de circuit que hem dissenyat fins ara, un després de l'altre. Apliquem la senyal de la part emissora, en podem distingir dos tipus de senyals diferents a la sortida:

- En el primer, veiem que la sortida de circuit hi ha una absència de senyal. Aquesta absència de tensió, l'atribuïm a que durant aquest interval, la part emissora està transmeten un bit de valor 0, o sigui, estem transmetent un silenci del brunzidor. Aquesta senyal no li fa falta cap tipus de modificació per fer-la compatible amb el microcontrolador
- El segon tipus de senyal, veiem que hi ha intervals on tenim una tensió que oscil·la a 2000Hz. Durant aquest interval la part emissora està emeten un bit de valor 1, per tant estem emeten un so del brunzidor. En aquest tipus de senyal li farem uns petits canvis. Convertirem aquest interval de senyal en un sol pols, així el farem útil pel microcontrolador.

Després del filtre passabanda, per culpa de les pèrdues del circuit, la senyal arriba molt atenuada. A l'entrada del filtre hem amplificat la senyal a un amplitud de 5V, creiem que és necessari tornar a amplificar la senyal fins a una amplitud de 5V un altre cop.

L'objectiu d'aquesta part és amplificar la senyal un altre cop perquè torni a tenir una amplitud de 5V, transformar els intervals de senyal que rebem de 2000Hz, per convertir-los en polsos de la mateixa durada que els intervals.

5.7.2 Amplificador

La sortida del filtre, té uns nivells d'energia molt baixos per culpa de les pèrdues i l'atenuació, és per això, que hem decidit tornar a amplificar la senyal.

Per amplificar la senyal, utilitzarem el mateix esquema que hem utilitzat en l'etapa anterior, però aquesta vegada col·locarem una potenciòmetre amb una resistència més alta així podrem amplificar senyals més petites

Com ja hem vist, l'amplificador que utilitzem respon a l'equació 5. Si volem que el nivell de sortida a l'amplificar sigui de 5 Volts, a R1 tenim un valor de 1KΩ i a R2 tenim un potenciòmetre de 100KΩ, si resolem l'equació, sabrem quin és el valor més petit, teòric, que podem amplificar fins a 5Volts.

(Equació 26) $$V_{out} = V_{in}(1 + \frac{R_2}{R_1})$$

(Equació 27) $$5 = V_{in}(1 + \frac{100K}{1K})$$

(Equació 28) $$5 = V_{in}(1 + 100K)$$

(Equació 29) $$V_{in} = \frac{5}{100001} = 0{,}049mV$$

Amb aquesta configuració podrem arribar a amplificar senyals a partir dels 0,049mV

5.7.3Detector de màxims

5.7.3.1Introducció

Gràcies a l'amplificador que acabem de col·locar en el circuit, tenim la senyal amb uns nivells d'energia prou alts per poder-li aplicar més canvis. Amplificant la senyal també ens hem assegurat que encara que l'etapa del circuit que dissenyarem a continuació tingui pèrdues, no perdrem la senyal del tot.

Com hem descrit anteriorment en aquest punt del circuit, observem que la senyal que rebem poden ser intervals d'absència de senyal o intervals amb una senyal oscil·lant a 2000Hz.

L'objectiu d'aquest apartat és modificar aquesta senyal oscil·lant i convertir-la en polsos. Si aconseguim aquest objectiu, la senyal al final del circuit serà una senyal de tren de polsos, que és l'objectiu final de tota aquesta part.

Per aconseguir aquest objectiu primer farem passar la senyal per un detector de màxims i després per un circuit Schmitt Trigger

5.7.3.2Descripció i disseny del detector de màxims

Abans de dissenyar el detector de màxims, farem una breu descripció.

El detector de màxims, és un circuit que utilitza elements com resistències, díodes i condensadors.

En la sortida del circuit aplica un corrent DC, equivalent el valor màxim aplicat a la senyal de l'entrada en AC. A partir d'una senyal formada per una sèrie de pics de tensió irregulars, com pot ser una senyal AC, en volem saber el valor màxim.

Aquest disseny, carrega un condensador amb la senyal d'entrada, quan la senyal desaparegui, per evitar que es descarregui, col·loquem un díode. En teoria, el condensador hauria de mantenir aquest nivell d'energia per sempre. El que volem aconseguir és que el condensador es vagi descarregant poc a poc. Així, podrem detectar un nou màxim pic de tensió. Per aconseguir que el condensador es descarregui poc a poc, col·locarem una resistència.
En la imatge 22, podem comprovar com és la configuració d'un detector de màxims

El temps de resposta del circuit depèn de la variable RC. El temps de resposta del detector de màxims respon l'equació 30 (La Tau ens indica el temps que triga un condensador a carregar-se):

(Equació 30) $\tau = R \cdot C$

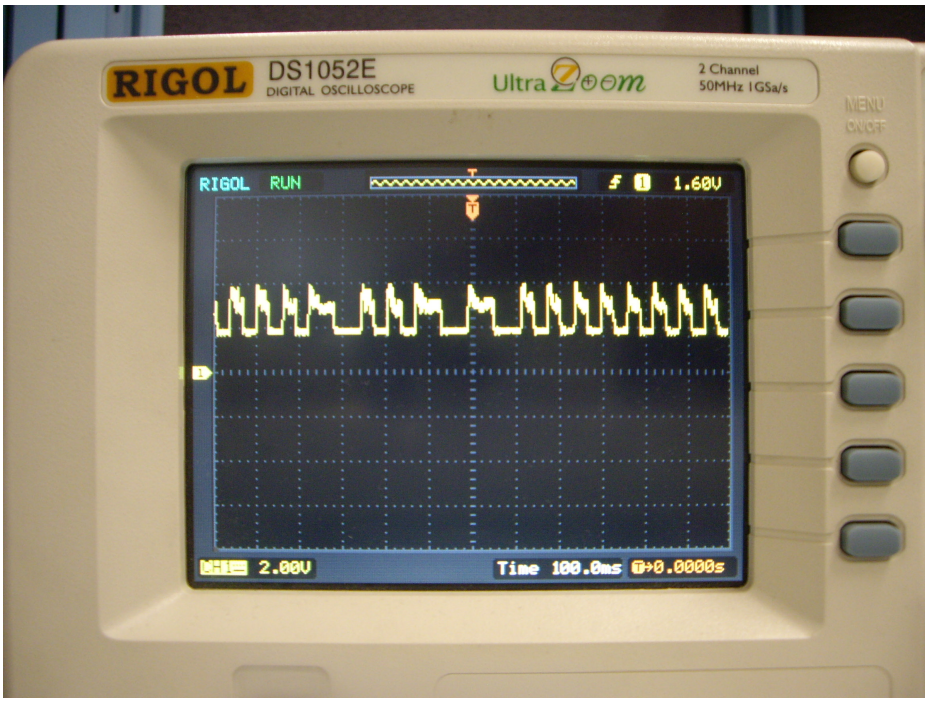
Arriba una senyal de 2000Hz per tant, amb un període de 5ms. Dissenyarem el detector de màxims perquè cada 5ms estigui preparat per rebre un nou pic. Fixem en l'equació el valor de 5ms i un condensador de 1μf. Resolem l'equació

[redacted]

5.7.4Schmitt Trigger

5.7.4.1Introducció

Si observem la senyal després de passar pel detector de màxims, observem que els intervals que transmeten un bit de valor 1, cada vegada s'assemblen més a un pols. Ara no tenim una senyal que oscil·la, sinó que tenim una senyal que s'assembla a una dent de serra. El detector de màxims eleva la senyal i després el condensador, va deixant anar la senyal poc a poc, fins que el següent pic la torna a elevar. Podem observar aquest fenomen en la imatge 23:



Imatge 23 Senyal a la sortida del detector de màxims

Per aconseguir que la senyal, passi de una dent de serra, a una senyal, on els intervals de senyal estiguin representats per un sol pols, dissenyarem un circuit Schmitt Trigger.

Schmitt Trigger, és un nom genèric que reben un classe de circuits que tenen un feedback positiu. Aquets circuits, són capaços de retenir el seu valor fins que l'entrada canvia un valor suficient per forçar un canvi en la sortida. En una configuració no inversora, quan el valor de

l'entrada passa d'un cert llindar, el valor de la sortida és alt, quan el valor d'entrada baixa d'un cert llindar la sortida té un nivell baix. Si baixant la senyal d'entrada, aquesta queda entre aquest dos nivells, la tensió a la sortida es manté alta

5.7.4.2Configuració Schmitt trigger no simètrica

Hem decidit que utilitzarem una configuració no simètrica Schmitt Trigger que utilitza amplificadors operacionals. Ens interessa aquesta configuració perquè volem decidir tan el valor del llindar de nivell alt, com el llindar de valor baix. A més, aquesta configuració no necessita de polaritat negativa i podem connectar directament un dels ports del amplificador operacional al terra. En la figura 18 es mostra aquesta configuració.

Volem que el nostre voltatge del llindar de nivell alt sigui de 3.2V i que el voltatge del llindar baix sigui de 3.0V. Per tant resolem les següents equacions que caracteritzen aquest model:

(Equació 34) $$R_{TOT} = \frac{R1 \cdot R_{FB}}{R1 + R_{FB}}$$

(Equació 35) $$V(llindar\ superior) = \frac{V \cdot R2}{R2 + R_{TOT}}$$

(Equació 36) $$R_{TOT} = \frac{R2 \cdot R_{FB}}{R2 + R_{FB}}$$

(Equació 37) $$V(llindar\ superior) = \frac{V \cdot R_{TOT}}{R1 + R_{TOT}}$$

Els resultats després de resoldre l'equació són:

R1= 3KΩ

R2=5KΩ

RFB=45KΩ

Per comprovar que hem calculat els valors dels components correctament, utilitzem les equacions que caracteritzen aquesta configuració i en busquem els valors dels llindars

Calculem el llindar superior:

(Equació 38) $$R_{TOT} = \frac{R1 \cdot R_{FB}}{R1 + R_{FB}}$$

(Equació 39) $$R_{TOT} = \frac{R1 \cdot R_{FB}}{R1 + R_{FB}} = \frac{3K \cdot 45K}{3K + 45K} = 2.812{,}5$$

(Equació 40) $$V(llindar\ superior) = \frac{V \cdot R2}{R2 + R_{TOT}} = \frac{5 \cdot 5K}{5K + 2{,}8125K} = 3{,}2V$$

Calculem el llindar inferior:

(Equació 41) $$R_{TOT} = \frac{R2 \cdot R_{FB}}{R2 + R_{FB}}$$

(Equació 42) $$R_{TOT} = \frac{R2 \cdot R_{FB}}{R2 + R_{FB}} = \frac{5K \cdot 45K}{5K + 45K} = 4500$$

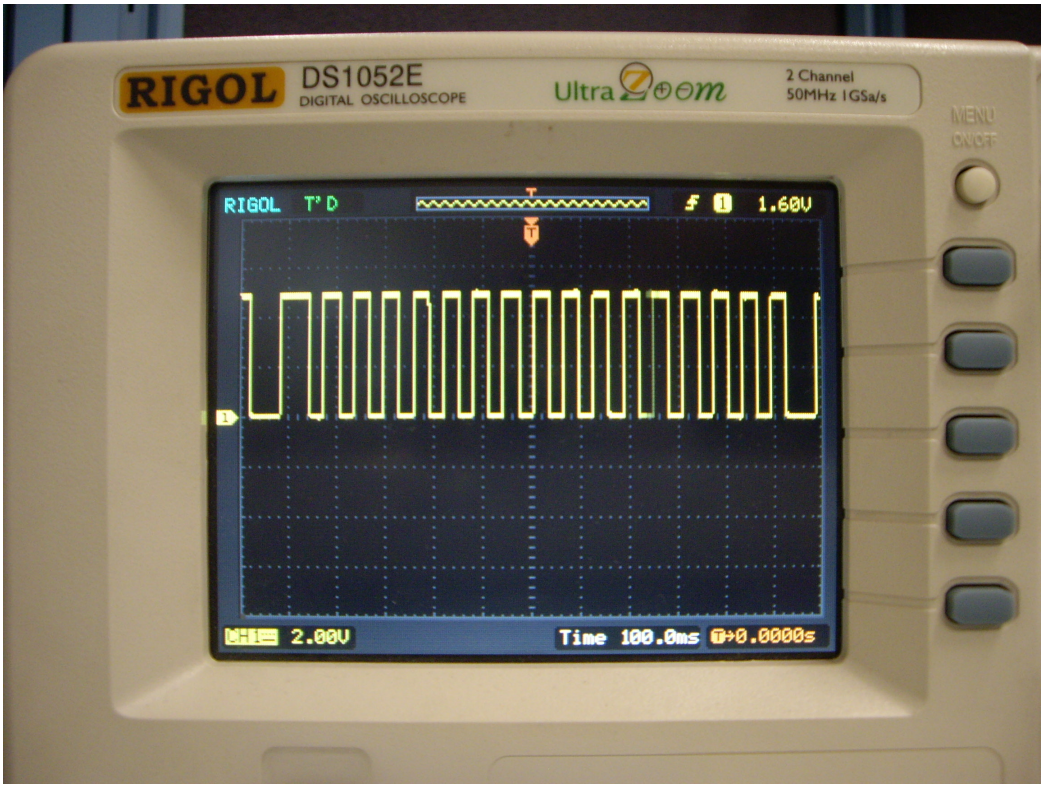
(Equació 43) $$V(llindar\ superior) = \frac{V \cdot R_{TOT}}{R1 + R_{TOT}} = \frac{5 \cdot 4500}{3K + 4500} = 3V$$

Podem veure l'esquemàtic de la configuració del nostre circuit Schmitt Trigger en la figura 19:

[redacted]

5.7.5Conclusions

Per comprovar que el disseny és correcte i comprovar que hem complert amb el nostre objectiu, connectarem totes les etapes que hem dissenyat. Comprovarem com és la senyal amb un oscil·loscopi. Injectem la senyal a l'entrada del circuit i comprovarem com és la senyal a la sortida



Imatge 24 Senyal a la sortida del circuit

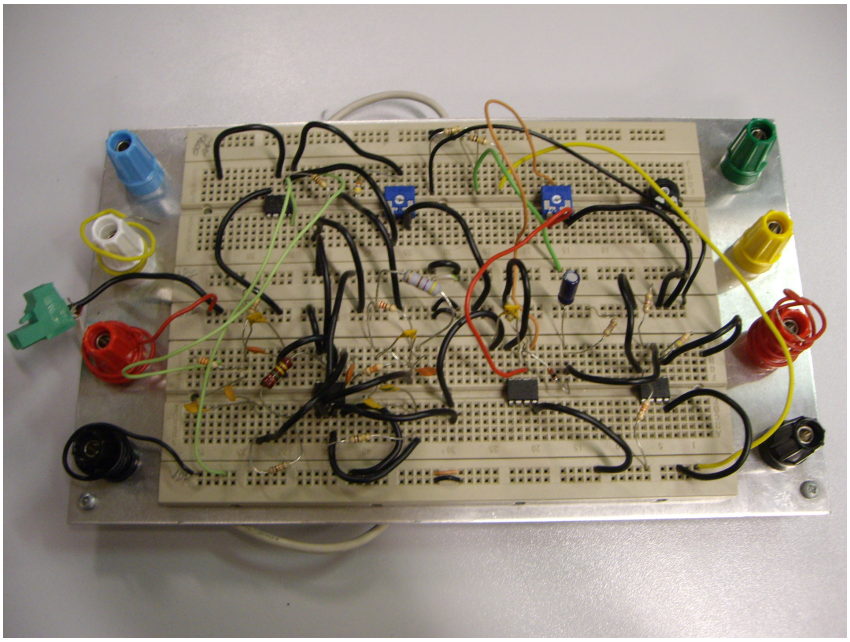
Com veiem en la imatge 24, la senyal a la sortida, és una senyal tren de polsos. Per tant, la senyal compleix amb tots els requisits que hem establert en els objectius.

5.8 Conclusions

En la placa de forats que hem utilitzat per desenvolupar, tenim totes les etapes del circuit posicionades. En la sortida del circuit tenim un tren de polsos, o sigui que tenim la senyal preparada perquè la informació pugui ser interpretada per un microcontrolador. Gràcies al compliment de tots els objectius parcials podem donar per finalitzada la part del disseny del hardware de recepció.

Tenim un circuit on a l'entrada hi ha la senyal que arriba per la xarxa telefònica i a la sortida tenim un senyal tren de polsos

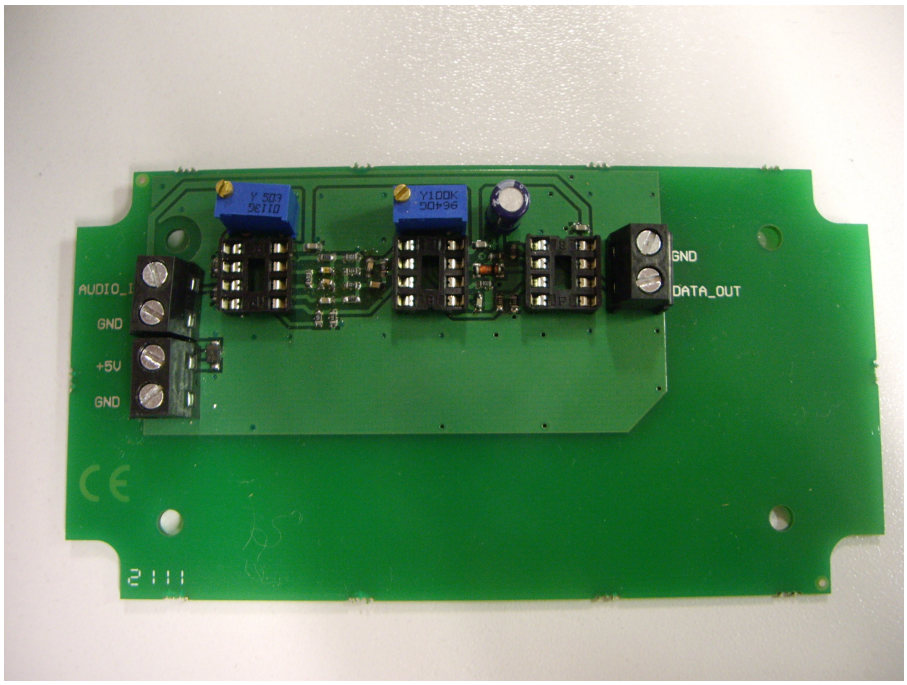
En la imatge 25 podem observar totes les etapes sobre la placa de forats. En l'annex 7 podem trobar un esquemàtic de totes les etapes.



Imatge 25 Circuit sobre la placa de forats

Ara que ja hem validat el disseny del hardware, posarem el nostre disseny sobre un suport més estable. Utilitzarem una placa PCB i components SMD amb la mesura del possible. Per fer arribar aquesta senyal fins el microcontrolador connectem un cable de la sortida del circuit fins al microcontrolador

En la imatge 26 es pot observar el resultat final. En l'annex 8 s'inclou un esquema on s'indica la posició de tots els components sobre de la placa.



Imatge 26 Circuit amb components SMD

6 Receptor (Software)

6.1 Introducció

Tenim un equip que ha creat un missatge gràcies a un brunzidor. Aquest missatge, ha viatjat a través de la xarxa telefònica. un cop ha arribat al seu destí un petit circuit ha sigut l'encarregat de transformar la senyal perquè més tard un microcontrolador la pugui interpretar.
Per acabar el desenvolupament d'aquest projecte, només ens fa falta que un microcontrolador descodifiqui la senyal i enviï la informació del missatge fins un PC.

L'objectiu d'aquesta part és: programar un microcontrolador perquè capti les dades que li arriben mitjançant un port d'entrada, les analitzi i enviï la informació a través del port USB

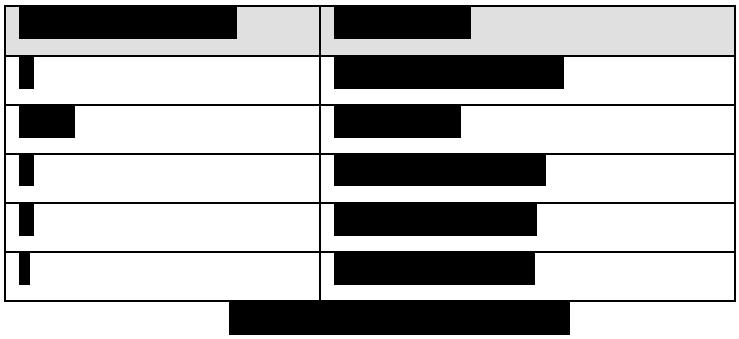
Per poder aconseguir l'objectiu d'aquesta part, l'hem dividit en objectius parcials de més fàcil compliment . La suma dels objectius parcials ens farà resoldre l'objectiu de desenvolupar un software de recepció
A continuació detallem els objectius parcials:

- Decidir quin port utilitzem pel microcontrolador
- Convertir un tren de polsos en una seqüència de bits
- Descodificar la senyal a codificació Manchester
- Passar de codificació Manchester a codificació natural
- Enviar la informació pel port USB

Aquesta és la última part del projecte. Si aconseguim complir amb aquests objectius haurem acabat amb el desenvolupament de la nostra aplicació.

Abans de continuar amb el desenvolupament de l'aplicació, creiem necessari recordar com està organitzada la informació que rebem. Hem de desenvolupar el software de recepció tenint en compte com estan organitzades les dades.

Rebem la informació en paquets de bytes. Els paquets, estan organitzats com s'indica en la taula 4:



Recordem que els bytes que rebem són amb codificació Manchester. Tots els bytes estan fets de 8 bits i cada bit té una durada de 40ms.

Per organitzar les funcions que volem dissenyar en aquesta part hem creat l'esquema de la figura 20

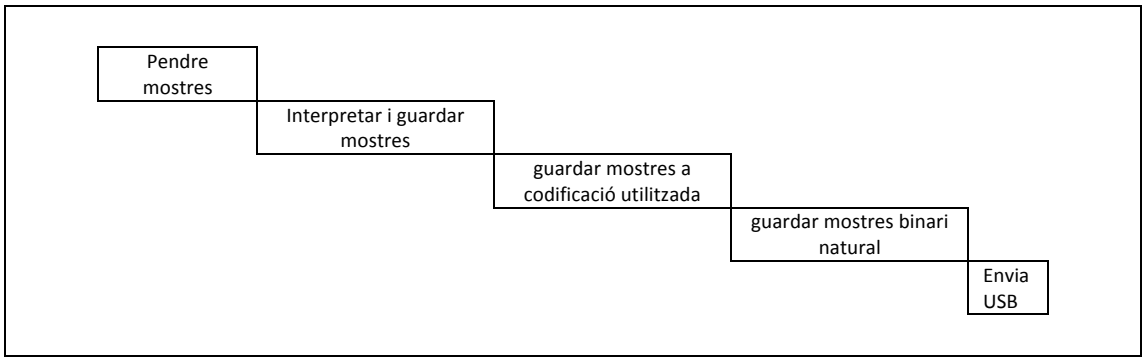
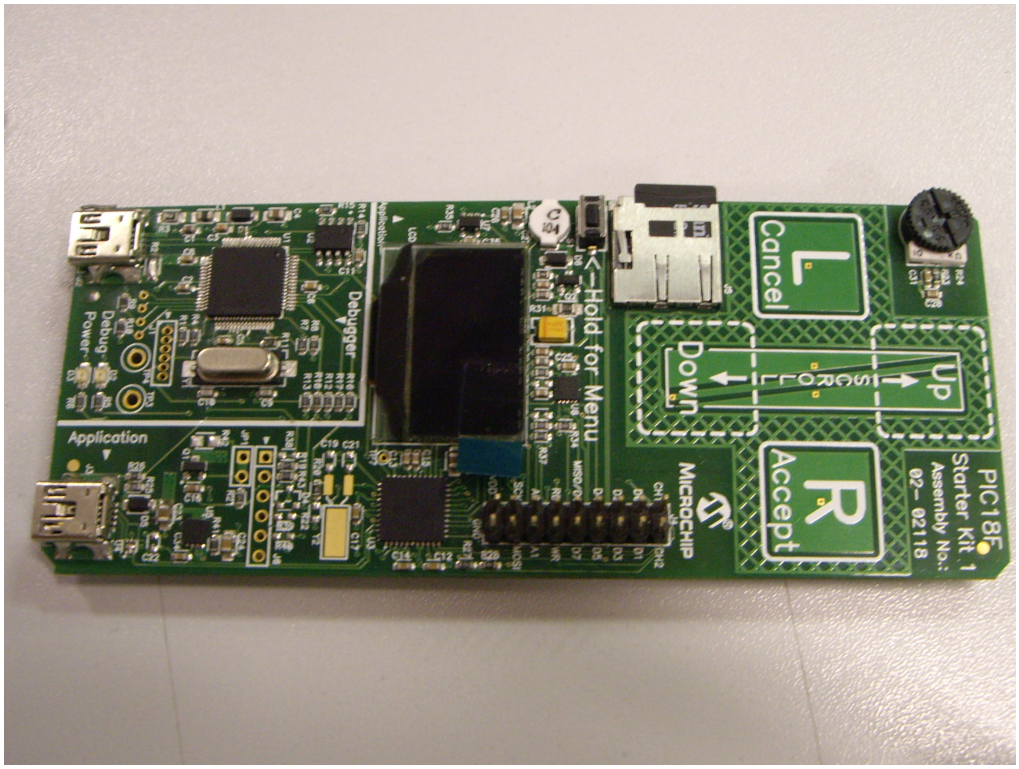


Figura 20 Organització funcions recepció

6.2 Material de desenvolupament

Abans de començar a desenvolupar les solucions pels objectius parcials, començarem amb una petita descripció sobre el material que utilitzarem per desenvolupar aquesta part.
Alguns dels materials que s'utilitzen en aquesta part, són els mateixos que s'han utilitzat durant el desenvolupament del software de transmissió, per tant, no els descriurem detalladament

Desenvoluparem el projecte sobre d'una placa d'avaluació, tal i com hem fet per l'emissor, en aquest cas, utilitzarem un altre model. En aquesta part desenvoluparem sobre la Demo board "PIC18F starter Kit 1" (Imatge 21). Aquesta placa utilitza el microcontrolador PIC18F46J50 de Microchip. En l'annex s'inclou el datasheet amb la informació detalla sobre la placa Demo i sobre el PIC.
La placa disposa de petits dispositius de hardware que ens ajudaran en el desenvolupament de la nostra aplicació, com per exemple: pantalla OLED, un polsador, etc. Aquets dispositius, es poden fer servir per detectar errors, entre d'altres coses. La placa, també disposa d'un port USB, és per això que hem escollit aquesta Demo board.
Amb aquesta Demo board, no fa falta gravador extern per gravar el codi dins del PIC. El gravador està integrat dins de la placa d'avaluació. Per gravar el codi dins del PIC, només hem de connectar la placa al PC a través del port USB.



Imatge 27 Demo board per la recepció

Desenvoluparem el software de la part de recepció utilitzant el llenguatge C. Editarem aquest codi utilitzant MPLAB.

6.3 Hardware

6.3.1 Introducció

Per poder desenvolupar el software que descodifiqui la senyal, el primer que necessitem, és que la senyal amb la informació arribi físicament fins al pin d'entrada del microcontrolador.

L'objectiu d'aquest apartat és: crear una connexió que faci que la informació que hi ha a la sortida de la placa, en forma de tren de polsos, viatgi fins al microcontrolador.

Per complir l'objectiu d'aquest apartat, el primer que fem, és buscar un port del PIC que el puguem configurar per poder-lo utilitzar per rebre la senyal. Observem que en la Demo board hi ha una tira doble de pins mascles per fer-hi connexions. Aquets pins corresponen els ports RD del microcontrolador.

Consultem el datasheet del PIC18F46J50 en l'apartat on es descriuen les característiques dels ports RD, Consultem la imatge 28. Podem trobar el datasheet de la demo board en l'annex 9 i en l'annex 10 el datasheet del microcontrolador que utilitza.

TABLE 1-4: PIC18F4XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

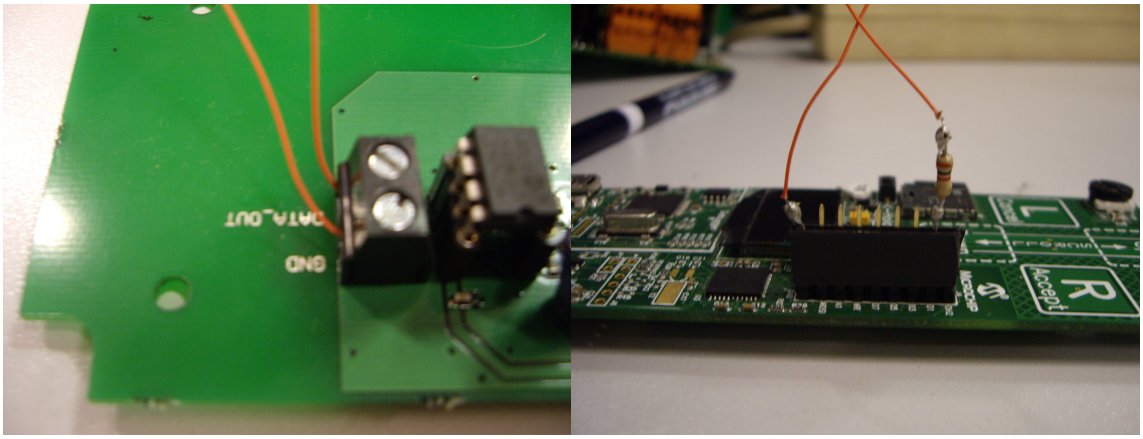
Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTD is a bidirectional I/O port.
RD0/PMD0/SCL2	38	38			
RD0			I/O	ST	Digital I/O.
PMD0			I/O	DIG	Parallel Master Port data.
SCL2			I/O	DIG	I^2C™ data input/output.
RD1/PMD1/SDA2	39	39			
RD1			I/O	ST	Digital I/O.
PMD1			I/O	DIG	Parallel Master Port data.
SDA2			I/O	DIG	I^2C data input/output.
RD2/PMD2/RP19	40	40			
RD2			I/O	ST	Digital I/O.
PMD2			I/O	DIG	Parallel Master Port data.
RP19			I/O	DIG	Remappable Peripheral Pin 19 input/output.
RD3/PMD3/RP20	41	41			
RD3			I/O	ST	Digital I/O.
PMD3			I/O	DIG	Parallel Master Port data.
RP20			I/O	DIG	Remappable Peripheral Pin 20 input/output.
RD4/PMD4/RP21	2	2			
RD4			I/O	ST	Digital I/O.
PMD4			I/O	DIG	Parallel Master Port data.
RP21			I/O	DIG	Remappable Peripheral Pin 21 input/output.
RD5/PMD5/RP22	3	3			
RD5			I/O	ST	Digital I/O.
PMD5			I/O	DIG	Parallel Master Port data.
RP22			I/O	DIG	Remappable Peripheral Pin 22 input/output.
RD6/PMD6/RP23	4	4			
RD6			I/O	ST	Digital I/O.
PMD6			I/O	DIG	Parallel Master Port data.
RP23			I/O	DIG	Remappable Peripheral Pin 23 input/output.
RD7/PMD7/RP24	5	5			
RD7			I/O	ST	Digital I/O.
PMD7			I/O	DIG	Parallel Master Port data.
RP24			I/O	DIG	Remappable Peripheral Pin 24 input/output.

Imatge 28 Descripció PIC18 entrades i sortides

Després de consultar el datasheet, decidim que utilitzarem el port RD1. El port RD1 es pot configurar com una entrada, per tant compleix amb els nostres requisits

Ara, només ens fa falta fer la connexió física. Amb un cable connectem el pin de la Demo board que correspon al pin RD1 amb la sortida de la placa i la sortida de terra amb el terra de la Demo board.

En la imatge 29 en podem observar la connexió



Imatge 29 Connexió a la demo board

6.3.2 Conclusions

Farem una comprovació molt simple per assegurar-nos que la connexió funciona correctament. Amb l'ajuda d'un oscil·loscopi, comprovarem que hi ha la mateixa senyal a la sortida del nostre circuit i en el port RD1.

Injectem una senyal i observem la senyal en aquests dos punts.

La senyal en els dos punts és exactament la mateixa, per tant podem donar per complert l'objectiu d'aquesta part. La senyal ja pot viatjar fins al pin d'entrada del microcontrolador gràcies a la connexió que hem creat en aquest apartat

6.4 De tren de polsos a seqüència de bits

6.4.1 Introducció

El microcontrolador rep un tren de polsos a través del port RD1. L'objectiu d'aquesta part, és crear el software necessari perquè el microcontrolador sigui capaç de convertir aquesta senyal, en una seqüència de bits.

Per poder distingir els 4 casos seguirem els passos següents:

- Crearem una funció que mostregi periòdicament la senyal que arriba pel port RD1
- Analitzarem les dades mostrejades i prendrem una decisió sobre quin tipus d'interval hem detectat.

6.4.2Timer0

Per dissenyar la funció que utilitzarem per prendre les mostres, utilitzarem el Timer0. El Timer0 el configurarem igual que en el software de transmissió, per tant, no entrarem en gaires detalls. El Timer0, queda configurat així

```
TOCON = 0b01001000
```

El Timer0 actuarà com un comptador de 8 bits. Això vol dir, que el Timer0 farà increments de 1. Cada vegada que arribi als 255 increments (2 (codi binari)^8 bits=255 increments), activarà un bit de flag.

En aquest cas, és molt important tenir en compte que el PIC d'aquesta part funciona a 48MHz (La part d'emissió a 4MHz). En les funcions que utilitzem el Timer0, hem de tenir en compte que el PIC funciona 12 vegades més ràpid.

6.4.3Char MostraTemps (int x)

L'objectiu d'aquesta funció, és comprovar quin valor hi ha al port RD1 cada cert temps. Recordem que aquesta entrada, només pot prendre el valor de 1 o 0. Més endavant, utilitzarem aquestes mostres per decidir com és el interval de senyal que hem rebut.

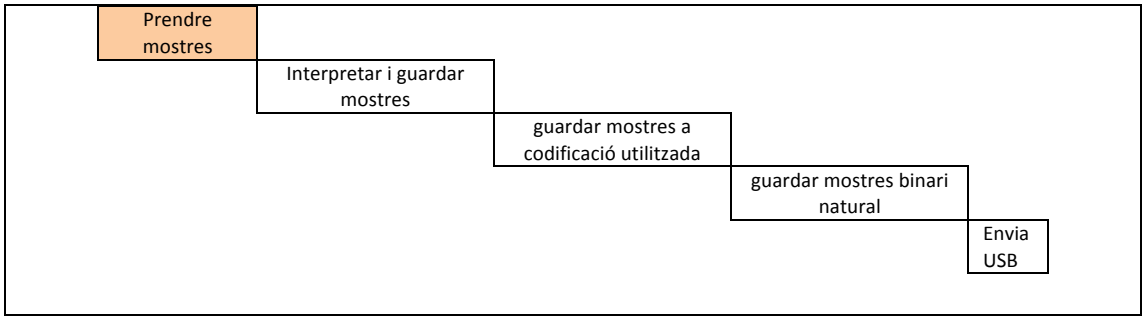


Figura 21 Organització funcions recepció. Desenvolupant prendre mostres

Aquesta funció té un paràmetre d'entrada:

- int x: indica cada quant temps, en mil·lisegons, volem prendre una mostra de la senyal

A partir del paràmetre d'entrada, calculem quantes activacions de flag de Timer0 tindrem durant els mil·lisegons que hem escollit.

Podem observar que aquesta funció, utilitza el mateix mètode que s'utilitza en el software de recepció per calcular les activacions del flag de Timer0.

```
temps=(12000000/255000)*x;
```

La diferència respecte a l'apartat de transmissió és el fet que hem multiplicat per 12 el temps que hem d'esperar, perquè el PIC que estem utilitzant funciona 12 vegades més ràpid.

Esperem al número d'activacions que acabem de calcular. Un cop han passat, prenem una mostra de la senyal d'entrada.

Aquesta funció retorna un paràmetre:

- return mostra: la funció retorna el valor de la mostra de senyal

1.1.1. Conclusions Char MostraTemps (int x)

Per poder continuar, comprovarem que aquesta part de codi funciona correctament. Per fer-ho utilitzarem els dos canals de l'oscil·loscopi.

Col·loquem la funció dins d'un bucle infinit. Passem un paràmetre d'entrada perquè agafi una mostra cada 4 ms. Cada vegada que prenem una mostra, fem que tregui el valor per un port.

Connectem un canal del oscil·loscopi a l'entrada del microcontrolador. Per aquest canal, veurem la senyal a mostrejar.

Connectem el segon canal del oscil·loscopi al port que estem utilitzant temporalment per veure els valors de les mostres.

Configurem l'oscil·loscopi per poder veure els dos canals simultàniament. Comencem la simulació de la transmissió i observem la pantalla de l'oscil·loscopi.

Observem que les senyals queden sobreposades, hi ha un petit retard en el segon canal respecte el primer canal de uns microsegons . Això vol dir, que els valors de les mostres són correctes, per tant la funció compleix amb el seu objectiu

6.4.4char WhatBitManchester (int x, int y, int z)

En aquest apartat, volem desenvolupar un funció que utilitzi la funció del apartat anterior per prendre mostres periòdicament. A partir de les mostres, la funció ha de prendre una decisió sobre com és la informació que ha rebut. A partir de la decisió, crearà una seqüència de bits i la guardarà en la memòria del microcontrolador. És molt important, que aquesta funció sigui capaç de distingir els 4 tipus diferents de senyal que poden arribar fins aquest punt

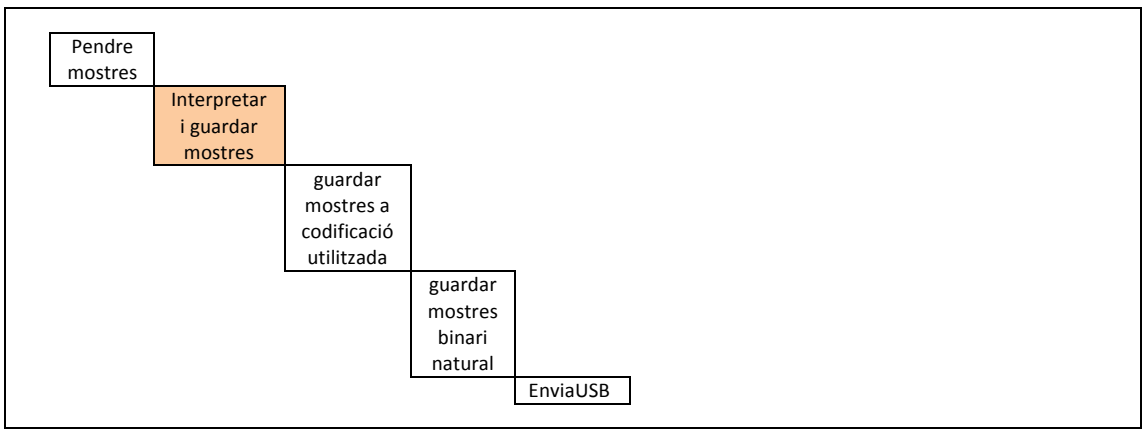
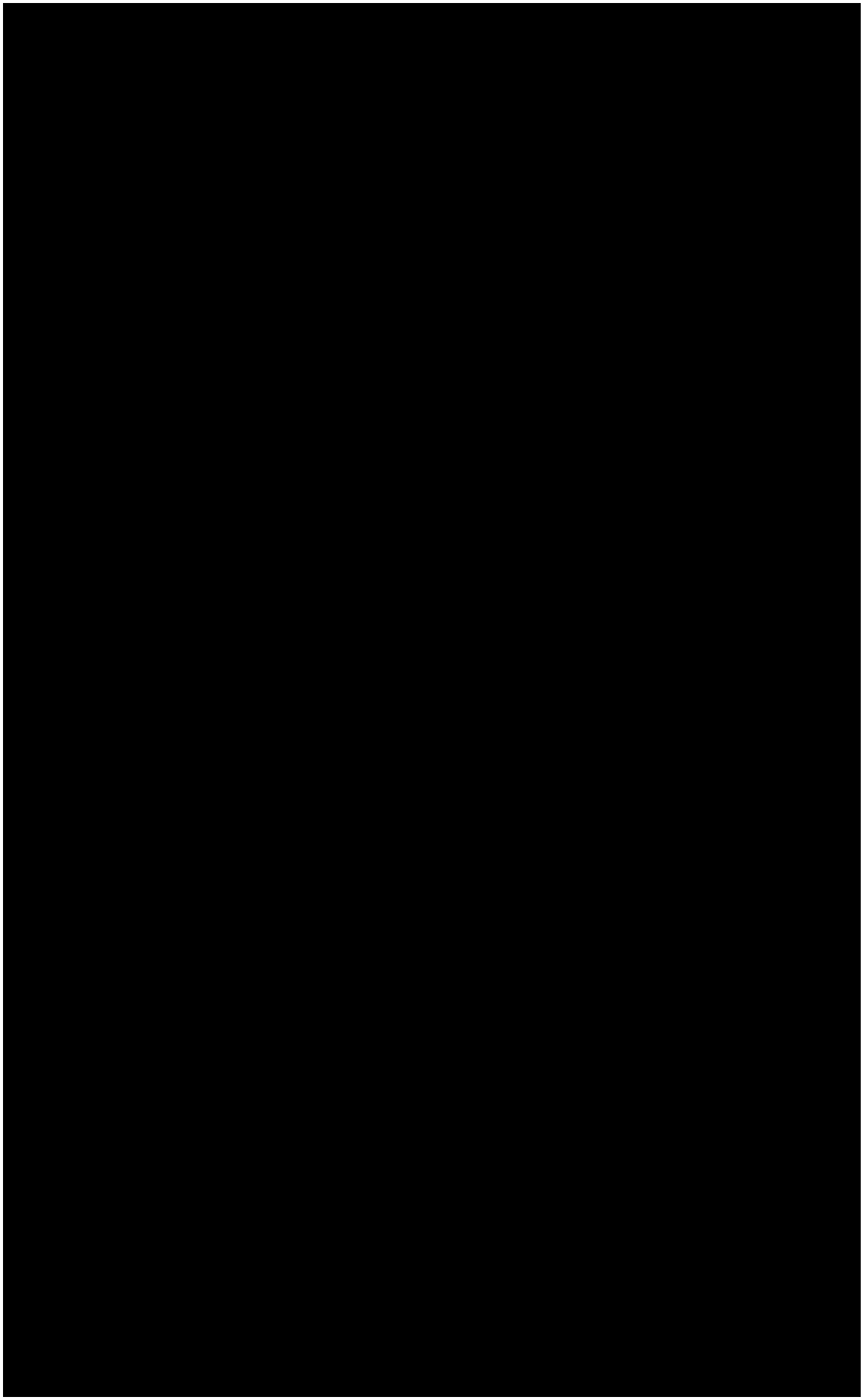


Figura 22 Organització funcions recepció. Desenvolupant interpretar i guardar mostres



Aquesta funció té tres paràmetres d'entrada:

- int x: indica el número mínim de mostres iguals i seguides que hem de tenir per decidir que hem rebut un bit

- int y: indica el número mínim de mostres iguals i seguides que hem de tenir per decidir que hem rebut dos bits iguals seguits

- int z: indica el número mínim de mostres iguals i seguides que hem de tenir per decidir que la transmissió d'un paquet ha acabat o que hi ha hagut un error

El primer que fa aquesta funció, és a través de la funció que hem descrit anteriorment agafa una mostra periòdicament cada 4ms. Cada vegada que tenim una mostra nova, comparem la mostra actual, amb la mostra anterior.

Tenim una variable que incrementa 1, cada vegada que la mostra actual és igual que l'anterior. Si aquest valor supera el valor de int z, vol dir que hi ha un error o que simplement s'ha acabat la transmissió del paquet. Com a conseqüència torna el valor de la variable b.
Si la mostra actual i la mostra anterior no coincideixen, vol dir que hi hagut un canvi d'estat en la senyal. Comprovem el marcador que ens indica el nombre de mostres seguides i iguals que hem tingut. Primer comprovem que no sigui superior a int y, que vol dir que a la senyal d'entrada han passat dos bits iguals seguits.
Si no és així, comprovem que el valor del marcador de mostres iguals i seguides, no sigui més gran que int x, que vol dir, que hem rebut un bit.

Aquest procés és el mateix tan si estem rebem bits de valor 0 o bits de valor 1. Cada vegada que detectem un dels 4 tipus de símbols possible, desplaçarem els bits d'un byte 2 posicions cap a l'esquerra i omplirem els dos espais que hem creat. Aquest valor canvia en funció del tipus de senyal detectat a l'entrada del microcontrolador. Per conveni hem establert que seguirem la taula 5:

Símbols que rebem a l'entrada	Símbols que retorna la funció
0	00
00	10

1	01
11	11

Taula5 Conversió de símbols

Aquest procés es repeteix 4 vegades, fins que omplim els 8 bits d'un byte. Anirem guardant tots els bytes en el microcontrolador, fins que no detectem que la transmissió del paquet ha acabat.

Aquesta funció té un paràmetre de sortida:
- **return b:** Aquesta funció retorna un byte cada vegada que ha omplerts els 8 bits.

1.1.1. Conclusions

Per comprovar que la funció compleix amb els requisits, col·loquem la funció dins d'un bucle infinit. Modifiquem la funció perquè en comptes de guardar els símbols, enviï els que va afegint per crear els bytes directament per un port.

Connectem un canal de l'oscil·loscopi a l'entrada del microcontrolador. Per aquest canal veurem la senyal a la qual li agafem les mostres
Connectem el segon canal de l'oscil·loscopi al port que estem utilitzant temporalment per treure els valors de les mostres.
Configurem l'oscil·loscopi per poder veure els dos canal simultàniament

Comencem la simulació de la transmissió i observem la pantalla de l'oscil·loscopi.

Observem que, les senyals queden sobreposades, si observem el port de sortida del bit de menys pes, veiem que hi ha un flanc cada vegada que passem de detectar un bit a detectar dos bits seguits.

Després de fer aquestes observacions, veiem que som capaços de convertir una senyal de tren de polsos, que arriba físicament a través d'un pin del PIC a una seqüència de bits, que queda emmagatzemada dins del microcontrolador. Per tant, podem arribar a la conclusió que hem complert amb l'objectiu d'aquesta part.

6.5 Descodificar a Manchester

6.5.1 Introducció

En aquest moments, tenim una seqüència de bits, que porta la informació sobre un equip, guardada en la memòria del microcontrolador. La senyal que arriba el microcontrolador està codificada amb Manchester. Per crear la seqüència de bits hem alterat aquesta codificació. Hem afegit bits que tot i que ens han resultat útils per crear la seqüència, no aporten informació extra al missatge.

L'objectiu de la funció que crearem en aquest apartat, és convertir els bits que tenim guardats en la memòria, en bits que reflecteixin com era la senyal quan ha arribat al pin d'entrada del PIC.

6.5.2 void BackToManchester (int frase)

En aquests moments, tenim la informació guardada en una matriu de bytes. Agafarem un byte i analitzarem els bits per parelles. El bit 7 amb el bit 6, el bit 5 amb el bit 4, etc. Fins que analitzem tots els bits. Analitzarem les parelles i les guardarem en una altre matriu de bytes tal i com s'indica en la taula 6.
Repetirem aquest procés fins haver analitzat tots els bytes de la matriu.

Parella de bits de la seqüència	Símbols en codificació Manchester
00	0
10	00
01	1
11	11

Taula6 Conversió de símbols Manchester

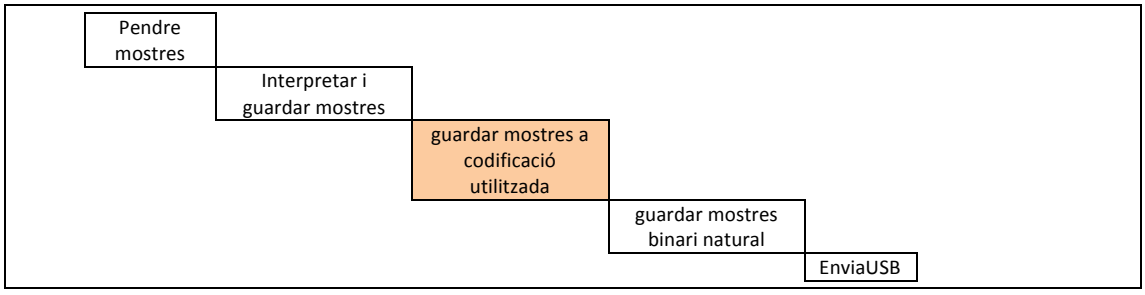
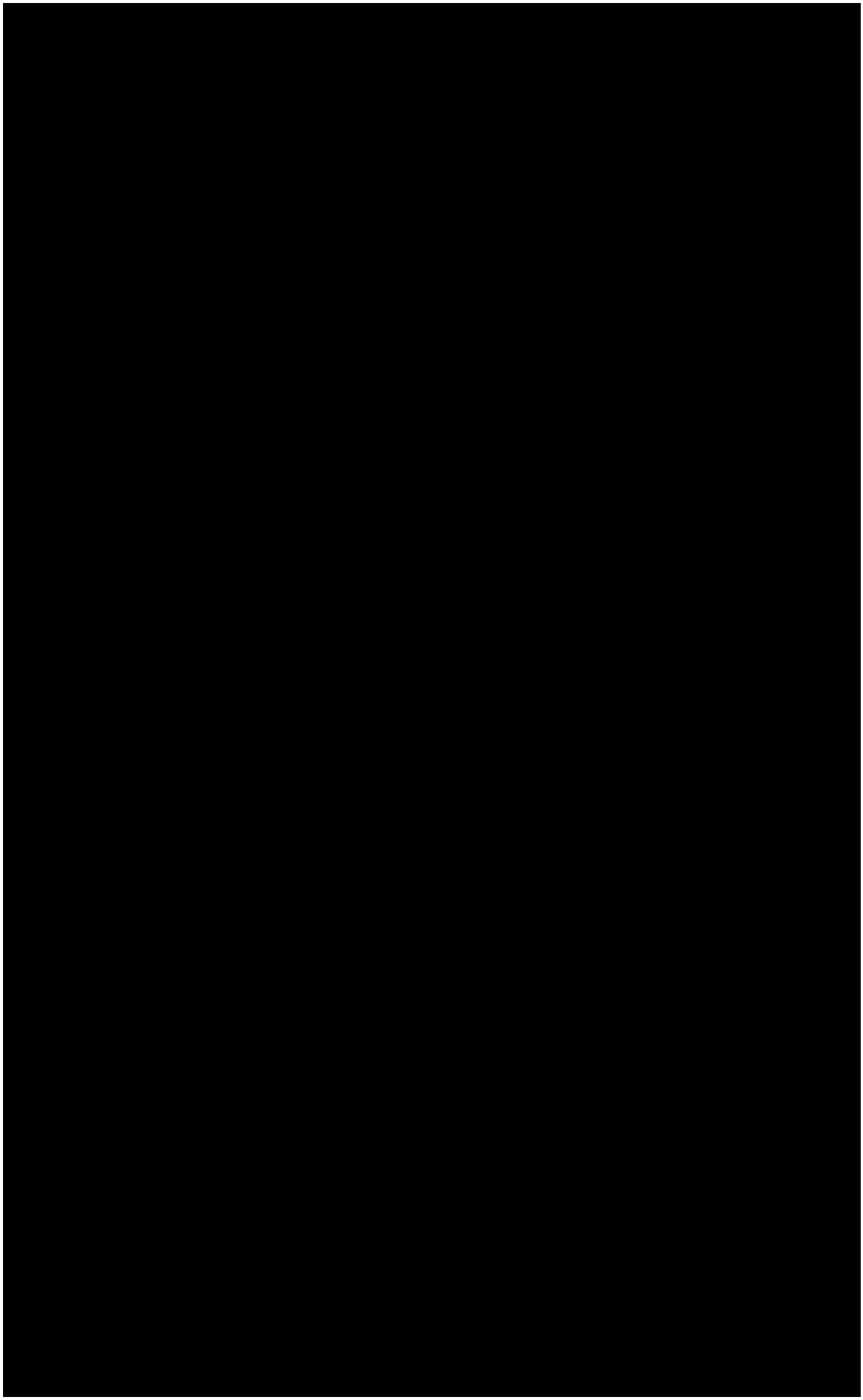


Figura 23 Organització funcions recepció. Desenvolupant guardar mostres a codificació utilitzada



Aquesta funció té un paràmetre d'entrada

- int frase: Indica el número de bytes que hem d'analitzar

La funció, entra dins del primer byte de la matriu i analitza la primera parella de bits (bit 7 i bit 6). Realitza una AND lògica bit a bit amb el valor binari 1100 0000. Quan realitzem aquesta operació, el resultat sempre serà el valor dels bits 7 i 6 del byte que estem analitzant, seguit de 0's.

Un cop tenim aquest resultat el comparem per saber a quin valor correspon en codificació Manchester. A continuació es mostra la taula 7 on podem observar la conversió segons el resultat de l'AND bit a bit

Byte resultat AND bit a bit	Símbol Manchester
0000 0000	0
1000 0000	00
0100 0000	1
1100 0000	11

Taula7 Resultats AND i conversió a símbol Manchester

Guardem el resultat de la nostra conversió en una matriu de bytes. Les dades es comencen a guardar en els bits de menys pes d'un byte. A mesura que es va omplint es desplaça la posició dels bits a la dreta, fins a tenir un byte ple. Quan el byte té els 8 bits ocupats es guarda la informació en un altre byte.

El procés es va repetint fins que haguem analitzat i guardat tota la matriu de bytes que tenim guardada.

6.5.3Conclusions

Per comprovar el bon funcionament d'aquest fragment de codi i validar la funció, utilitzarem la opció que ofereix l'editor MPLAB per poder debugar el codi.

Posem un breakpoint en el moment de la funció on es guarda en la memòria el primer byte amb codificació Manchester.
Comencem una simulació de transmissió i el programa es para allà on hem deixat el breakpoint. A través del editor MPLAB, visualitzem la variable on es guarden els bytes.

Segons les nostres especificacions la funció s'hauria de comportar així:
Els dos primer bytes que ha d'analitzar la funció són
- 01 00 01 00
- 01 00 01 00
Després de passar per la funció el resultat hauria de ser un byte de valor
- 1010 1010

Observem quin és el valor que ha guardat la funció en la variable. El valor teòric és el mateix que observem en la pràctica. Hem aconseguit que aquesta funció converteixi la seqüència de bits que hem capturat a codificació Manchester un altre cop. Per tant, podem concloure que hem complert amb els objectius d'aquesta part.

6.6 De codificació Manchester a codificació natural

6.6.1 Introducció

Aquest apartat és molt similar al apartat anterior. Aquesta vegada però, disposem d'una seqüència de bits en codificació Manchester guardats en la memòria del PIC. La codificació Manchester ens ha sigut molt útil alhora de transmetre la informació, perquè així ens ha fet més fàcil el sincronisme. Com que ja no ens fa falta aquest avantatge, descodificarem la seqüència de bits a una codificació natural.

L'objectiu d'aquesta funció serà descodificar tots els bytes de la matriu que hem creat amb la funció anterior a codificació natural. La funció ha de guardar la seqüència en una matriu de bytes.

6.6.2 void BackToNormal ()

Aquesta funció analitza tots els bytes de la matriu analitzant els bits de dos en dos. Codifica la informació de Manchester a codificació natural. No entrem amb més detalls perquè funciona exactament igual que la funció anterior, però aquesta vegada utilitzem la taula de conversió 8

Símbol codificació Manchester	Símbol codificació natural
10	1
01	0

Taula8 Conversió de Manchester a Natural

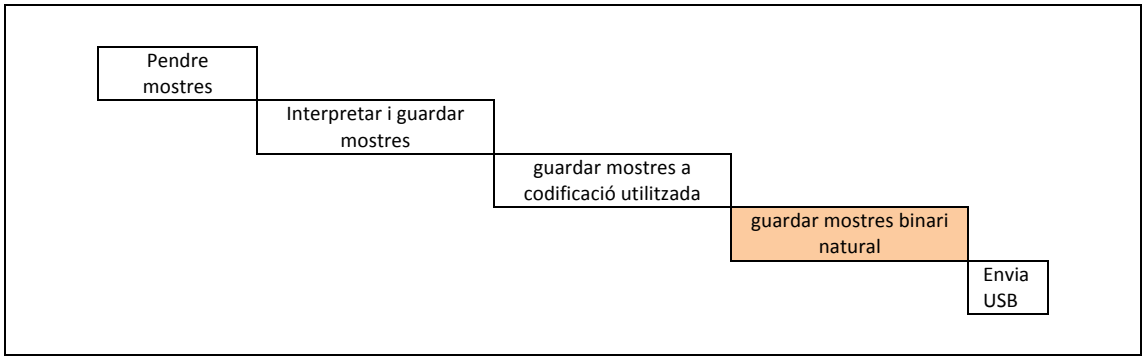


Figura 24 Organització funcions recepció. Desenvolupant guardar mostres binari natural

La funció, entra dins del primer byte de la matriu i analitza la primera parella de bits (bit 7 i bit 6). Realitza una AND lògica bit a bit amb el valor binari 1100 0000. Quan realitzem aquesta operació, el resultat sempre serà el valor dels bits 7 i 6 del byte que estem analitzant, seguit de 0's.

Un cop tenim aquest resultat el comparem per saber a quin valor de la taula 9 correspon en codificació Natural. A continuació, es mostra com es fa aquesta conversió segons el resultat de l'AND bit a bit

Símbl del resultat AND bit a bit	Bit a guardar
1000 0000	1
0100 0000	0

Taula9 Resultat AND i bit a guardar

Guardem el resultat de la nostra conversió en una matriu de bytes. Les dades es comencen a guardar en els bits de menys pes d'un byte, però a mesura que es va omplint es desplaça la posició dels bits cap a la dreta, fins a tenir un byte ple. Quan el byte té els 8 bits ocupats es guarda la informació en un altre byte.

Tot aquest procés, es va repetint fins que haguem analitzat i guardat tots els bits de la matriu de bytes que hem creat amb la funció de l'apartat anterior.

6.6.3Conclusions

Posem un breakpoint en el moment de la funció, on es guarda a la memòria el primer byte amb codificació natural.
Comencem una simulació de transmissió i el programa, es para allà on hem deixat el breakpoint. A través del editor MPLAB, visualitzem la variable on es guarden els bytes un cop els hem convertit a codificació natural.

Segons les nostres especificacions la funció s'hauria de comportar així:
Els dos primer bytes que ha d'analitzar la funció són
-10 10 10 10
- 10 10 10 10
Després de passar per la funció el resultat hauria de ser un byte de valor
- 1111 1111

Observem, quin és el valor que ha guardat la funció en la variable. El valor teòric és el mateix que observem en la pràctica. Hem aconseguit que aquesta funció converteixi la seqüència de bits que hem capturar a codificació natural. Per tant, podem concloure que hem complert amb els objectius d'aquesta part.

6.7 Enviar la informació via port USB

6.7.1 Introducció

El brunzidor d'un equip de JCM ha creat un missatge codificat, aquest missatge a viatjat fins a la part receptora a través de la xarxa telefònica. La part receptora ha transformat la senyal i l'ha descodificat. La informació està guardada en la memòria d'un PIC.

En la memòria del microcontrolador tenim guardada la informació respecte a la configuració del equip de JCM. Perquè aquesta informació sigui útil, pel SAT, l'hem d'aconseguir passar fins a un PC.
Per realitzar aquesta transmissió utilitzarem el port USB de la Demo board i enviarem les dades fins al port USB d'un PC.

L'objectiu d'aquesta part serà doncs, enviar la informació que hi ha guardada en la memòria del microcontrolador a través del port USB fins a un ordinador.

6.7.2 Codi

Farem una descripció del codi que hem desenvolupat perquè la informació viatgi del microcontrolador fins al PC a través del port USB.

Abans de començar amb el desenvolupament farem una puntualització per explicar algunes de les decisions. L'estudi del funcionament del port USB, no és un dels objectius d'aquest projecte, és per això, que les funcions que afecten directament el funcionament del port USB les hem agafat dels exemples que ofereix Microchip per fer demostracions del port USB. A partir d'aquí, hem afegit les funcions que hem desenvolupat en els apartats anteriors per poder aconseguir complir amb l'objectiu d'aquesta part

Com que el codi és molt extens, a continuació només s'inclouen les parts que creiem més importants i en l'annex 11 s'inclou el fitxer sencer.

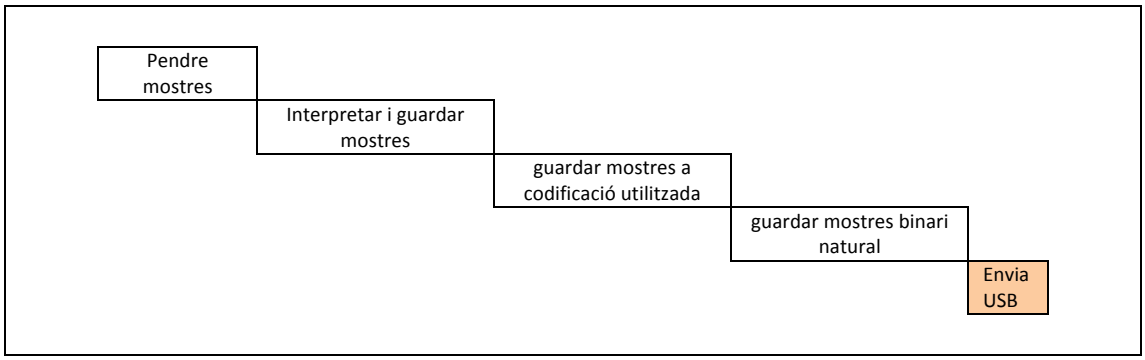


Figura 25 Organització funcions recepció. Desenvolupant Envia USB

```
/**VARIABLES ********************************/
#pragma udata

unsigned char commandIndex = 0;
unsigned int executionTick = 0;
unsigned char DemoIntroState = 0;
unsigned char var = 0;
int a=0, on=1;
char aux [46];
char Codi [23];
char Memoria [92];
int CodiA, CodiB, CodiC=92;
int count;
int estat=0,da=0;
int esperador;
int Dades [23];


/*********************************************/
void main()
{
        int a,b=0,c;

        TRISDbits.TRISD1=1;         // Pin RD1 input
   INTCON2bits.RBPU = 0;       // enable PORTB internal pullups

//oscil∑lador
   OSCCON=0b01010000;

   // Init Timer
   INTCONbits.TMR0IF = 0;        // clear roll-over interrupt flag
   T0CON = 001001000;   // no prescale - increments every instruction clock
   TMR0H = 0;                  // clear timer - always write upper byte first
   TMR0L = 0;
   T0CONbits.TMR0ON = 1;         // start timer

   InitializeSystem();

   while(1)
   {
                // Check bus status and service USB interrupts.
    USBDeviceTasks(); // Interrupt or polling method.  If using polling, must
call
                // Appication related code may be added here, or in the
ProcessIO() function.
     CDCTxService();
                ProcessIO();

          }//end while
}//end main

/*********************************************/
```

```
void ProcessIO(void)
{

  BMA150_REG reg_MSB;
  BMA150_REG reg_LSB;

        char buffer[64];

        if(DemoIntroState == 0xFF)
  {
          BL_CheckLoaderEnabled();
  }

  // User Application USB tasks
  if((SBDeviceState    <    CONFIGURED_STATE)||(USBSuspendControl==1))
return;

        // Soft Start the APP_VDD
  if(AppPowerReady() == FALSE) return;

  DemoIntroduction();


        if (executionTick == 18000)
        {
                switch(commandIndex)
                {
                        case 0:
                                executionTick = executionTick - 100;

                                if (PORTDbits.RD1==0)
                                {
                                        commandIndex ++;
                                }

                                break;

                        case 1:
                                executionTick = executionTick - 100;

                                for (count=0;count<CodiC;count++)
                                {
                               Memoria [count] = WhatBitManchester (4,13,20);
                                }

                                        BackToManchester (CodiC);
                                        BackToNormal ();
                                        count =0;

                                commandIndex ++;

                                for (count=0; count<CodiA+1; count ++)
                                {
                                        Dades [count]=Codi [count];
```

```
                    Dades                    [count]=Dades
[count]&0b0000000011111111 ;
                    }

                    break;

                case 2:
                    executionTick = executionTick - 100;

                    if (CodiA==10)
                    {
                sprintf((char           *)buffer,"          codi:
%X,%X,%X,%X,%X,X,%X,%X,%X,%X,%X \r\n", Dades [0], Dades [1], Dades
[2], Dades [3], ades [4], Dades [5], Dades [6], Dades [7], Dades [8], Dades [9],
Dades [10] );
                    putsUSBUSART(buffer);
                    }

                    else if (CodiA==14)
                    {
                    sprintf((char           *)uffer,"          codi:
%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X%X,%X \r\n",Codi [0], Codi
[1], Codi [2], Codi [3], Codi [4], Codi [5], Codi [6], odi [7], Codi [8], Codi [9],
Codi [10], Codi [11], Codi [12], Codi [13], Codi [14] );
                    putsUSBUSART(buffer);
                    }

                    else if (CodiA==16)
                    {
                sprintf((char           *)buffer,"          codi:
%X,%X,%X,X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X \r\n",Codi [0],
odi [1], Codi [2], Codi [3], Codi [4], Codi [5], Codi [6], Codi [7], Codi [8], Codi
9], Codi [10], Codi [11], Codi [12], Codi [13], Codi [14], Codi [15] , Codi [16]);
                    putsUSBUSART(buffer);
                    }

                    else if (CodiA==22)
                    {
                    sprintf((char           *)buffer,          codi:
%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,X,%X,%X,%X,%X,%
X \r\n",Codi [0], Codi [1], Codi [2], Codi [3], Codi [4], Cdi [5], Codi [6], Codi
[7], Codi [8], Codi [9], Codi [10], Codi [11], Codi [12], Cod [13], Codi [14], Codi
[15] , Codi [16], Codi [17], Codi [18], Codi [19], Codi [20]);
                    putsUSBUSART(buffer);
                    }

                    commandIndex ++;
                    break;

                case 3:
                    executionTick = 0;
                    sprintf(buffer, (const far rom char*)"\r\n");
                    commandIndex = 0;
                    break;
```

```
                    default:
                        commandIndex = 0;
                        break;
                }
        }
        executionTick ++;
        CDCTxService();
}
```

En la primera part del codi, es descriuen totes les variables que utilitzen les funcions del nostre codi.

A continuació, es descriu la funció Main. En el Main, primer es descriu la freqüència d'oscil·lació i es descriu el funcionament del Timer0, que s'utilitza en les funcions que hem dissenyat.
Tot seguit el codi entra en un bucle infinit. Aquest bucle primer crida la funció USBDeviceTasks() i després la funció ProcessIO().
En USBDeviceTasks(), s'executen totes les funcions necessàries per activar els port USB, també en fa el manteniment perquè la connexió no es tanqui fins el final de la transmissió.
En ProcessIO(), hem afegit totes les funcions que hem dissenyat que permeten descodificar la senyal que rebem i guardar-la en la memòria. En aquesta funció també hi col·loquem les sentencies necessàries perquè s'enviï la informació a través del port USB.

En la funció ProcessIO(), primer prenem mostres de la senyal al port d'entrada del microcontrolador, prenem les decisions sobre com és la senyal d'entrada i guardem els resultat en una matriu de bytes. Més tard, convertim aquesta matriu a codificació Manchester per convertir-la a codificació natural i guardar-la en un altre matriu. Per acabar amb aquesta funció, col·loquem tota la informació paquet a paquet (tal com i s'ha creat el missatge d'emissió) sobre la funció sprintf.
La funció sprintf, llança la informació que li passem a través del port USB. La informació que enviem segueix la següent estructura (la llargada pot variar, com hem vist perquè no tots els paquets tenen la mateixa mida):

Codi: Dades [0], Dades [1], Dades [2]…

6.7.3Conclusions

En aquest apartat comprovarem que la funció que acabem de dissenyar compleix el seu objectiu. Per fer-ho, connectem la Demo board a través del port USB a un PC i connectem la resta de peces del projecte per fer una simulació de transmissió

Utilitzarem un programa que inclou Windows, Hyper Terminal. Aquest programa ens mostra per pantalla la informació que rep a través del port USB.

Amb aquesta prova emetem un fitxer de configuració, hem de comprovar que la informació sigui igual a la que rebem i es mostra per pantalla a través de Hyper Terminal.

Comencem amb la simulació. Observem que la informació que rebem és la mateixa que hem enviat per tant podem donar per complert el nostre objectiu. Hem aconseguit enviar la informació a través del port USB.

6.8 Conclusions

Gràcies a la prova que hem realitzat a l'apartat anterior, no només hem comprovat el bon funcionament de l'enviament de dades pel USB, sinó, que hem comprovat el bon funcionament de tota la part de recepció.

La senyal arriba fins al PIC. Un cop allà, al software decideix quina forma té la senyal d'entrada i en guarda la informació. La informació es transforma a codificació Manchester i més tard a codificació natural i es guarda un altre cop dins del microcontrolador. Per últim aquesta informació s'envia a través del port USB.

Gràcies al compliment de tots els objectius parcials dels apartats anteriors, podem afirmar que la part de recepció funciona correctament i que hem complert amb l'objectiu d'aquesta part.

7 Conclusions

Hem dissenyat un sistema dividit en dues parts. La part d'emissió i la part de recepció.

La part d'emissió, està integrada en un equip de JCM. Emetem una senyal sonora a través del brunzidor del equip. El missatge conté la informació sobre la configuració del equip. Aquesta senyal sonora, és un missatge codificat que segueix la codificació Manchester. La informació està organitzada en paquets. Els paquets tenen una llargada variable, però entre paquet i paquet hi ha un silenci de 700ms. A cada paquet hem afegit més informació per fer més fàcil la sincronització i la detecció d'errors.

La senyal viatja a traves de la xarxa telefònica fins a la part receptora.

La part receptora rep el missatge. La senyal, passa per un filtre per eliminar tot el soroll que s'hagi pogut afegir durant l'emissió. La senyal rep un seguit de transformacions fins a convertir-se en un tren de polsos. La senyal arriba a un microcontrolador que descodifica la senyal. un cop s'ha descodificat, la senyal passa per tot un seguit de processos per eliminar tota la informació que hem afegit per fer més segura la transmissió. Un cop tenim tota la informació en el seu estat "natural", obtenim exactament els mateixos paquets amb la mateixa informació que hi havia en la part emissora. Tot seguit enviem la informació a través del port USB.

Un cop la informació arriba al PC donem per finalitzada la comunicació.

Completar tot aquest cicle de comunicació implica que hem superat tots els objectius que planteja el Projecte P.I.T.O.S.

8 Bibliografia

- OBO, Zumbador, 2011. http:// www.obopro2.com/ (consulta 21 de Març de 2011)
- MICROCHIP, MPLAB, 2011. http://www.microchip.com (consulta 29 de Març de 20011)
- WIKIPEDIA, Line Code, 2011. http://en.wikipedia.org/wiki/Line_code (consulta 5 de Abril de 20011)
- WIKIPEDIA, Unipolar encoding, 2011. http://en.wikipedia.org/wiki/Unipolar_encoding (consulta 13 de Abril de 2011)
- WIKIPEDIA, Bipolar encoding, 2011. http://en.wikipedia.org/wiki/Bipolar_encoding (consulta 13 de Abril de 2011)
- WIKIPEDIA, Return-to-zero, 2011. http://en.wikipedia.org/wiki/Return-to-zero (consulta 13 de Abril de 2011)
- WIKIPEDIA, Non-Return-To-Zero, 2011. http://en.wikipedia.org/wiki/Non-return-to-zero (consulta 13 de Abril de 2011)
- WIKIPEDIA, Manchester Code, 2011. http://en.wikipedia.org/wiki/Manchester_code (consulta 13 de Abril de 2011)
- ASCII, ASCII, 2011. http://www.ascii.cl/es (consulta 4 de Maig de 2011)
- APPLE INSIDER, Audio connectors, 2011. http://www.appleinsider.com (consulta 18 de Maig de 2011)
- WIKIPEDIA, Digital Filter, 2011. http://en.wikipedia.org/wiki/Digital_filter (consulta 20 de Maig de 2011)
- WIKIPEDIA, Butterworth filter, 2011. http://en.wikipedia.org/wiki/Butterworth_filter (consulta 20 de Maig de 2011)
- WIKIPEDIA, Chebyshev filter, 2011. http://en.wikipedia.org/wiki/Chebyshev_filter (consulta 20 de Maig de 2011)
- WIKIPEDIA, Eliptic filter, 2011. http://en.wikipedia.org/wiki/Elliptic_filter (consulta 20 de Maig de 2011)
- WIKIPEDIA, Bessel filter, 2011. http://en.wikipedia.org/wiki/Bessel_filter (consulta 20 de Maig de 2011)
- ALL ABOUT CIRCUITS, Voltage dividers, 2011. http:// www.allaboutcircuits.com/vol_1/chpt_6/1.html (consulta 25 de Maig de 2011)
- ALL ABOUT CIRCUITS, Peak Detector, 2011. http://www.allaboutcircuits.com/vol_3/chpt_3/5.html (consulta 28 de Maig de 2011)
- WIKIPEDIA, Schmitt trigger, 2011. http://en.wikipedia.org/wiki/Schmitt_trigger (consulta 2 de Juny de 2011)
- PCB HEAVEN, Schmitt trigger, 2008. http://pcbheaven.com/wikipages/The_Schmitt_Trigger/ (consulta 2 de Juny de 2011)

9 Annexes

9.1 Annex 1. Brunzidor OBO

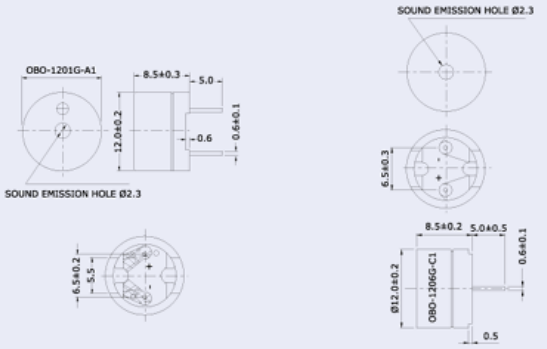
FEATURES

Small size, high sound output

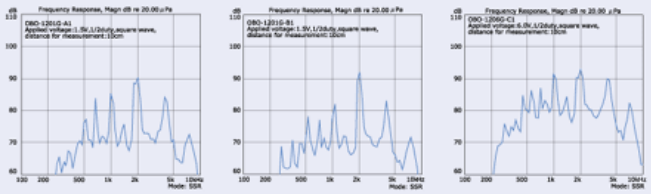
☆ Value applying rated voltage (2048Hz, 1/2 duty square wave)
★ Value applying (2048Hz, sine wave, measuring current 60μ A)

	OBO-1201G-A1	OBO-1201G-B1	OBO-1206G-C1
RATED VOLTAGE	1.5VDC	1.5VDC	6VDC
OPERATING VOLTAGE	1~2V	1~2V	4~8V
☆ RATED CURRENT	max. 55mA	max. 10mA	max. 30mA
☆ SOUND PRESSURE LEVEL AT 10 CM	min. 85dB	min. 85dB	min. 82dB
COIL RESISTANCE	16 ± 2Ω	42 ± 6Ω	73 ± 6Ω
★ COIL IMPEDANCE	abt 28Ω	abt 120Ω	abt 140Ω
RESONANT FREQUENCY	2,048 Hz ± 150 Hz	2,048 Hz ± 150 Hz	2,048 Hz ± 150 Hz
OPERATING TEMP. RANGE	-25°C to+70°C	-25°C to+70°C	-25°C to+70°C
STORAGE TEMP. RANGE	-30°C to+80°C	-30°C to+80°C	-30°C to+80°C
WEIGHT	2gms	2gms	2gms

Dimensions (Unit: mm ± 0.5)



Sound Pressure Level vs. Frequency
Applied Voltage : Rated Voltage, 1/2 duty Square wave
Distance for Measurement:10cm



9.2 Annex 2. PIC18F45K20

PIC18F23K20/24K20/25K20/26K20/ 43K20/44K20/45K20/46K20 Data Sheet

28/40/44-Pin Flash Microcontrollers with nanoWatt XLP Technology

 DS41303G

MICROCHIP

PIC18F2XK20/4XK20

28/40/44-Pin Flash Microcontrollers with nanoWatt XLP Technology

High-Performance RISC CPU:

- C Compiler Optimized Architecture:
  - Optional extended instruction set designed to optimize re-entrant code
- Up to 1024 bytes Data EEPROM
- Up to 64 Kbytes Linear Program Memory Addressing
- Up to 3936 bytes Linear Data Memory Addressing
- Up to 16 MIPS Operation
- 16-bit Wide Instructions, 8-bit Wide Data Path
- Priority Levels for Interrupts
- 31-Level, Software Accessible Hardware Stack
- 8 x 8 Single-Cycle Hardware Multiplier

Flexible Oscillator Structure:

- Precision 16 MHz Internal Oscillator Block:
  - Factory calibrated to ± 1%
  - Software selectable frequencies range of 31 kHz to 16 MHz
  - 64 MHz performance available using PLL – no external components required
- Four Crystal modes up to 64 MHz
- Two External Clock modes up to 64 MHz
- 4X Phase Lock Loop (PLL)
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops
  - Two-Speed Oscillator Start-up

Special Microcontroller Features:

- Operating Voltage Range: 1.8V to 3.6V
- Self-Programmable under Software Control
- Programmable 16-Level High/Low-Voltage Detection (HLVD) module:
  - Interrupt on High/Low-Voltage Detection
- Programmable Brown-out Reset (BOR):
  - With software enable option
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-Supply 3V In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) via Two Pins

Extreme Low-Power Management with nanoWatt XLP:

- Sleep mode: < 100 nA @ 1.8V
- Watchdog Timer: < 800 nA @ 1.8V
- Timer1 Oscillator: < 800 nA @ 32 kHz and 1.8V

Analog Features:

- Analog-to-Digital Converter (ADC) module:
  - 10-bit resolution, 13 External Channels
  - Auto-acquisition capability
  - Conversion available during Sleep
  - 1.2V Fixed Voltage Reference (FVR) channel
  - Independent input multiplexing
- Analog Comparator module:
  - Two rail-to-rail analog comparators
  - Independent input multiplexing
- Voltage Reference (CV_{REF}) module
  - Programmable (% V_{DD}), 16 steps
  - Two 16-level voltage ranges using V_{REF} pins

Peripheral Highlights:

- Up to 35 I/O Pins plus 1 Input-only Pin:
  - High-Current Sink/Source 25 mA/25 mA
  - Three programmable external interrupts
  - Four programmable interrupt-on-change
  - Eight programmable weak pull-ups
  - Programmable slew rate
- Capture/Compare/PWM (CCP) module
- Enhanced CCP (ECCP) module:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Master Synchronous Serial Port (MSSP) module
  - 3-wire SPI (supports all 4 modes)
  - I^2C™ Master and Slave modes with address mask
- Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module:
  - Supports RS-485, RS-232 and LIN
  - RS-232 operation using internal oscillator
  - Auto-Wake-up on Break
  - Auto-Baud Detect

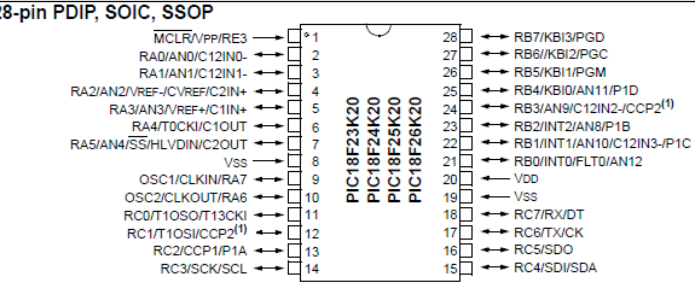
PIC18F2XK20/4XK20

Device	Program Memory		Data Memory		I/O[1]	10-bit A/D (ch)[2]	CCP/ ECCP (PWM)	MSSP		EUSART	Comp.	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I^2C™			
PIC18F23K20	8K	4096	512	256	25	11	1/1	Y	Y	1	2	1/3
PIC18F24K20	16K	8192	768	256	25	11	1/1	Y	Y	1	2	1/3
PIC18F25K20	32K	16384	1536	256	25	11	1/1	Y	Y	1	2	1/3
PIC18F26K20	64k	32768	3936	1024	25	11	1/1	Y	Y	1	2	1/3
PIC18F43K20	8K	4096	512	256	36	14	1/1	Y	Y	1	2	1/3
PIC18F44K20	16K	8192	768	256	36	14	1/1	Y	Y	1	2	1/3
PIC18F45K20	32K	16384	1536	256	36	14	1/1	Y	Y	1	2	1/3
PIC18F46K20	64k	32768	3936	1024	36	14	1/1	Y	Y	1	2	1/3

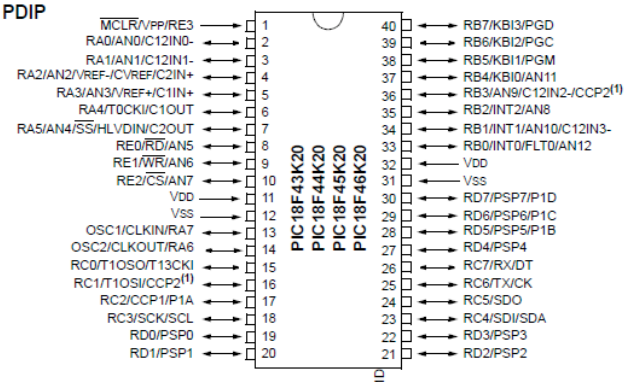
Note 1: One pin is input only.
2: Channel count includes internal fixed voltage reference channel.

Pin Diagrams

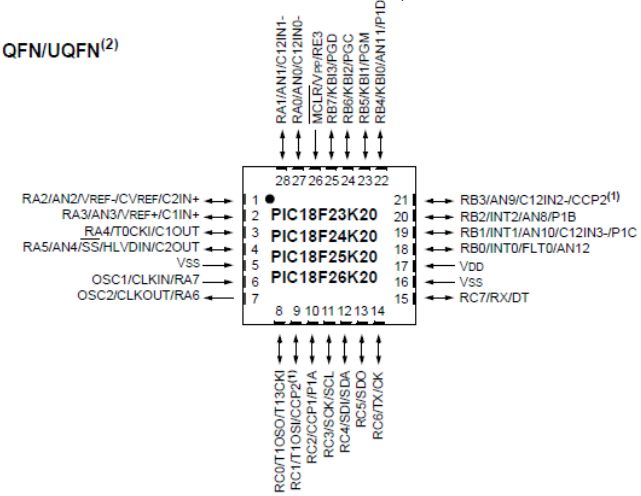
28-pin PDIP, SOIC, SSOP



40-pin PDIP



28-pin QFN/UQFN(2)

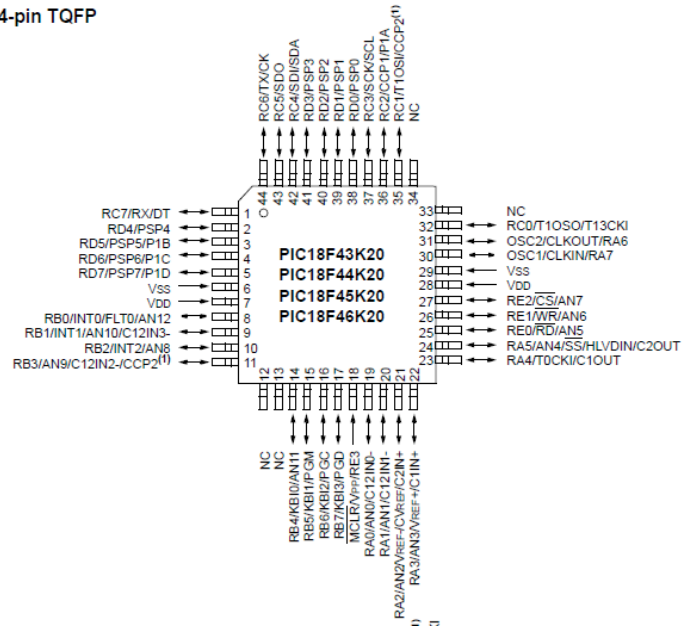


Note 1: RB3 is the alternate pin for CCP2 multiplexing.
2: UQFN package availability applies only to PIC18F23K20.

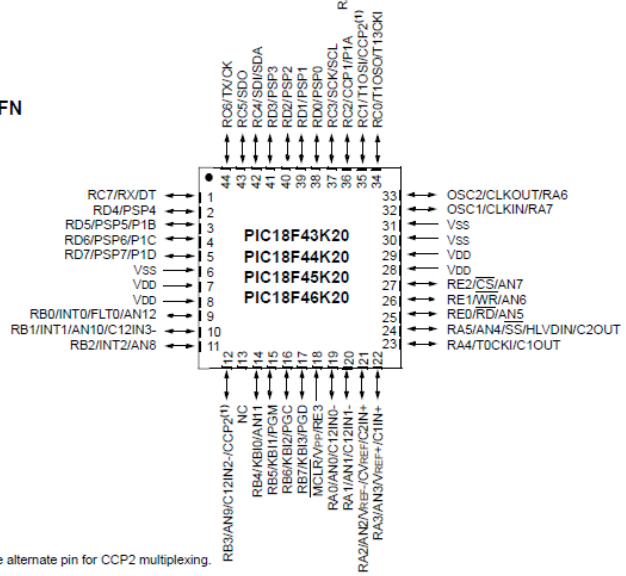
PIC18F2XK20/4XK20

Pin Diagrams (Cont.'d)

44-pin TQFP



44-pin QFN



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

TABLE 1: PIC18F4XK20 PIN SUMMARY

DIL Pin	TQFP Pin	QFN Pin	I/O	Analog	Comparator	Reference	ECCP	EUSART	MSSP	Timers	Slave	Interrupts	Pull-up	Basic
2	19	19	RA0	AN0	C12IN0-	—	—	—	—	—	—	—	—	—
3	20	20	RA1	AN1	C12IN1-	—	—	—	—	—	—	—	—	—
4	21	21	RA2	AN2	C2IN+	VREF-/ CVREF	—	—	—	—	—	—	—	—
5	22	22	RA3	AN3	C1IN+	VREF+	—	—	—	—	—	—	—	—
6	23	23	RA4		C1OUT	—	—	—	—	T0CKI	—	—	—	—
7	24	24	RA5	AN4	C2OUT	HLVDIN	—	—	$\overline{SS}$	—	—	—	—	—
14	31	33	RA6	—	—	—	—	—	—	—	—	—	—	OSC2/ CLKOUT
13	30	32	RA7	—	—	—	—	—	—	—	—	—	—	OSC1/CLKIN
33	8	9	RB0	AN12	—	—	FLT0	—	—	—	—	INT0	Yes	—
34	9	10	RB1	AN10	C12IN3-	—	—	—	—	—	—	INT1	Yes	—
35	10	11	RB2	AN8	—	—	—	—	—	—	—	INT2	Yes	—
36	11	12	RB3	AN9	C12IN2-	—	CCP2(1)	—	—	—	—	—	Yes	—
37	14	14	RB4	AN11	—	—	—	—	—	—	—	KBI0	Yes	—
38	15	15	RB5	—	—	—	—	—	—	—	—	KBI1	Yes	PGM
39	16	16	RB6	—	—	—	—	—	—	—	—	KBI2	Yes	PGC
40	17	17	RB7	—	—	—	—	—	—	—	—	KBI3	Yes	PGD
15	32	34	RC0	—	—	—	—	—	—	T1OSO/ T13CKI	—	—	—	—
16	35	35	RC1	—	—	—	CCP2(2)	—	—	T1OSI	—	—	—	—
17	36	36	RC2	—	—	—	CCP1/ P1A	—	—	—	—	—	—	—
18	37	37	RC3	—	—	—	—	—	SCK/ SCL	—	—	—	—	—
23	42	42	RC4	—	—	—	—	—	SDI/ SDA	—	—	—	—	—
24	43	43	RC5	—	—	—	—	—	SDO	—	—	—	—	—
25	44	44	RC6	—	—	—	—	TX/CK	—	—	—	—	—	—
26	1	1	RC7	—	—	—	—	RX/DT	—	—	—	—	—	—
19	38	38	RD0	—	—	—	—	—	—	—	PSP0	—	—	—
20	39	39	RD1	—	—	—	—	—	—	—	PSP1	—	—	—
21	40	40	RD2	—	—	—	—	—	—	—	PSP2	—	—	—
22	41	41	RD3	—	—	—	—	—	—	—	PSP3	—	—	—
27	2	2	RD4	—	—	—	—	—	—	—	PSP4	—	—	—
28	3	3	RD5	—	—	—	P1B	—	—	—	PSP5	—	—	—
29	4	4	RD6	—	—	—	P1C	—	—	—	PSP6	—	—	—
30	5	5	RD7	—	—	—	P1D	—	—	—	PSP7	—	—	—
8	25	25	RE0	AN5	—	—	—	—	—	—	$\overline{RD}$	—	—	—
9	26	26	RE1	AN6	—	—	—	—	—	—	$\overline{WR}$	—	—	—
10	27	27	RE2	AN7	—	—	—	—	—	—	$\overline{CS}$	—	—	—
1	18	18	RE3(3)	—	—	—	—	—	—	—	—	—	—	$\overline{MCLR}$/VPP
11	7	7	—	—	—	—	—	—	—	—	—	—	—	VDD
32	28	28	—	—	—	—	—	—	—	—	—	—	—	VDD
12	6	6	—	—	—	—	—	—	—	—	—	—	—	VSS
31	29	30	—	—	—	—	—	—	—	—	—	—	—	VSS
–	NC	8	—	—	—	—	—	—	—	—	—	—	—	VDD
–	NC	29	—	—	—	—	—	—	—	—	—	—	—	VDD
—	NC	31	—	—	—	—	—	—	—	—	—	—	—	VSS

Note 1: CCP2 multiplexed with RB3 when CONFIG3H<0> = 0
2: CCP2 multiplexed with RC1 when CONFIG3H<0> = 1
3: Input-only.

PIC18F2XK20/4XK20

TABLE 2: PIC18F2XK20 PIN SUMMARY

Pin DIL	Pin QUAD	I/O	Analog	Comparator	Reference	ECCP	EUSART	MSSP	Timers	Slave	Interrupts	Pull-up	Basic
2	27	RA0	AN0	C12IN0-									
3	28	RA1	AN1	C12IN1-									
4	1	RA2	AN2	C2IN+	VREF-/ CVREF								
5	2	RA3	AN3	C1IN+	VREF+								
6	3	RA4		C1OUT					T0CKI				
7	4	RA5	AN4	C2OUT	HLVDIN			$\overline{SS}$					
10	7	RA6											OSC2/ CLKOUT
9	6	RA7											OSC1/ CLKIN
21	18	RB0	AN12			FLT0					INT0	Yes	
22	19	RB1	AN10	C12IN3-		P1C					INT1	Yes	
23	20	RB2	AN8			P1B					INT2	Yes	
24	21	RB3	AN9	C12IN2-		CCP2(1)						Yes	
25	22	RB4	AN11			P1D					KBI0	Yes	
26	23	RB5									KBI1	Yes	PGM
27	24	RB6									KBI2	Yes	PGC
28	25	RB7									KBI3	Yes	PGD
11	8	RC0							T1OSO/ T13CKI				
12	9	RC1				CCP2(2)			T1OSI				
13	10	RC2				CCP1/ P1A							
14	11	RC3						SCK/ SCL					
15	12	RC4						SDI/ SDA					
16	13	RC5						SDO					
17	14	RC6					TX/CK						
18	15	RC7					RX/DT						
1	26	RE3(3)											$\overline{MCLR}$/ VPP
8	5												VSS
19	16												VSS
20	17												VDD

Note 1: CCP2 multiplexed with RB3 when CONFIG3H<0> = 0
2: CCP2 multiplexed with RC1 when CONFIG3H<0> = 1
3: Input-only

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F23K20
- PIC18F24K20
- PIC18F25K20
- PIC18F26K20
- PIC18F43K20
- PIC18F44K20
- PIC18F45K20
- PIC18F46K20

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Flash program memory. On top of these features, the PIC18F2XK20/4XK20 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F2XK20/4XK20 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes**: By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes**: The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching**: The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules**: The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 26.0 "Electrical Characteristics"** for values.

1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2XK20/4XK20 family offer ten different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes, using crystals or ceramic resonators
- Two External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O)
- Two External RC Oscillator modes with the same pin options as the External Clock modes
- An internal oscillator block which contains a 16 MHz HFINTOSC oscillator and a 31 kHz LFINTOSC oscillator which together provide 8 user selectable clock frequencies, from 31 kHz to 16 MHz. This option frees the two oscillator pins for use as additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the high-speed crystal and internal oscillator modes, which allows clock speeds of up to 64 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 64 MHz – all without using an external crystal or clock circuit.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor**: This option constantly monitors the main clock source against a reference signal provided by the LFINTOSC. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued operation or a safe application shutdown.
- **Two-Speed Start-up**: This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.2 Other Special Features

- **Memory Endurance:** The Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 10K for program memory and 100K for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F2XK20/4XK20 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Enhanced CCP module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include:
  - Auto-Shutdown, for disabling PWM outputs on interrupt or other select conditions
  - Auto-Restart, to reactivate outputs once the condition has cleared
  - Output steering to selectively enable one or more of 4 outputs to provide the PWM signal.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the USART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit postscaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 26.0 "Electrical Characteristics"** for time-out periods.

1.3 Details on Individual Family Members

Devices in the PIC18F2XK20/4XK20 family are available in 28-pin and 40/44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in five ways:

1. Flash program memory (8 Kbytes for PIC18F23K20/43K20 devices, 16 Kbytes for PIC18F24K20/44K20 devices, 32 Kbytes for PIC18F25K20/45K20 AND 64 Kbytes for PIC18F26K20/46K20).
2. A/D channels (11 for 28-pin devices, 14 for 40/44-pin devices).
3. I/O ports (3 bidirectional ports on 28-pin devices, 5 bidirectional ports on 40/44-pin devices).
4. Parallel Slave Port (present only on 40/44-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1.

The pinouts for all devices are listed in the pin summary tables: Table 1 and Table 2, and I/O description tables: Table 1-2 and Table 1-3.

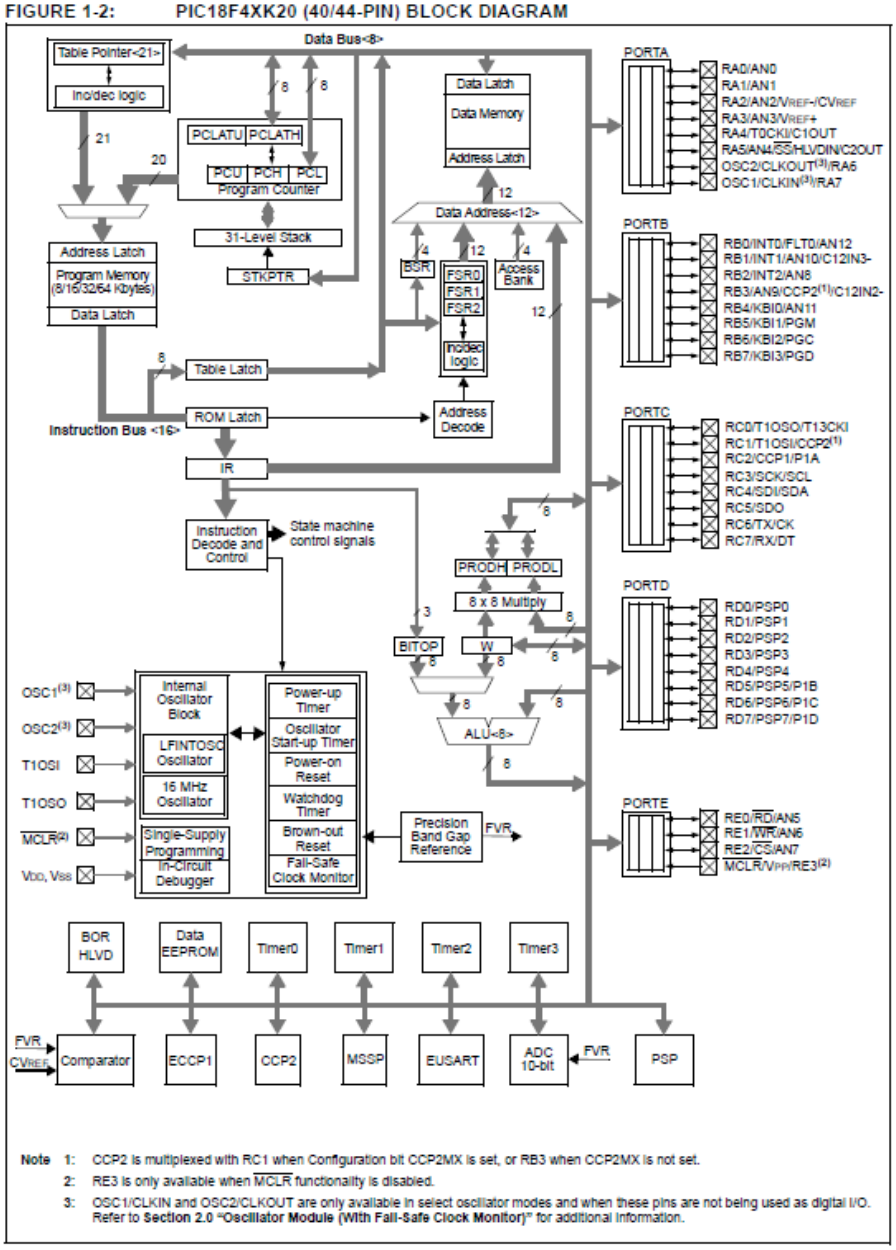
TABLE 1-1: DEVICE FEATURES

Features	PIC18F23K20	PIC18F24K20	PIC18F25K20	PIC18F26K20	PIC18F43K20	PIC18F44K20	PIC18F45K20	PIC18F46K20
Operating Frequency(2)	DC – 64 MHz	DC – 64 MHz	DC – 64 MHz	DC – 64 MHz	DC – 64 MHz	DC – 64 MHz	DC – 64 MHz	DC – 64 MHz
Program Memory (Bytes)	8192	16384	32768	65536	8192	16384	32768	65536
Program Memory (Instructions)	4096	8192	16384	32768	4096	8192	16384	32768
Data Memory (Bytes)	512	768	1536	3936	512	768	1536	3936
Data EEPROM Memory (Bytes)	256	256	256	1024	256	256	256	1024
Interrupt Sources	19	19	19	19	20	20	20	20
I/O Ports	A, B, C, (E)(1)	A, B, C, (E)(1)	A, B, C, (E)(1)	A, B, C, (E)(1)	A, B, C, D, E	A, B, C, D, E	A, B, C, D, E	A, B, C, D, E
Timers	4	4	44		44		44	
Capture/Compare/PWM Modules	1	1	1	1	1	1	1	1
Enhanced Capture/ Compare/PWM Modules	1	1	11		11		11	
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Parallel Communications (PSP)	No	No	No	No	Yes	Yes	Yes	Yes
10-bit Analog-to-Digital Module	1 internal plus 10 Input Channels	1 internal plus 10 Input Channels	1 internal plus 10 Input Channels	1 internal plus 10 Input Channels	1 internal plus 13 Input Channels	1 internal plus 13 Input Channels	1 internal plus 13 Input Channels	1 internal plus 13 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable High/ Low-Voltage Detect	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC 28-pin QFN 28-pin SSOP 28-pin UQFN	28-pin PDIP 28-pin SOIC 28-pin QFN 28-pin SSOP	28-pin PDIP 28-pin SOIC 28-pin QFN 28-pin SSOP	28-pin PDIP 28-pin SOIC 28-pin QFN 28-pin SSOP	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

Note 1: PORTE contains the single RE3 read-only bit. The LATE and TRISE registers are not implemented.

2: Frequency range shown applies to industrial range devices only. Maximum frequency for extended range devices is 48 MHz.

FIGURE 1-2: PIC18F4XK20 (40/44-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when Configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.

2: RE3 is only available when MCLR functionality is disabled.

3: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Module (With Fail-Safe Clock Monitor)"** for additional information.

PIC18F2XK20/4XK20

TABLE 1-2: PIC18F2XK20 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP, SOIC	QFN			
MCLR/VPP/RE3	1	26			Master Clear (input) or programming voltage (input)
MCLR			I	ST	Active-low Master Clear (device Reset) input
VPP			P		Programming voltage input
RE3			I	ST	Digital input
OSC1/CLKIN/RA7	9	6			Oscillator crystal or external clock input
OSC1			I	ST	Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise
CLKIN			I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKIN, OSC2/CLKOUT pins)
RA7			I/O	TTL	General purpose I/O pin
OSC2/CLKOUT/RA6	10	7			Oscillator crystal or clock output
OSC2			O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode
CLKOUT			O	—	In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate
RA6			I/O	TTL	General purpose I/O pin

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Default assignment for CCP2 when Configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when Configuration bit CCP2MX is cleared.

PIC18F2XK20/4XK20

TABLE 1-3: PIC18F4XK20 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
MCLR/VPP/RE3	1	18	18			Master Clear (input) or programming voltage (input)
MCLR				I	ST	Active-low Master Clear (device Reset) input
VPP				P		Programming voltage input
RE3				I	ST	Digital input
OSC1/CLKIN/RA7	13	32	30			Oscillator crystal or external clock input
OSC1				I	ST	Oscillator crystal input or external clock source input ST buffer when configured in RC mode; analog otherwise
CLKIN				I	CMOS	External clock source input. Always associated with pin function OSC1 (See related OSC1/CLKIN, OSC2/CLKOUT pins)
RA7				I/O	TTL	General purpose I/O pin
OSC2/CLKOUT/RA6	14	33	31			Oscillator crystal or clock output
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode
CLKOUT				O	—	In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate
RA6				I/O	TTL	General purpose I/O pin

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

Note 1: Default assignment for CCP2 when Configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when Configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4XK20 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTA is a bidirectional I/O port.
RA0/AN0/C12IN0-	2	19	19			
RA0				I/O	TTL	Digital I/O
AN0				I	Analog	Analog input 0, ADC channel 0
C12IN0-				I	Analog	Comparator C1 and C2 inverting input
RA1/AN1/C12IN0-	3	20	20			
RA1				I/O	TTL	Digital I/O
AN1				I	Analog	Analog input 1, ADC channel 1
C12IN0-				I	Analog	Comparator C1 and C2 inverting input
RA2/AN2/VREF-/CVREF/ C2IN+	4	21	21			
RA2				I/O	TTL	Digital I/O
AN2				I	Analog	Analog input 2, ADC channel 2
VREF-				I	Analog	A/D reference voltage (low) input
CVREF				O	Analog	Comparator reference voltage output
C2IN+				I	Analog	Comparator C2 non-inverting input
RA3/AN3/VREF+/ C1IN+	5	22	22			
RA3				I/O	TTL	Digital I/O
AN3				I	Analog	Analog input 3, ADC channel 3
VREF+				I	Analog	A/D reference voltage (high) input
C1IN+				I	Analog	Comparator C1 non-inverting input
RA4/T0CKI/C1OUT	6	23	23			
RA4				I/O	ST	Digital I/O
T0CKI				I	ST	Timer0 external clock input
C1OUT				O	CMOS	Comparator C1 output
RA5/AN4/$\overline{SS}$/HLVDIN/ C2OUT	7	24	24			
RA5				I/O	TTL	Digital I/O
AN4				I	Analog	Analog input 4, ADC channel 4
$\overline{SS}$				I	TTL	SPI slave select input
HLVDIN				I	Analog	High/Low-Voltage Detect input
C2OUT				O	CMOS	Comparator C2 output
RA6						See the OSC2/CLKOUT/RA6 pin
RA7						See the OSC1/CLKIN/RA7 pin

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Default assignment for CCP2 when Configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when Configuration bit CCP2MX is cleared.

PIC18F2XK20/4XK20

TABLE 1-3: PIC18F4XK20 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-up on each input.
RB0/INT0/FLT0/AN12	33	9	8			
RB0				I/O	TTL	Digital I/O
INT0				I	ST	External interrupt 0
FLT0				I	ST	PWM Fault input for Enhanced CCP1
AN12				I	Analog	Analog input 12, ADC channel 12
RB1/INT1/AN10/ C12IN3-	34	10	9			
RB1				I/O	TTL	Digital I/O
INT1				I	ST	External interrupt 1
AN10				I	Analog	Analog input 10, ADC channel 10
C12IN3-				I	Analog	Comparator C1 and C2 inverting input
RB2/INT2/AN8	35	11	10			
RB2				I/O	TTL	Digital I/O
INT2				I	ST	External interrupt 2
AN8				I	Analog	Analog input 8, ADC channel 8
RB3/AN9/C12IN2-/ CCP2	36	12	11			
RB3				I/O	TTL	Digital I/O
AN9				I	Analog	Analog input 9, ADC channel 9
C12IN23-				I	Analog	Comparator C1 and C2 inverting input
CCP2(2)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output
RB4/KBI0/AN11	37	14	14			
RB4				I/O	TTL	Digital I/O
KBI0				I	TTL	Interrupt-on-change pin
AN11				I	Analog	Analog input 11, ADC channel 11
RB5/KBI1/PGM	38	15	15			
RB5				I/O	TTL	Digital I/O
KBI1				I	TTL	Interrupt-on-change pin
PGM				I/O	ST	Low-Voltage ICSP™ Programming enable pin
RB6/KBI2/PGC	39	16	16			
RB6				I/O	TTL	Digital I/O
KBI2				I	TTL	Interrupt-on-change pin
PGC				I/O	ST	In-Circuit Debugger and ICSP™ programming clock pin
RB7/KBI3/PGD	40	17	17			
RB7				I/O	TTL	Digital I/O
KBI3				I	TTL	Interrupt-on-change pin
PGD				I/O	ST	In-Circuit Debugger and ICSP™ programming data pin

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

Note 1: Default assignment for CCP2 when Configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when Configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4XK20 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	15	34	32			
RC0				I/O	ST	Digital I/O
T1OSO				O	—	Timer1 oscillator output
T13CKI				I	ST	Timer1/Timer3 external clock input
RC1/T1OSI/CCP2	16	35	35			
RC1				I/O	ST	Digital I/O
T1OSI				I	CMOS	Timer1 oscillator input
CCP2(1)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output
RC2/CCP1/P1A	17	36	36			
RC2				I/O	ST	Digital I/O
CCP1				I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output
P1A				O	—	Enhanced CCP1 output
RC3/SCK/SCL	18	37	37			
RC3				I/O	ST	Digital I/O
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode
SCL				I/O	ST	Synchronous serial clock input/output for I^2C™ mode
RC4/SDI/SDA	23	42	42			
RC4				I/O	ST	Digital I/O
SDI				I	ST	SPI data in
SDA				I/O	ST	I^2C™ data I/O
RC5/SDO	24	43	43			
RC5				I/O	ST	Digital I/O
SDO				O	—	SPI data out
RC6/TX/CK	25	44	44			
RC6				I/O	ST	Digital I/O
TX				O	—	EUSART asynchronous transmit
CK				I/O	ST	EUSART synchronous clock (see related RX/DT)
RC7/RX/DT	26	1	1			
RC7				I/O	ST	Digital I/O
RX				I	ST	EUSART asynchronous receive
DT				I/O	ST	EUSART synchronous data (see related TX/CK)

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

Note 1: Default assignment for CCP2 when Configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when Configuration bit CCP2MX is cleared.

PIC18F2XK20/4XK20

TABLE 1-3: PIC18F4XK20 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTD is a bidirectional I/O port or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
RD0/PSP0	19	38	38			
RD0				I/O	ST	Digital I/O
PSP0				I/O	TTL	Parallel Slave Port data
RD1/PSP1	20	39	39			
RD1				I/O	ST	Digital I/O
PSP1				I/O	TTL	Parallel Slave Port data
RD2/PSP2	21	40	40			
RD2				I/O	ST	Digital I/O
PSP2				I/O	TTL	Parallel Slave Port data
RD3/PSP3	22	41	41			
RD3				I/O	ST	Digital I/O
PSP3				I/O	TTL	Parallel Slave Port data
RD4/PSP4	27	2	2			
RD4				I/O	ST	Digital I/O
PSP4				I/O	TTL	Parallel Slave Port data
RD5/PSP5/P1B	28	3	3			
RD5				I/O	ST	Digital I/O
PSP5				I/O	TTL	Parallel Slave Port data
P1B				O	—	Enhanced CCP1 output
RD6/PSP6/P1C	29	4	4			
RD6				I/O	ST	Digital I/O
PSP6				I/O	TTL	Parallel Slave Port data
P1C				O	—	Enhanced CCP1 output
RD7/PSP7/P1D	30	5	5			
RD7				I/O	ST	Digital I/O
PSP7				I/O	TTL	Parallel Slave Port data
P1D				O	—	Enhanced CCP1 output

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels I = Input
O = Output P = Power

Note 1: Default assignment for CCP2 when Configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when Configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4XK20 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTE is a bidirectional I/O port
RE0/$\overline{RD}$/AN5	8	25	25			
RE0				I/O	ST	Digital I/O
$\overline{RD}$				I	TTL	Read control for Parallel Slave Port (see related $\overline{WR}$ and $\overline{CS}$ pins)
AN5				I	Analog	Analog input 5, ADC channel 5
RE1/$\overline{WR}$/AN6	9	26	26			
RE1				I/O	ST	Digital I/O
$\overline{WR}$				I	TTL	Write control for Parallel Slave Port (see related $\overline{CS}$ and $\overline{RD}$ pins)
AN6				I	Analog	Analog input 6, ADC channel 6
RE2/$\overline{CS}$/AN7	10	27	27			
RE2				I/O	ST	Digital I/O
$\overline{CS}$				I	TTL	Chip Select control for Parallel Slave Port (see related $\overline{RD}$ and $\overline{WR}$)
AN7				I	Analog	Analog input 7, ADC channel 7
RE3	—	—	—	—	—	See $\overline{MCLR}$/VPP/RE3 pin
VSS	12, 31	6, 30, 31	6, 29	P	—	Ground reference for logic and I/O pins
VDD	11, 32	7, 8, 28, 29	7, 28	P	—	Positive supply for logic and I/O pins
NC	—	13	12, 13, 33, 34	—	—	No connect

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Default assignment for CCP2 when Configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when Configuration bit CCP2MX is cleared.

9.3 Annex. 44-Pin demo board

44-PIN DEMO BOARD USER'S GUIDE

 DS41296A

MICROCHIP

44-PIN DEMO BOARD USER'S GUIDE

Chapter 1. 44-Pin Demo Board Overview

1.1 INTRODUCTION

The 44-Pin Demo Board is a small and simple demonstration PCB for Microchip's 44-pin Thin Quad Flatpack (TQFP) PICmicro® Microcontroller Units (MCU). It is populated with a PIC16F917 MCU, eight LEDs, push button and potentiometer. The demo board has several test points to access the I/O pins of the MCU and a surface mount prototyping area. The MCU can be programmed with the PICkit™ 2 Microcontroller Programmer or the MPLAB® ICD 2 using the RJ-11 to 6-pin inline adapter (AC164110).

1.2 HIGHLIGHTS

This chapter discusses:

- Devices supported by the 44-Pin Demo Board
- The 44-Pin Demo Board Overview
- Running the Default Demonstration

1.3 DEVICES SUPPORTED BY THE 44-PIN DEMO BOARD

The 44-Pin Demo Board can be used with virtually any 44-pin Thin Quad Flatpack (TQFP) PICmicro MCU. The assembled 44-Pin Demo Board is populated with a PIC16F917-I/PT microcontroller.

Additional 44-Pin Demo Boards can be ordered from Microchip Technology and distributors. Part number, DM164120-2, comes with one assembled and two blank 44-Pin Demo Boards. The blank demo board can be used for evaluating or prototyping circuits using any of the 44-pin devices listed below.

44-pin TQFP Flash Devices:

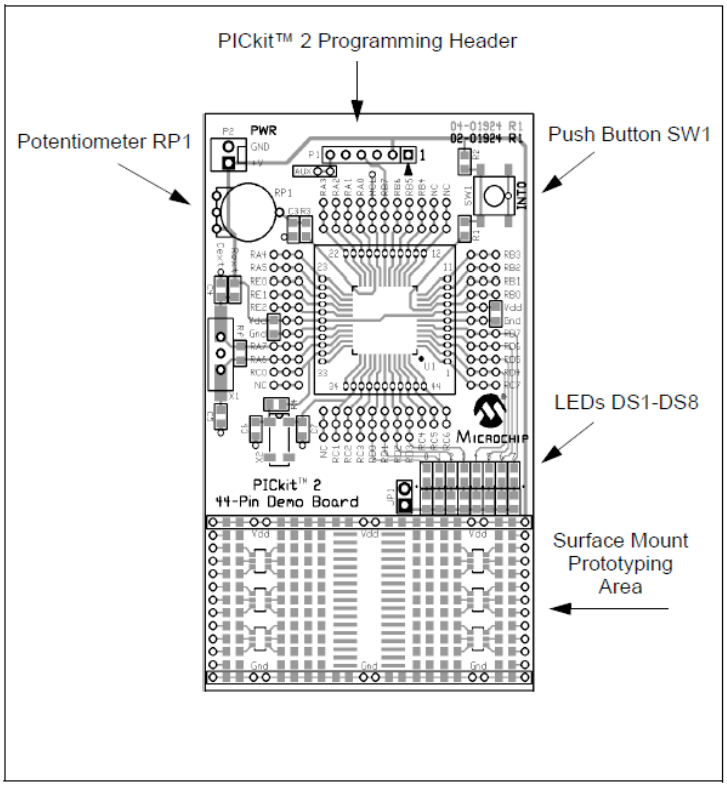
- PIC16F74
- PIC16F747
- PIC16F77
- PIC16F777
- PIC16F871
- PIC16F874A
- PIC16F877A
- PIC16F917
- PIC18F4220
- PIC18F4320
- PIC18F4331
- PIC18F4410
- PIC18F4420
- PIC18F4431
- PIC18F4450
- PIC18F4455
- PIC18F4480
- PIC18F44J10
- PIC18F4510
- PIC18F4515
- PIC18F4520
- PIC18F4525
- PIC18F4550
- PIC18F4580
- PIC18F4585
- PIC18F45J10
- PIC18F4610
- PIC18F4620
- PIC18F4680

44-Pin Demo Board User's Guide

1.4 44-PIN DEMO BOARD OVERVIEW

The 44-Pin Demo Board is populated with a PIC16F917 MCU (U1), eight LEDs (DS1-DS8), push button (SW1) and potentiometer (RP1). The board layout is shown in Figure 1-1. The demo board has several test points to access the I/O pins of the MCU and a surface mount prototyping area. The MCU can be programmed with the PICkit™ 2 Microcontroller Programmer from header P1.

FIGURE 1-1: 44-PIN DEMO BOARD



1.5 RUNNING THE DEFAULT DEMONSTRATION

The assembled 44-Pin Demo Board comes preprogrammed with a demonstration program. To use this program, power the demo board (3.0 - 5.5VDC) using a PICkit™ 2 Microcontroller Programmer, or a bench power supply connected to header P2. To use the PICkit™ 2 Microcontroller Programmer, connect it to a PC USB port using the USB cable. Start the PICkit™ 2 Microcontroller Programmer PC application and click on the target power box to apply power to the demo board. The demo program will blink the eight red lights in succession. Press the push button switch, labeled SW1, and the sequence of the lights will reverse. Rotate the potentiometer, RP1, and the light sequence will blink at a different rate.

MICROCHIP

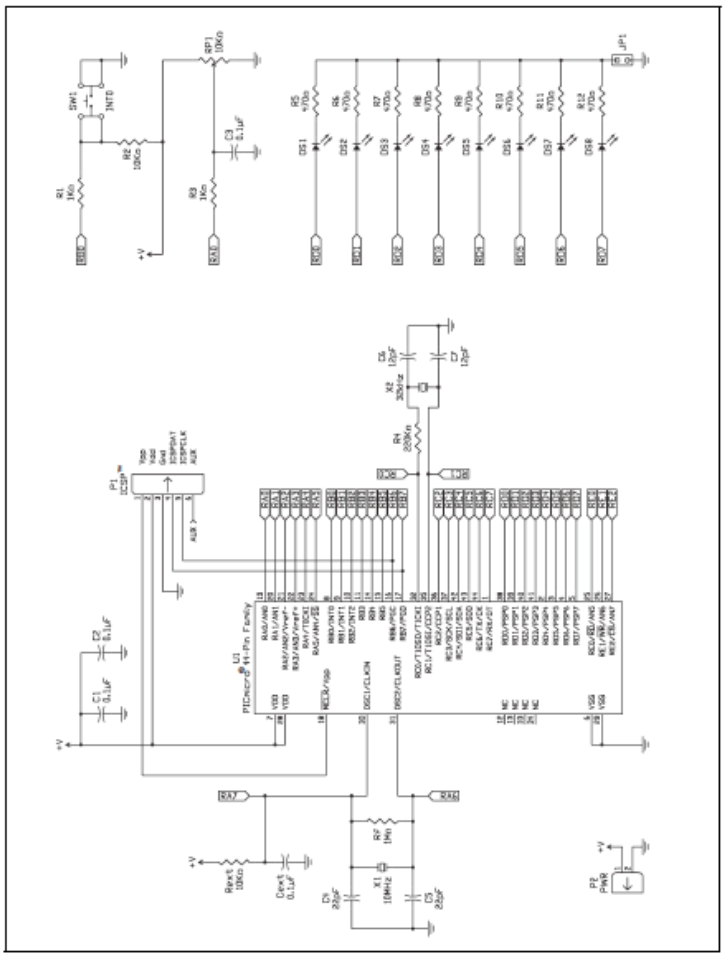
44-PIN DEMO BOARD USER'S GUIDE

Appendix A. Hardware Schematics

A.1 INTRODUCTION

This appendix contains the 44-Pin Demo Board schematic, PCB layout and Bill of Materials.

FIGURE A-1: 44-PIN DEMO BOARD SCHEMATIC DIAGRAM



44-Pin Demo Board User's Guide

FIGURE A-2: 44-PIN DEMO BOARD SILKSCREEN

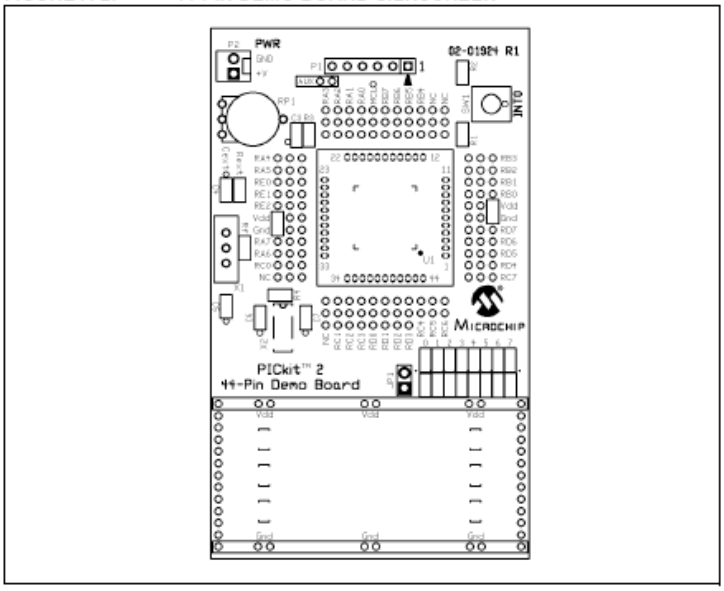


FIGURE A-3: 44-PIN DEMO BOARD TOP COPPER

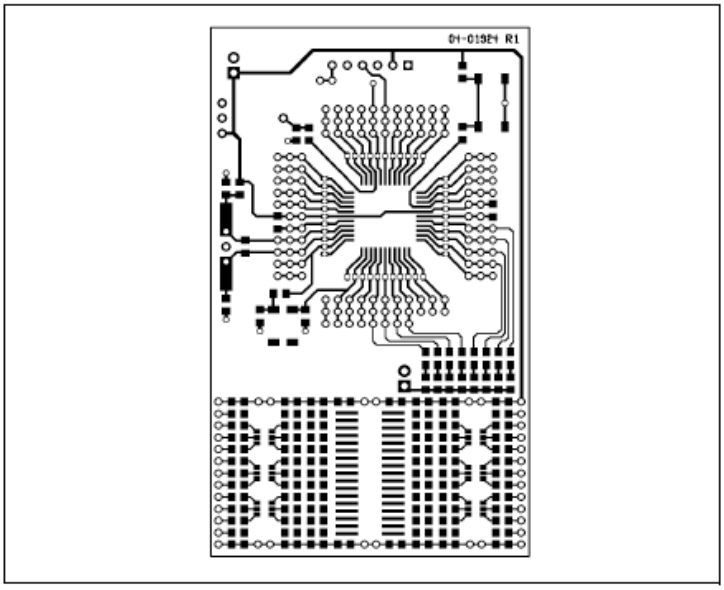


FIGURE A-4: 44-PIN DEMO BOARD BOTTOM COPPER

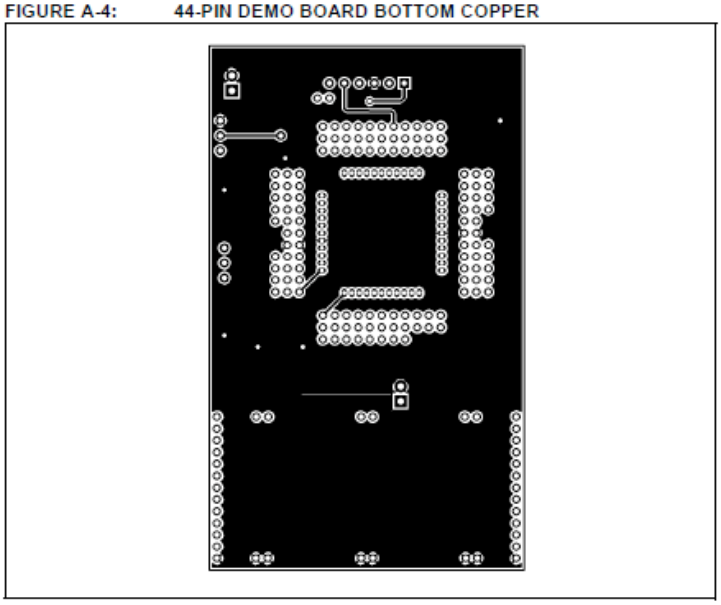


TABLE A-1: 44-PIN DEMO BOARD BILL OF MATERIALS

Bill of Materials		
Designation	**Qty**	**Description**
C1, C2, C3	3	Capacitor, Ceramic, 0805 SMT, 0.1 µF, 16V, 5%, X7R
R5-R12	8	Resistor, 0805 SMT, 470Ω, 5%, 1/8W
R1, R3	2	Resistor, 0805 SMT, 1 kΩ, 5%, 1/8W
R2	1	Resistor, 0805 SMT, 10 kΩ, 5%, 1/8W
RP1	1	Potentiometer 10 kΩ, thumbwheel
DS1-DS8	8	LED, 0805 SMT, Red Clear
SW1	1	Switch, push button, momentary
U1 – Microcontroller	1	44-pin PICmicro® MCU
P1	1	Connector, header, right-angle, 6-pin, 0.100" spacing, 0.025" square
JP1	1	Connector, header, 2-pin, 0.100" spacing, 0.025" square
Rubber feet	4	Bumpon square, 0.40 x 0.10, black

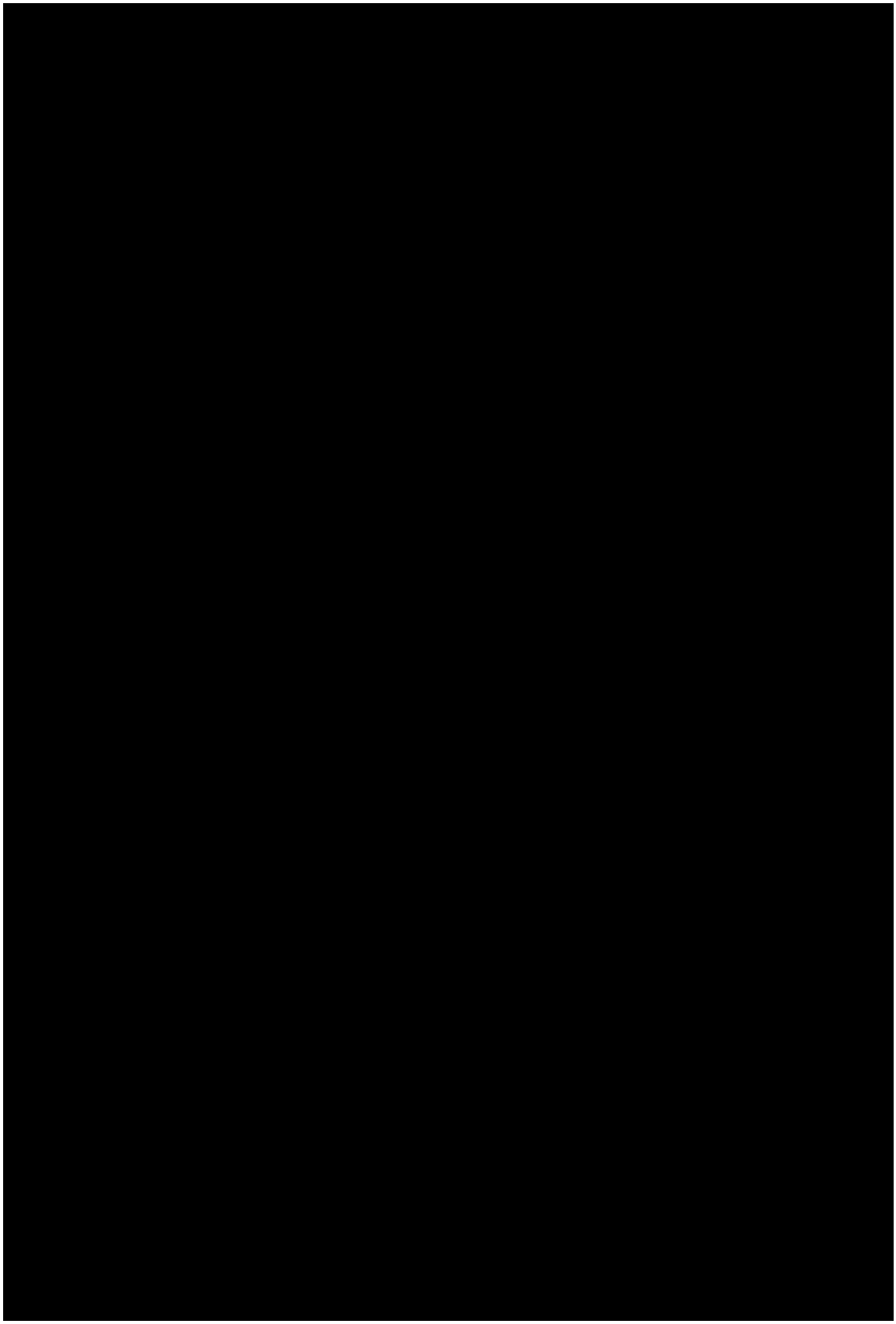
9.4 Annex 4. Taula ASCII

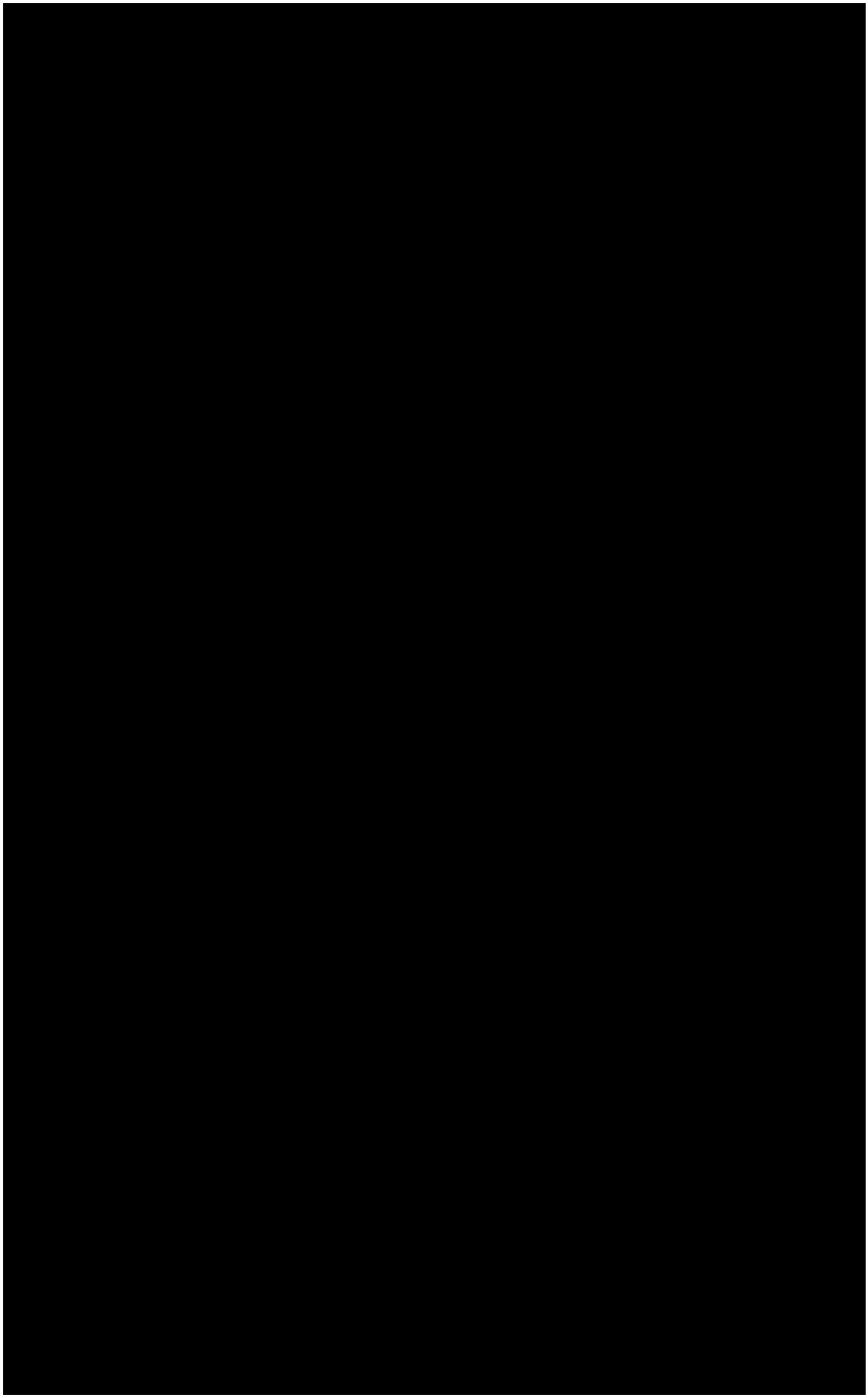
Binario	Dec	Hex	Representación
0010 0000	32	20	espacio ( )
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Binario	Dec	Hex	Representación
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

Binario	Dec	Hex	Representación
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

9.5 Annex 5. Exemple configuració equip





9.6 Annex 6. Amplificadors operacionls

MICROCHIP

MCP6241/1R/1U/2/4

50 µA, 550 kHz Rail-to-Rail Op Amp

Features

- Gain Bandwidth Product: 550 kHz (typical)
- Supply Current: I_Q = 50 µA (typical)
- Supply Voltage: 1.8V to 5.5V
- Rail-to-Rail Input/Output
- Extended Temperature Range: -40°C to +125°C
- Available in 5-pin SC-70 and SOT-23 packages

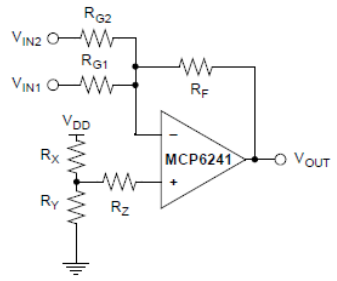
Applications

- Automotive
- Portable Equipment
- Photodiode (Transimpedance) Amplifier
- Analog Filters
- Notebooks and PDAs
- Battery-Powered Systems

Design Aids

- SPICE Macro Models
- Mindi™ Circuit Designer & Simulator
- Microchip Advanced Part Selector (MAPS)
- Analog Demonstration and Evaluation Boards
- Application Notes

Typical Application

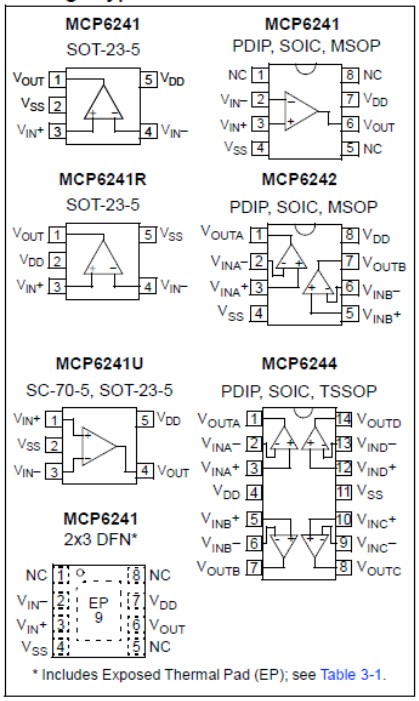


Summing Amplifier Circuit

Description

The Microchip Technology Inc. MCP6241/1R/1U/2/4 operational amplifiers (op amps) provide wide bandwidth for the quiescent current. The MCP6241/1R/1U/2/4 has a 550 kHz gain bandwidth product and 68° (typical) phase margin. This family operates from a single supply voltage as low as 1.8V, while drawing 50 µA (typical) quiescent current. In addition, the MCP6241/1R/1U/2/4 family supports rail-to-rail input and output swing, with a common mode input voltage range of V_{DD} + 300 mV to V_{SS} – 300 mV. These op amps are designed in one of Microchip's advanced CMOS processes.

Package Types



* Includes Exposed Thermal Pad (EP); see Table 3-1.

1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

$V_{DD} - V_{SS}$ 7.0V
Current at Analog Input Pins ($V_{IN}+$, $V_{IN}-$).....±2 mA
Analog Inputs ($V_{IN}+$, $V_{IN}-$) †† ..... V_{SS} – 1.0V to V_{DD} + 1.0V
All Other Inputs and Outputs ..... V_{SS} – 0.3V to V_{DD} + 0.3V
Difference Input Voltage ..... $|V_{DD} - V_{SS}|$
Output Short Circuit Current .....Continuous
Current at Output and Supply Pins .....±30 mA
Storage Temperature .....–65° C to +150°C
Maximum Junction Temperature (T_J).....+150°C
ESD Protection On All Pins (HBM; MM) ..... ≥ 4 kV; 300V

† **Notice:** Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

†† See **Section 4.1.2 "Input Voltage and Current Limits"**.

DC ELECTRICAL CHARACTERISTICS

Electrical Characteristics: Unless otherwise indicated, T_A = +25°C, V_{DD} = +1.8V to +5.5V, V_{SS} = GND, $V_{CM} = V_{DD}/2$, R_L = 100 kΩ to $V_{DD}/2$ and $V_{OUT} \approx V_{DD}/2$.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Input Offset						
Input Offset Voltage	V_{OS}	-5.0	—	+5.0	mV	$V_{CM} = V_{SS}$
Extended Temperature	V_{OS}	-7.0	—	+7.0	mV	T_A= -40°C to +125°C, $V_{CM} = V_{SS}$ **(Note 1)**
Input Offset Drift with Temperature	$\Delta V_{OS}/\Delta T_A$	—	±3.0	—	µV/°C	T_A= -40°C to +125°C, $V_{CM} = V_{SS}$
Power Supply Rejection	PSRR	—	83	—	dB	$V_{CM} = V_{SS}$
Input Bias Current and Impedance						
Input Bias Current:	I_B	—	±1.0	—	pA	
At Temperature	I_B	—	20	—	pA	T_A = +85°C
At Temperature	I_B	—	1100	—	pA	T_A = +125°C
Input Offset Current	I_{OS}	—	±1.0	—	pA	
Common Mode Input Impedance	Z_{CM}	—	10^{13}||6	—	Ω||pF	
Differential Input Impedance	Z_{DIFF}	—	10^{13}||3	—	Ω||pF	
Common Mode						
Common Mode Input Range	V_{CMR}	V_{SS} – 0.3	—	V_{DD} + 0.3	V	
Common Mode Rejection Ratio	CMRR	60	75	—	dB	V_{CM} = -0.3V to 5.3V, V_{DD} = 5V
Open-Loop Gain						
DC Open-Loop Gain (large signal)	A_{OL}	90	110	—	dB	V_{OUT} = 0.3V to V_{DD} – 0.3V, $V_{CM} = V_{SS}$
Output						
Maximum Output Voltage Swing	V_{OL}, V_{OH}	V_{SS} + 35	—	V_{DD} – 35	mV	R_L = 10 kΩ, 0.5V Input Overdrive
Output Short-Circuit Current	I_{SC}	—	±6	—	mA	V_{DD} = 1.8V
	I_{SC}	—	±23	—	mA	V_{DD} = 5.5V
Power Supply						
Supply Voltage	V_{DD}	1.8	—	5.5	V	
Quiescent Current per Amplifier	I_Q	30	50	70	µA	I_O = 0, $V_{CM} = V_{DD}$ – 0.5V

Note 1: The SC-70 package is only tested at +25°C.

MCP6241/1R/1U/2/4

AC ELECTRICAL CHARACTERISTICS

Electrical Characteristics: Unless otherwise indicated, T_A = +25°C, V_{DD} = +1.8 to 5.5V, V_{SS} = GND, V_{CM} = V_{DD}/2, V_{OUT} ≈ V_{DD}/2, R_L = 10 kΩ to V_{DD}/2 and C_L = 60 pF.

Parameters	Sym	Min	Typ	Max	Units	Conditions
AC Response						
Gain Bandwidth Product	GBWP	—	550	—	kHz	
Phase Margin	PM	—	68	—	°	G = +1 V/V
Slew Rate	SR	—	0.30	—	V/µs	
Noise						
Input Noise Voltage	E_{ni}	—	10	—	μV_{P-P}	f = 0.1 Hz to 10 Hz
Input Noise Voltage Density	e_{ni}	—	45	—	nV/√Hz	f = 1 kHz
Input Noise Current Density	i_{ni}	—	0.6	—	fA/√Hz	f = 1 kHz

TEMPERATURE CHARACTERISTICS

Electrical Characteristics: Unless otherwise indicated, V_{DD} = +1.8V to +5.5V and V_{SS} = GND.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Temperature Ranges						
Extended Temperature Range	T_A	-40	—	+125	°C	
Operating Temperature Range	T_A	-40	—	+125	°C	**(Note)**
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, 5L-SC70	θ_{JA}	—	331	—	°C/W	
Thermal Resistance, 5L-SOT-23	θ_{JA}	—	256	—	°C/W	
Thermal Resistance, 8L-DFN (2x3)	θ_{JA}	—	84.5	—	°C/W	
Thermal Resistance, 8L-MSOP	θ_{JA}	—	206	—	°C/W	
Thermal Resistance, 8L-PDIP	θ_{JA}	—	85	—	°C/W	
Thermal Resistance, 8L-SOIC	θ_{JA}	—	163	—	°C/W	
Thermal Resistance, 14L-PDIP	θ_{JA}	—	70	—	°C/W	
Thermal Resistance, 14L-SOIC	θ_{JA}	—	120	—	°C/W	
Thermal Resistance, 14L-TSSOP	θ_{JA}	—	100	—	°C/W	

Note: The internal Junction Temperature (T_J) must not exceed the Absolute Maximum specification of +150°C.

1.1 Test Circuits

The test circuits used for the DC and AC tests are shown in Figure 1-1 and Figure 1-2. The bypass capacitors are laid out according to the rules discussed in **Section 4.6 "PCB Surface Leakage"**.

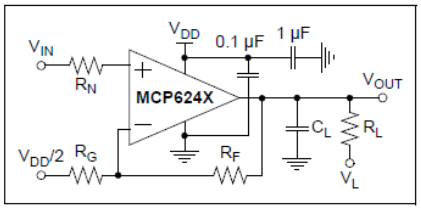


FIGURE 1-1: *AC and DC Test Circuit for Most Non-Inverting Gain Conditions.*

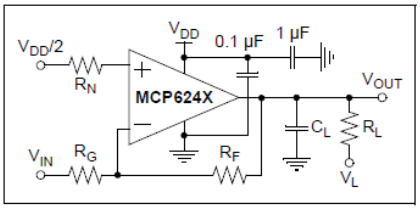


FIGURE 1-2: *AC and DC Test Circuit for Most Inverting Gain Conditions.*

9.7 Annex 7. Esquemàtic hardware de recepció

9.8 Annex 8. Esquema de muntatge hardware de recepció

9.9 Annex 9. PIC18F46J50

PIC18F46J50
Data Sheet

28/44-Pin, Low-Power,
High-Performance USB Microcontrollers
with nanoWatt XLP Technology

 DS39931D

MICROCHIP

PIC18F46J50 FAMILY

28/44-Pin, Low-Power, High-Performance USB Microcontrollers

Power Management Features with nanoWatt XLP™ for Extreme Low-Power:

- Deep Sleep mode: CPU off, Peripherals off, Currents Down to 13 nA and 850 nA with RTCC:
  - Able to wake-up on external triggers, programmable WDT or RTCC alarm
  - Ultra Low-Power Wake-up (ULPWU)
- Sleep mode: CPU off, Peripherals off, SRAM on, Fast Wake-up, Currents Down to 105 nA, Typical
- Idle: CPU off, Peripherals on, Currents Down to 2.3 µA, Typical
- Run: CPU on, Peripherals on, Currents Down to 6.2 µA, Typical
- Timer1 Oscillator w/RTCC: 1 µA, 32 kHz, Typical
- Watchdog Timer: 0.8 µA, 2V, Typical

Special Microcontroller Features:

- Low-Power, High-Speed CMOS Flash Technology
- C Compiler Optimized Architecture for Re-Entrant Code
- Priority Levels for Interrupts
- Self-Programmable under Software Control
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-Supply In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) w/Three Breakpoints via 2 Pins
- Operating Voltage Range of 2.0V to 3.6V
- On-Chip 2.5V Regulator
- Flash Program Memory of 10,000 Erase/Write Cycles Minimum and 20-Year Data Retention

Universal Serial Bus (USB) Features

- USB V2.0 Compliant
- Full Speed (12 Mbps) and Low Speed (1.5 Mbps)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- USB module can use any RAM Location on the Device as USB Endpoint Buffers
- On-Chip USB Transceiver with Crystal-less operation

Flexible Oscillator Structure:

- High-Precision Internal Oscillator (±0.15% typ.) for USB
- Two External Clock modes, up to 48 MHz (12 MIPS)
- Low-Power, 31 kHz Internal RC Oscillator
- Tunable Internal Oscillator (31 kHz to 8 MHz, or up to 48 MHz with PLL)
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops
- Two-Speed Oscillator Start-up
- Programmable Reference Clock Output Generator

Peripheral Highlights:

- Peripheral Pin Select:
  - Allows independent I/O mapping of many peripherals
  - Continuous hardware integrity checking and safety interlocks prevent unintentional configuration changes
- Hardware Real-Time Clock and Calendar (RTCC):
  - Provides clock, calendar and alarm functions
- High-Current Sink/Source 25 mA/25 mA (PORTB and PORTC)
- 5.5V Tolerant Inputs (digital only pins)
- Four Programmable External Interrupts
- Four Input Change Interrupts
- Two Enhanced Capture/Compare/PWM (ECCP) modules:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
  - Pulse steering control
- Two Master Synchronous Serial Port (MSSP) modules Supporting Three-Wire SPI (all four modes) and I^2C™ Master and Slave modes
- Full-Duplex Master/Slave SPI DMA Engine
- 8-Bit Parallel Master Port/Enhanced Parallel Slave Port
- Two-Rail – Rail Analog Comparators with Input Multiplexing
- 10-Bit, up to 13-Channel Analog-to-Digital (A/D) Converter module:
  - Auto-acquisition capability
  - Conversion available during Sleep
  - Self-calibration
- High/Low-Voltage Detect module
- Charge Time Measurement Unit (CTMU):
  - Supports capacitive touch sensing for touch screens and capacitive switches
  - Provides a precise resolution time measurement for both flow measurement and simple temperature sensing
- Two Enhanced USART modules:
  - Supports RS-485, RS-232 and LIN/J2602
  - Auto-Wake-up on Start bit
- Auto-Baud Detect

PIC18F46J50 FAMILY

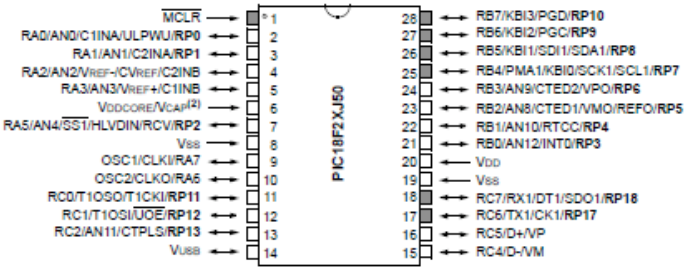
PIC18F/LF[1] Device	Pins	Program Memory (bytes)	SRAM (bytes)	Remappable Pins	Timers 8/16-Bit	ECCP/(PWM)	EUSART	MSSP		10-Bit A/D (ch)	Comparators	Deep Sleep	PMP/PSP	CTMU	RTCC	USB
									SPI w/DMA	I^2C™							
PIC18F24J50	28	16K	3776	16	2/3	2	2	2	Y	Y	10	2	Y	N	Y	Y	Y
PIC18F25J50	28	32K	3776	16	2/3	2	2	2	Y	Y	10	2	Y	N	Y	Y	Y
PIC18F26J50	28	64K	3776	16	2/3	2	2	2	Y	Y	10	2	Y	N	Y	Y	Y
PIC18F44J50	44	16K	3776	22	2/3	2	2	2	Y	Y	13	2	Y	Y	Y	Y	Y
PIC18F45J50	44	32K	3776	22	2/3	2	2	2	Y	Y	13	2	Y	Y	Y	Y	Y
PIC18F46J50	44	64K	3776	22	2/3	2	2	2	Y	Y	13	2	Y	Y	Y	Y	Y
PIC18LF24J50	28	16K	3776	16	2/3	2	2	2	Y	Y	10	2	N	N	Y	Y	Y
PIC18LF25J50	28	32K	3776	16	2/3	2	2	2	Y	Y	10	2	N	N	Y	Y	Y
PIC18LF26J50	28	64K	3776	16	2/3	2	2	2	Y	Y	10	2	N	N	Y	Y	Y
PIC18LF44J50	44	16K	3776	22	2/3	2	2	2	Y	Y	13	2	N	Y	Y	Y	Y
PIC18LF45J50	44	32K	3776	22	2/3	2	2	2	Y	Y	13	2	N	Y	Y	Y	Y
PIC18LF46J50	44	64K	3776	22	2/3	2	2	2	Y	Y	13	2	N	Y	Y	Y	Y

Note 1: See Section 1.3 "Details on Individual Family Devices", Section 4.6 "Deep Sleep Mode" and Section 27.3 "On-Chip Voltage Regulator" for details describing the functional differences between PIC18F and PIC18LF variants in this device family.

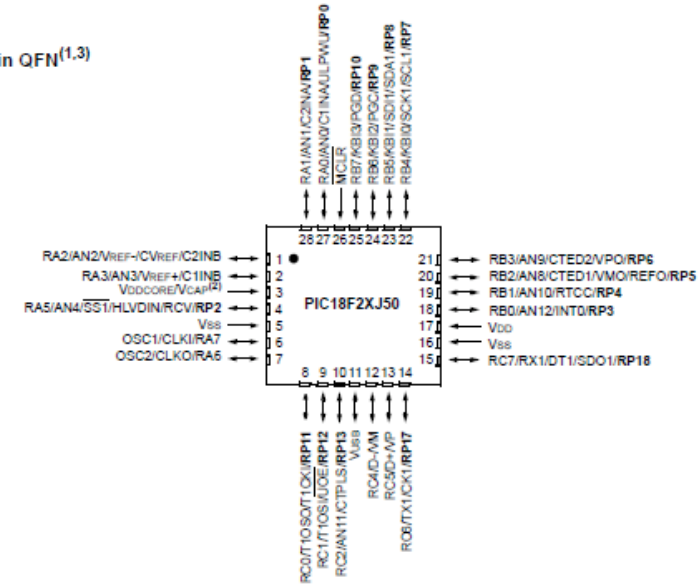
Pin Diagrams

28-Pin SPDIP/SOIC/SSOP(1)

= Pins are up to 5.5V tolerant



28-Pin QFN(1,3)



Legend: RPn represents remappable pins.

Note 1: Some input and output functions are routed through the Peripheral Pin Select (PPS) module and can be dynamically assigned to any of the RPn pins. For a list of the input and output functions, see Table 10-13 and Table 10-14, respectively. For details on configuring the PPS module, see **Section 10.7 "Peripheral Pin Select (PPS)"**.

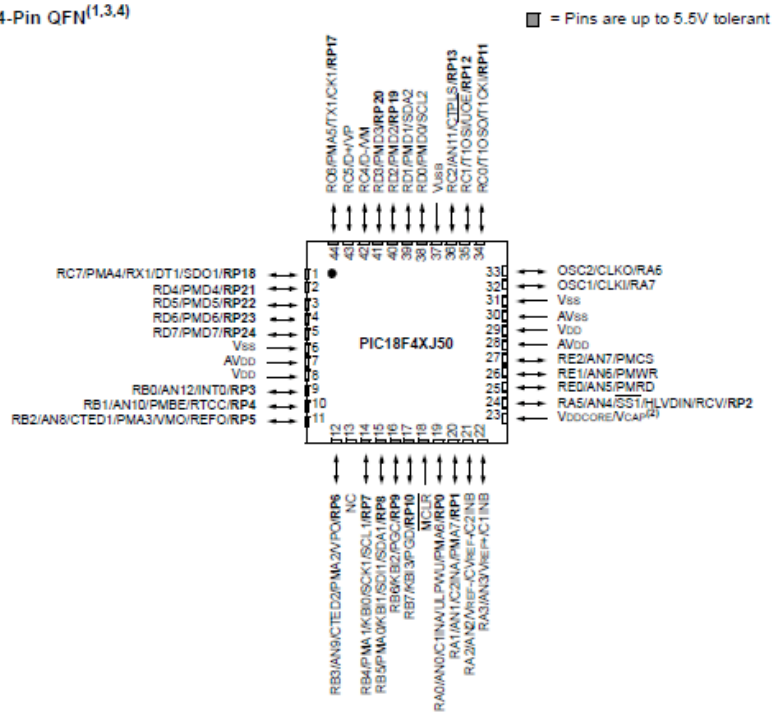
2: See **Section 27.3 "On-Chip Voltage Regulator"** for details on how to connect the VDDCORE/VCAP pin.

3: For the QFN package, it is recommended that the bottom pad be connected to VSS.

PIC18F46J50 FAMILY

Pin Diagrams (Continued)

44-Pin QFN(1,3,4)



Legend: RPn represents remappable pins.

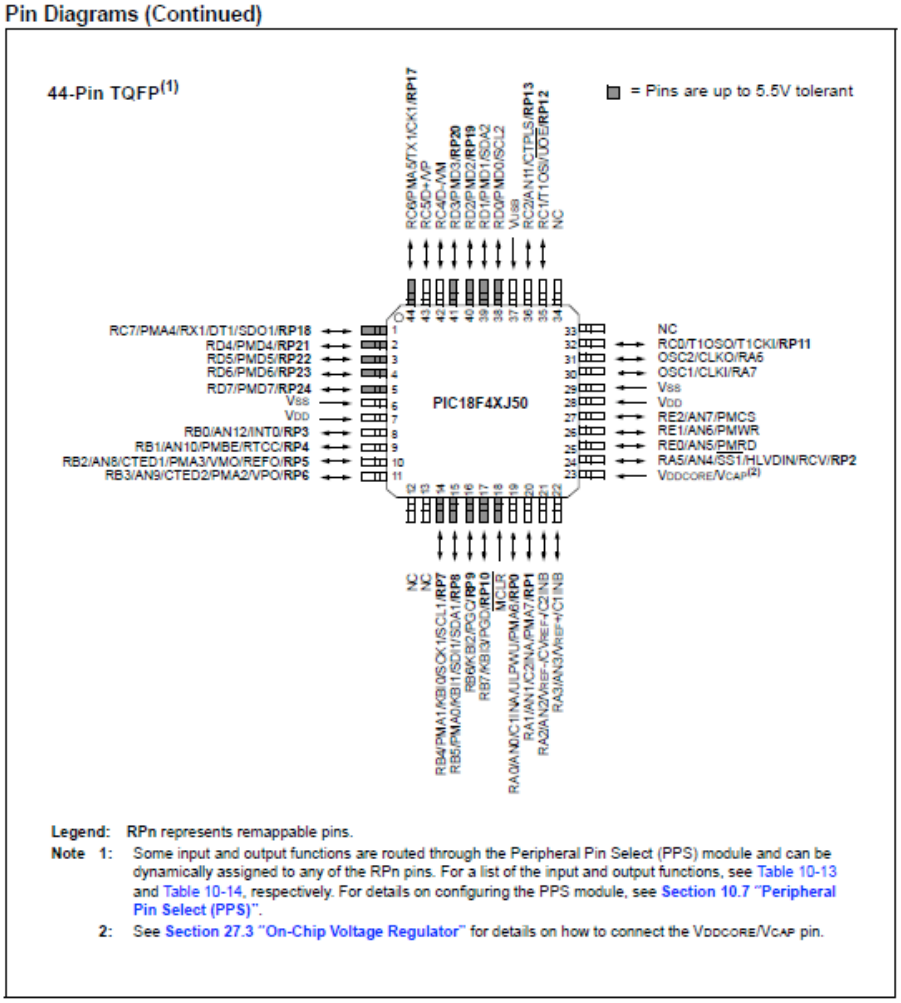
Note 1: Some input and output functions are routed through the Peripheral Pin Select (PPS) module and can be dynamically assigned to any of the RPn pins. For a list of the input and output functions, see Table 10-13 and Table 10-14, respectively. For details on configuring the PPS module, see Section 10.7 "Peripheral Pin Select (PPS)".

2: See Section 27.3 "On-Chip Voltage Regulator" for details on how to connect the VDDCORE/VCAP pin.

3: For the QFN package, it is recommended that the bottom pad be connected to VSS.

4: On 44-pin QFN devices, AVDD and AVSS reference sources are intended to be externally connected to VDD and VSS levels. Other package types tie AVDD and AVSS to VDD and VSS internally.

Pin Diagrams (Continued)



Legend: RPn represents remappable pins.

Note 1: Some input and output functions are routed through the Peripheral Pin Select (PPS) module and can be dynamically assigned to any of the RPn pins. For a list of the input and output functions, see Table 10-13 and Table 10-14, respectively. For details on configuring the PPS module, see **Section 10.7 "Peripheral Pin Select (PPS)"**.

2: See **Section 27.3 "On-Chip Voltage Regulator"** for details on how to connect the VDDCORE/VCAP pin.

1.0 DEVICE OVERVIEW

This document contains device-specific information for the following devices:

- PIC18F24J50
- PIC18F25J50
- PIC18F26J50
- PIC18F44J50
- PIC18F45J50
- PIC18F46J50
- PIC18LF24J50
- PIC18LF25J50
- PIC18LF26J50
- PIC18LF44J50
- PIC18LF45J50
- PIC18LF46J50

This family introduces a new line of low-voltage Universal Serial Bus (USB) microcontrollers with the main traditional advantage of all PIC18 microcontrollers, namely, high computational performance and a rich feature set at an extremely competitive price point. These features make the PIC18F46J50 family a logical choice for many high-performance applications, where cost is a primary consideration.

1.1 Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F46J50 family incorporate a range of features that can significantly reduce power consumption during operation. Key features are:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal RC oscillator, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operational requirements.
- **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the users to incorporate power-saving ideas into their application's software design.

1.1.2 UNIVERSAL SERIAL BUS (USB)

Devices in the PIC18F46J50 family incorporate a fully-featured USB communications module with a built-in transceiver that is compliant with the *"USB Specification Revision 2.0"*. The module supports both low-speed and full-speed communication for all supported data transfer types.

1.1.3 OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F46J50 family offer five different oscillator options, allowing users a range of choices in developing application hardware. These include:

- Two Crystal modes, using crystals or ceramic resonators.
- Two External Clock modes, offering the option of a divide-by-4 clock output.
- An internal oscillator block, which provides an 8 MHz clock and an INTRC source (approximately 31 kHz, stable over temperature and VDD), as well as a range of six user-selectable clock frequencies, between 125 kHz to 4 MHz, for a total of eight clock frequencies. This option frees an oscillator pin for use as an additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to the high-speed crystal, and external and internal oscillators, providing a clock speed up to 48 MHz.
- Dual clock operation, allowing the USB module to run from a high-frequency oscillator while the rest of the microcontroller is clocked at a different frequency.

The internal oscillator block provides a stable reference source that gives the PIC18F46J50 family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset (POR), or wake-up from Sleep mode, until the primary clock source is available.

1.1.4 EXPANDED MEMORY

The PIC18F46J50 family provides ample room for application code, from 16 Kbytes to 64 Kbytes of code space. The Flash cells for program memory are rated to last in excess of 10000 erase/write cycles. Data retention without refresh is conservatively estimated to be greater than 20 years.

The Flash program memory is readable and writable during normal operation. The PIC18F46J50 family also provides plenty of room for dynamic application data with up to 3.8 Kbytes of data RAM.

1.1.5 EXTENDED INSTRUCTION SET

The PIC18F46J50 family implements the optional extension to the PIC18 instruction set, adding eight new instructions and an Indexed Addressing mode. Enabled as a device configuration option, the extension has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.

1.1.6 EASY MIGRATION

Regardless of the memory size, all devices share the same rich set of peripherals, allowing for a smooth migration path as applications grow and evolve.

The consistent pinout scheme used throughout the entire family also aids in migrating to the next larger device.

The PIC18F46J50 family is also pin compatible with other PIC18 families, such as the PIC18F4550, PIC18F2450 and PIC18F45J10. This allows a new dimension to the evolution of applications, allowing developers to select different price points within Microchip's PIC18 portfolio, while maintaining the same feature set.

1.2 Other Special Features

- **Communications:** The PIC18F46J50 family incorporates a range of serial and parallel communication peripherals, including a fully featured USB communications module that is compliant with the *"USB Specification Revision 2.0"*. This device also includes two independent Enhanced USARTs and two Master Synchronous Serial Port (MSSP) modules, capable of both Serial Peripheral Interface (SPI) and I^2C™ (Master and Slave) modes of operation. The device also has a parallel port and can be configured to serve as either a Parallel Master Port (PMP) or as a Parallel Slave Port (PSP).
- **ECCP Modules:** All devices in the family incorporate three Enhanced Capture/Compare/PWM (ECCP) modules to maximize flexibility in control applications. Up to four different time bases may be used to perform several different operations at once. Each of the ECCPs offers up to four PWM outputs, allowing for a total of eight PWMs. The ECCPs also offer many beneficial features, including polarity selection, programmable dead time, auto-shutdown and restart and Half-Bridge and Full-Bridge Output modes.
- **10-Bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period, and thus, reducing code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit prescaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 30.0 "Electrical Characteristics"** for time-out periods.

1.3 Details on Individual Family Devices

Devices in the PIC18F46J50 family are available on 28-pin and 44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2. The devices are differentiated from each other in two ways:

- Flash program memory (three sizes: 16 Kbytes for the PIC18FX4J50, 32 Kbytes for PIC18FX5J50 devices and 64 Kbytes for PIC18FX6J50)
- I/O ports (three bidirectional ports on 28-pin devices, five bidirectional ports on 44-pin devices)

All other features for devices in this family are identical. These are summarized in Table 1-1 and Table 1-2.

The pinouts for the PIC18F2XJ50 devices are listed in Table 1-3. The pinouts for the PIC18F4XJ50 devices are shown in Table 1-4.

The PIC18F46J50 family of devices provides an on-chip voltage regulator to supply the correct voltage levels to the core. Parts designated with an "F" part number (such as PIC18F46J50) have the voltage regulator enabled.

These parts can run from 2.15V-3.6V on VDD, but should have the VDDCORE pin connected to VSS through a low-ESR capacitor. Parts designated with an "LF" part number (such as PIC18LF46J50) do not enable the voltage regulator. For "LF" parts, an external supply of 2.0V-2.7V has to be supplied to the VDDCORE pin while 2.0V-3.6V can be supplied to VDD (VDDCORE should never exceed VDD).

For more details about the internal voltage regulator, see **Section 27.3 "On-Chip Voltage Regulator"**.

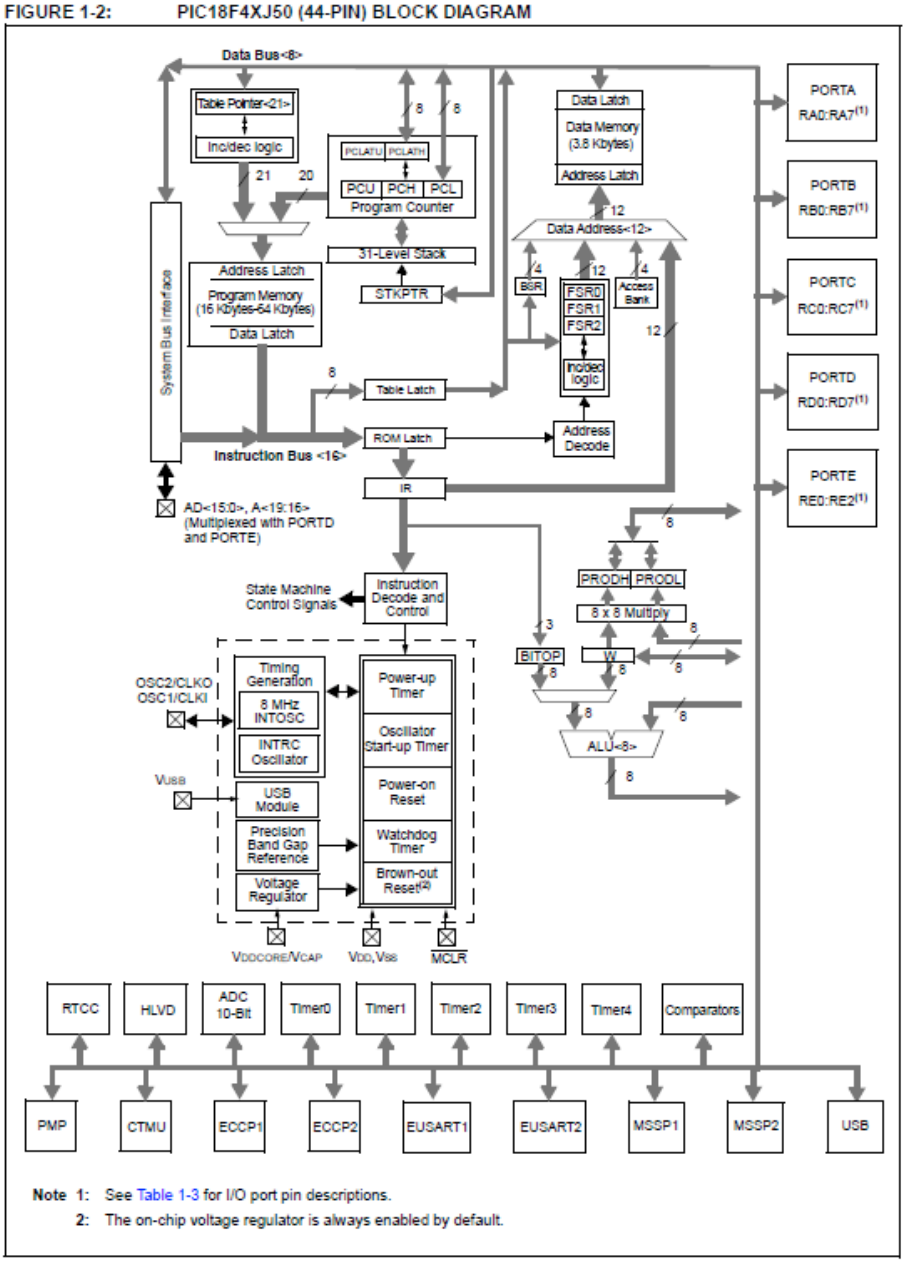
TABLE 1-1: DEVICE FEATURES FOR THE PIC18F2XJ50 (28-PIN DEVICES)

Features	PIC18F24J50	PIC18F25J50	PIC18F26J50
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	16K	32K	64K
Program Memory (Instructions)	8,192	16,384	32,768
Data Memory (Bytes)	3.8K	3.8K	3.8K
Interrupt Sources	30		
I/O Ports	Ports A, B, C		
Timers	5		
Enhanced Capture/Compare/PWM Modules	2		
Serial Communications	MSSP (2), Enhanced USART (2), USB		
Parallel Communications (PMP/PSP)	No		
10-Bit Analog-to-Digital Module	10 Input Channels		
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow, $\overline{\text{MCLR}}$, WDT (PWRT, OST)		
Instruction Set	75 Instructions, 83 with Extended Instruction Set Enabled		
Packages	28-Pin QFN, SOIC, SSOP and SPDIP (300 mil)		

TABLE 1-2: DEVICE FEATURES FOR THE PIC18F4XJ50 (44-PIN DEVICES)

Features	PIC18F44J50	PIC18F45J50	PIC18F46J50
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	16K	32K	64K
Program Memory (Instructions)	8,192	16,384	32,768
Data Memory (Bytes)	3.8K	3.8K	3.8K
Interrupt Sources	30		
I/O Ports	Ports A, B, C, D, E		
Timers	5		
Enhanced Capture/Compare/PWM Modules	2		
Serial Communications	MSSP (2), Enhanced USART (2), USB		
Parallel Communications (PMP/PSP)	Yes		
10-Bit Analog-to-Digital Module	13 Input Channels		
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow, $\overline{\text{MCLR}}$, WDT (PWRT, OST)		
Instruction Set	75 Instructions, 83 with Extended Instruction Set Enabled		
Packages	44-Pin QFN and TQFP		

FIGURE 1-2: PIC18F4XJ50 (44-PIN) BLOCK DIAGRAM



Note 1: See Table 1-3 for I/O port pin descriptions.

2: The on-chip voltage regulator is always enabled by default.

PIC18F46J50 FAMILY

TABLE 1-3: PIC18F2XJ50 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
MCLR	1	26	I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
OSC1/CLKI/RA7	9	6			Oscillator crystal or external clock input.
OSC1			I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; CMOS otherwise. Main oscillator input connection.
CLKI			I	CMOS	External clock source input; always associated with pin function, OSC1 (see related OSC1/CLKI pins).
RA7(1)			I/O	TTL	Digital I/O.
OSC2/CLKO/RA6	10	7			Oscillator crystal or clock output.
OSC2			O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O	—	Main oscillator feedback output connection. In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6(1)			I/O	TTL	Digital I/O.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

TABLE 1-3: PIC18F2XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number: 28-SPDIP/ SSOP/ SOIC	Pin Number: 28-QFN	Pin Type	Buffer Type	Description
					PORTA is a bidirectional I/O port.
RA0/AN0/C1INA/ULPWU/RP0	2	27			
RA0			I/O	DIG	Digital I/O.
AN0			I	Analog	Analog Input 0.
C1INA			I	Analog	Comparator 1 Input A.
ULPWU			I	Analog	Ultra Low-Power Wake-up input.
RP0			I/O	DIG	Remappable Peripheral Pin 0 input/output.
RA1/AN1/C2INA/RP1	3	28			
RA1			I	DIG	Digital I/O.
AN1			O	Analog	Analog Input 1.
C2INA			I	Analog	Comparator 2 Input A.
RP1			I/O	DIG	Remappable Peripheral Pin 1 input/output.
RA2/AN2/VREF-/CVREF/C2INB	4	1			
RA2			I/O	DIG	Digital I/O.
AN2			I	Analog	Analog Input 2.
VREF-			O	Analog	A/D reference voltage (low) input.
CVREF			I	Analog	Comparator reference voltage output.
C2INB			I	Analog	Comparator 2 Input B.
RA3/AN3/VREF+/C1INB	5	2			
RA3			I/O	DIG	Digital I/O.
AN3			I	Analog	Analog Input 3.
VREF+			I	Analog	A/D reference voltage (high) input.
C1INB			I	Analog	Comparator 1 Input B.
RA5/AN4/$\overline{SS1}$/HLVDIN/ RCV/RP2	7	4			
RA5			I/O	DIG	Digital I/O.
AN4			I	Analog	Analog Input 4.
$\overline{SS1}$			I	TTL	SPI slave select input.
HLVDIN			I	Analog	Low-Voltage Detect (LVD) input.
RCV			I	Analog	External USB transceiver RCV input.
RP2			I/O	DIG	Remappable Peripheral Pin 2 input/output.
RA6(1)					See the OSC2/CLKO/RA6 pin.
RA7(1)					See the OSC1/CLKI/RA7 pin.

Legend: TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels; I = Input; P = Power; DIG = Digital output; CMOS = CMOS compatible input or output; Analog = Analog input; O = Output; OD = Open-Drain (no P diode to VDD); I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

PIC18F46J50 FAMILY

TABLE 1-3: PIC18F2XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/RP3	21	18			
RB0			I/O	DIG	Digital I/O.
AN12			I	Analog	Analog Input 12.
INT0			I	ST	External Interrupt 0.
RP3			I/O	DIG	Remappable Peripheral Pin 3 input/output.
RB1/AN10/RTCC/RP4	22	19			
RB1			I/O	DIG	Digital I/O.
AN10			I	Analog	Analog Input 10.
RTCC			O	DIG	Real-Time Clock Calendar (RTCC) output.
RP4			I/O	DIG	Remappable Peripheral Pin 4 input/output.
RB2/AN8/CTED1/VMO/ REFO/RP5	23	20			
RB2			I/O	DIG	Digital I/O.
AN8			I	Analog	Analog Input 8.
CTED1			I	ST	CTMU Edge 1 input.
VMO			O	DIG	External USB transceiver D- data output.
REFO			O	DIG	Reference output clock.
RP5			I/O	DIG	Remappable Peripheral Pin 5 input/output.
RB3/AN9/CTED2/VPO/RP6	24	21			
RB3			I/O	DIG	Digital I/O.
AN9			I	Analog	Analog Input 9.
CTED2			I/O	ST	CTMU Edge 2 input.
VPO			O	DIG	External USB transceiver D+ data output.
RP6			I	DIG	Remappable Peripheral Pin 6 input/output.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

TABLE 1-3: PIC18F2XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number: 28-SPDIP/ SSOP/ SOIC	Pin Number: 28-QFN	Pin Type	Buffer Type	Description
					PORTB (continued)
RB4/KBI0/SCK1/SCL1/RP7	25	22			
RB4			I/O	DIG	Digital I/O.
KBI0			I	TTL	Interrupt-on-change pin.
SCK1			I/O	DIG	Synchronous serial clock input/output.
SCL1			I/O	I^2C	I^2C clock input/output.
RP7			I/O	DIG	Remappable Peripheral Pin 7 input/output.
RB5/KBI1/SDI1/SDA1/RP8	26	23			
RB5			I/O	DIG	Digital I/O.
KBI1			I	TTL	Interrupt-on-change pin.
SDI1			I	ST	SPI data input.
SDA1			I/O	I^2C	I^2C™ data input/output.
RP8			I/O	DIG	Remappable Peripheral Pin 8 input/output.
RB6/KBI2/PGC/RP9	27	24			
RB6			I/O	DIG	Digital I/O.
KBI2			I	TTL	Interrupt-on-change pin.
PGC			I	ST	ICSP™ clock input.
RP9			I/O	DIG	Remappable Peripheral Pin 9 input/output.
RB7/KBI3/PGD/RP10	28	25			
RB7			I/O	DIG	Digital I/O.
KBI3			I	TTL	Interrupt-on-change pin.
PGD			I/O	ST	In-Circuit Debugger and ICSP programming data pin.
RP10			I/O	DIG	Remappable Peripheral Pin 10 input/output.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

PIC18F46J50 FAMILY

TABLE 1-3: PIC18F2XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI/RP11	11	8			
RC0			I/O	ST	Digital I/O.
T1OSO			O	Analog	Timer1 oscillator output.
T1CKI			I	ST	Timer1 external digital clock input.
RP11			I/O	DIG	Remappable Peripheral Pin 11 input/output.
RC1/T1OSI/$\overline{UOE}$/RP12	12	9			
RC1			I/O	ST	Digital I/O.
T1OSI			I	Analog	Timer1 oscillator input.
$\overline{UOE}$			O	DIG	External USB transceiver NOE output.
RP12			I/O	DIG	Remappable Peripheral Pin 12 input/output.
RC2/AN11/CTPLS/RP13	13	10			
RC2			I/O	ST	Digital I/O.
AN11			I	Analog	Analog Input 11.
CTPLS			O	DIG	CTMU pulse generator output.
RP13			I/O	DIG	Remappable Peripheral Pin 13 input/output.
RC4/D-/VM	15	12			
RC4			I	TTL	Digital I.
D-			I/O	—	USB bus minus line input/output.
VM			I	TTL	External USB transceiver FM input.
RC5/D+/VP	16	13			
RC5			I	TTL	Digital I.
D+			I/O	DIG	USB bus plus line input/output.
VP			I	TTL	External USB transceiver VP input.
RC6/TX1/CK1/RP17	17	14			
RC6			I/O	ST	Digital I/O.
TX1			O	DIG	EUSART1 asynchronous transmit.
CK1			I/O	ST	EUSART1 synchronous clock (see related RX1/DT1).
RP17			I/O	DIG	Remappable Peripheral Pin 17 input/output.
RC7/RX1/DT1/SDO1/RP18	18	15			
RC7			I/O	ST	Digital I/O.
RX1			I	ST	Asynchronous serial receive data input.
DT1			I/O	ST	Synchronous serial data output/input.
SDO1			O	DIG	SPI data output.
RP18			I/O	DIG	Remappable Peripheral Pin 18 input/output.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

TABLE 1-3: PIC18F2XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
VSS1	8	5	P	—	Ground reference for logic and I/O pins.
VSS2	19	16	—	—	
VDD	20	17	P	—	Positive supply for peripheral digital logic and I/O pins.
VDDCORE/VCAP	6	3	—	—	Core logic power or external filter capacitor connection.
VDDCORE			P	—	Positive supply for microcontroller core logic (regulator disabled).
VCAP			P	—	External filter capacitor connection (regulator enabled).
VUSB	14	11	P	—	USB voltage input pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

PIC18F46J50 FAMILY

TABLE 1-4: PIC18F4XJ50 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
MCLR	18	18	I	ST	Master Clear (Reset) input; this is an active-low Reset to the device.
OSC1/CLKI/RA7	32	30			Oscillator crystal or external clock input.
OSC1			I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS. Main oscillator input connection.
CLKI			I	CMOS	External clock source input; always associated with pin function, OSC1 (see related OSC1/CLKI pins).
RA7(1)			I/O	TTL	Digital I/O.
OSC2/CLKO/RA6	33	31			Oscillator crystal or clock output.
OSC2			O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O	—	Main oscillator feedback output connection in RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6(1)			I/O	TTL	Digital I/O.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

TABLE 1-4: PIC18F4XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number 44-QFN	Pin Number 44-TQFP	Pin Type	Buffer Type	Description
					PORTA is a bidirectional I/O port.
RA0/AN0/C1INA/ULPWU/PMA6/RP0	19	19			
RA0			I/O	DIG	Digital I/O.
AN0			I	Analog	Analog Input 0.
C1INA			I	Analog	Comparator 1 Input A.
ULPWU			I	Analog	Ultra Low-Power Wake-up input.
PMA6			O	DIG	Parallel Master Port digital output.
RP0			I/O	DIG	Remappable Peripheral Pin 0 input/output.
RA1/AN1/C2INA/PMA7/RP1	20	20			
RA1			I	DIG	Digital I/O.
AN1			O	Analog	Analog Input 1.
C2INA			I	Analog	Comparator 2 Input A.
PMA7			O	DIG	Parallel Master Port digital output.
RP1			I/O	DIG	Remappable Peripheral Pin 1 input/output.
RA2/AN2/VREF-/CVREF/C2INB	21	21			
RA2			I/O	DIG	Digital I/O.
AN2			I	Analog	Analog Input 2.
VREF-			O	Analog	A/D reference voltage (low) input.
CVREF			I	Analog	Comparator reference voltage output.
C2INB			I	Analog	Comparator 2 Input B.
RA3/AN3/VREF+/C1INB	22	22			
RA3			I/O	DIG	Digital I/O.
AN3			I	Analog	Analog Input 3.
VREF+			I	Analog	A/D reference voltage (high) input.
C1INB			I	Analog	Comparator 1 Input B.
RA5/AN4/$\overline{SS1}$/HLVDIN/RCV/RP2	24	24			
RA5			I/O	DIG	Digital I/O.
AN4			I	Analog	Analog Input 4.
$\overline{SS1}$			I	TTL	SPI slave select input.
HLVDIN			I	Analog	Low-Voltage Detect (LVD) input.
RCV			I	Analog	External USB transceiver RCV input.
RP2			I/O	DIG	Remappable Peripheral Pin 2 input/output.
RA6(1)					See the OSC2/CLKO/RA6 pin.
RA7(1)					See the OSC1/CLKI/RA7 pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

PIC18F46J50 FAMILY

TABLE 1-4: PIC18F4XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/RP3	9	8			
RB0			I/O	DIG	Digital I/O.
AN12			I	Analog	Analog Input 12.
INT0			I	ST	External Interrupt 0.
RP3			I/O	DIG	Remappable Peripheral Pin 3 input/output.
RB1/AN10/PMBE/RTCC/RP4	10	9			
RB1			I/O	DIG	Digital I/O.
AN10			I	Analog	Analog Input 10.
PMBE			O	DIG	Parallel Master Port byte enable.
RTCC			O	DIG	Real-Time Clock Calendar (RTCC) output.
RP4			I/O	DIG	Remappable Peripheral Pin 4 Input/output.
RB2/AN8/CTED1/PMA3/VMO/ REFO/RP5	11	10			
RB2			I/O	DIG	Digital I/O.
AN8			I	Analog	Analog Input 8.
CTED1			I	ST	CTMU Edge 1 input.
PMA3			O	DIG	Parallel Master Port address.
VMO			O	DIG	External USB transceiver D- data output.
REFO			O	DIG	Reference output clock.
RP5			I/O	DIG	Remappable Peripheral Pin 5 input/output.
RB3/AN9/CTED2/PMA2/VPO/ RP6	12	11			
RB3			I/O	DIG	Digital I/O.
AN9			I	Analog	Analog Input 9.
CTED2			I	ST	CTMU Edge 2 input.
PMA2			O	DIG	Parallel Master Port address.
VPO			O	DIG	External USB transceiver D+ data output.
RP6			I/O	DIG	Remappable Peripheral Pin 6 input/output.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

TABLE 1-4: PIC18F4XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number 44-QFN	Pin Number 44-TQFP	Pin Type	Buffer Type	Description
					PORTB (continued)
RB4/PMA1/KBI0/SCK1/SCL1/RP7	14	14			
RB4			I/O	DIG	Digital I/O.
PMA1			I/O	DIG	Parallel Master Port address.
KBI0			I	TTL	Interrupt-on-change pin.
SCK1			I/O	DIG	Synchronous serial clock input/output.
SCL1			I/O	I^2C	I^2C clock input/output.
RP7			I/O	DIG	Remappable Peripheral Pin 7 input/output.
RB5/PMA0/KBI1/SDI1/SDA1/RP8	15	15			
RB5			I/O	DIG	Digital I/O.
PMA0			I/O	DIG	Parallel Master Port address.
KBI1			I	TTL	Interrupt-on-change pin.
SDI1			I	ST	SPI data input.
SDA1			I/O	I^2C	I^2C™ data input/output.
RP8			I/O	DIG	Remappable Peripheral Pin 8 input/output.
RB6/KBI2/PGC/RP9	16	16			
RB6			I/O	DIG	Digital I/O.
KBI2			I	TTL	Interrupt-on-change pin.
PGC			I	ST	ICSP™ clock input.
RP9			I/O	DIG	Remappable Peripheral Pin 9 input/output.
RB7/KBI3/PGD/RP10	17	17			
RB7			I/O	DIG	Digital I/O.
KBI3			I	TTL	Interrupt-on-change pin.
PGD			I/O	ST	In-Circuit Debugger and ICSP programming data pin.
RP10			I/O	DIG	Remappable Peripheral Pin 10 input/output.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

PIC18F46J50 FAMILY

TABLE 1-4: PIC18F4XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number 44-QFN	Pin Number 44-TQFP	Pin Type	Buffer Type	Description
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI/RP11	34	32			
RC0			I/O	ST	Digital I/O.
T1OSO			O	Analog	Timer1 oscillator output.
T1CKI			I	ST	Timer1/Timer3 external clock input.
RP11			I/O	DIG	Remappable Peripheral Pin 11 input/output.
RC1/T1OSI/$\overline{UOE}$/RP12	35	35			
RC1			I/O	ST	Digital I/O.
T1OSI			I	Analog	Timer1 oscillator input.
$\overline{UOE}$			O	DIG	External USB transceiver NOE output.
RP12			I/O	DIG	Remappable Peripheral Pin 12 input/output.
RC2/AN11/CTPLS/RP13	36	36			
RC2			I/O	ST	Digital I/O.
AN11			I	Analog	Analog Input 11.
CTPLS			O	DIG	CTMU pulse generator output.
RP13			I/O	DIG	Remappable Peripheral Pin 13 input/output.
RC4/D-/VM	42	42			
RC4			I	TTL	Digital I.
D-			O	—	USB bus minus line input/output.
VM			I	TTL	External USB transceiver FM input.
RC5/D+/VP	43	43			
RC5			I	TTL	Digital I.
D+			I/O	DIG	USB bus plus line input/output.
VP			I	TTL	External USB transceiver VP input.
RC6/PMA5/TX1/CK1/RP17	44	44			
RC6			I/O	ST	Digital I/O.
PMA5			O	DIG	Parallel Master Port address.
TX1			O	DIG	EUSART1 asynchronous transmit.
CK1			I/O	ST	EUSART1 synchronous clock (see related RX1/DT1).
RP17			I/O	DIG	Remappable Peripheral Pin 17 input/output.
RC7/PMA4/RX1/DT1/SDO1/RP18	1	1			
RC7			I/O	ST	Digital I/O.
PMA4			O	DIG	Parallel Master Port address.
RX1			I	ST	EUSART1 asynchronous receive.
DT1			I/O	ST	EUSART1 synchronous data output/input (see related TX1/CK1).
SDO1			O	DIG	SPI data output.
RP18			I/O	DIG	Remappable Peripheral Pin 18 input/output.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

TABLE 1-4: PIC18F4XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number 44-QFN	Pin Number 44-TQFP	Pin Type	Buffer Type	Description
					PORTD is a bidirectional I/O port.
RD0/PMD0/SCL2	38	38			
RD0			I/O	ST	Digital I/O.
PMD0			I/O	DIG	Parallel Master Port data.
SCL2			I/O	DIG	I^2C™ data input/output.
RD1/PMD1/SDA2	39	39			
RD1			I/O	ST	Digital I/O.
PMD1			I/O	DIG	Parallel Master Port data.
SDA2			I/O	DIG	I^2C data input/output.
RD2/PMD2/RP19	40	40			
RD2			I/O	ST	Digital I/O.
PMD2			I/O	DIG	Parallel Master Port data.
RP19			I/O	DIG	Remappable Peripheral Pin 19 input/output.
RD3/PMD3/RP20	41	41			
RD3			I/O	ST	Digital I/O.
PMD3			I/O	DIG	Parallel Master Port data.
RP20			I/O	DIG	Remappable Peripheral Pin 20 input/output.
RD4/PMD4/RP21	2	2			
RD4			I/O	ST	Digital I/O.
PMD4			I/O	DIG	Parallel Master Port data.
RP21			I/O	DIG	Remappable Peripheral Pin 21 input/output.
RD5/PMD5/RP22	3	3			
RD5			I/O	ST	Digital I/O.
PMD5			I/O	DIG	Parallel Master Port data.
RP22			I/O	DIG	Remappable Peripheral Pin 22 input/output.
RD6/PMD6/RP23	4	4			
RD6			I/O	ST	Digital I/O.
PMD6			I/O	DIG	Parallel Master Port data.
RP23			I/O	DIG	Remappable Peripheral Pin 23 input/output.
RD7/PMD7/RP24	5	5			
RD7			I/O	ST	Digital I/O.
PMD7			I/O	DIG	Parallel Master Port data.
RP24			I/O	DIG	Remappable Peripheral Pin 24 input/output.

Legend: TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels; I = Input; P = Power; DIG = Digital output; CMOS = CMOS compatible input or output; Analog = Analog input; O = Output; OD = Open-Drain (no P diode to VDD); I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

PIC18F46J50 FAMILY

TABLE 1-4: PIC18F4XJ50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTE is a bidirectional I/O port.
RE0/AN5/PMRD	25	25			
RE0			I/O	ST	Digital I/O.
AN5			I	Analog	Analog Input 5.
PMRD			I/O	DIG	Parallel Master Port input/output.
RE1/AN6/PMWR	26	26			
RE1			I/O	ST	Digital I/O.
AN6			I	Analog	Analog Input 6.
PMWR			I/O	DIG	Parallel Master Port write strobe.
RE2/AN7/PMCS	27	27			
RE2			I/O	ST	Digital I/O.
AN7			I	Analog	Analog Input 7.
PMCS			O	—	Parallel Master Port chip select.
VSS1	6	6	P	—	Ground reference for logic and I/O pins.
VSS2	31	29	—	—	
AVSS1	30	—	P	—	Ground reference for analog modules.
VDD1	8	7	P	—	Positive supply for peripheral digital logic and I/O pins.
VDD2	29	28	P	—	
VDDCORE/VCAP	23	23			Core logic power or external filter capacitor connection.
VDDCORE			P	—	Positive supply for microcontroller core logic (regulator disabled).
VCAP			P	—	External filter capacitor connection (regulator enabled).
AVDD1	7	—	P	—	Positive supply for analog modules.
AVDD2	28	—	—	—	Positive supply for analog modules.
VUSB	37	37	P	—	USB voltage input pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output; I^2C™ = Open-Drain, I^2C-specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

9.10 Annex 10. MPLAB starter kit

MPLAB Starter Kit for PIC18F User's Guide

 DS51852A

MICROCHIP

MPLAB STARTER KIT FOR PIC18F
USER'S GUIDE

Chapter 1. Introduction to the Starter Kit

Thank you for purchasing Microchip Technology's MPLAB Starter Kit for PIC18F. This board is intended to introduce and demonstrate the capabilities and features of the PIC18F J-series of Flash microcontrollers. In addition, the starter kit has on-board, in-circuit debug circuitry so that you may develop and debug your own applications.

This chapter introduces the starter kit and provides an overview of its features. Topics covered include:

- Overview
- Operational Requirements
- Initial Board Setup

1.1 OVERVIEW

The MPLAB Starter Kit for PIC18F provides an all-in-one solution for debugging and programming applications using Microchip's own MPLAB Integrated Development Environment (IDE). A USB connection to a host computer supplies communications and power to the board; no additional external power supply is needed.

The starter kit includes integrated debug and programmer circuitry that allows applications to be programmed onto the application side of the PIC18F MCU and then debugged, all using MPLAB IDE. The need for an additional programmer or hardware interface has been completely eliminated.

The application side of the starter kit contains a range of hardware components to demonstrate the utility and processing power of Microchip's PIC18F46J50 family USB microcontrollers.

1.2 OPERATIONAL REQUIREMENTS

To communicate with and program the starter kit, the following hardware and software requirements must be met:

- PC compatible system with CD-ROM drive
- One available USB port on the PC or a powered USB hub
- MicroSD flash memory card (a card preloaded with demo applications is provided)
- Microsoft® Windows® 2000 SP4, Windows XP SP2 or Windows Vista® (32-bit)

Note: Only initial testing has been performed on 32-bit Windows Vista for this release of the demo applications. The 64-bit version is not supported at this time.

1.3 INITIAL BOARD SETUP

With its pre-installed demo application, the MPLAB Starter Kit for PIC18F is designed to be used straight out of the box. Except for a single connection to a computer, no additional hardware or configuration is necessary.

1.3.1 Installing the Software

Before connecting the starter kit to any computer for the first time, it is important to install the accompanying software on the MPLAB Starter Kit for PIC18F CD first. This ensures that both the MPLAB development environment and C compiler, as well as the relevant tutorial and help files for the starter kit, are ready to use when the board is connected.

To install the software, insert the starter kit CD into the CD-ROM drive. The installation process starts automatically. The process pauses for user responses to accept the Microchip software licenses and to confirm the installation directories; respond appropriately.

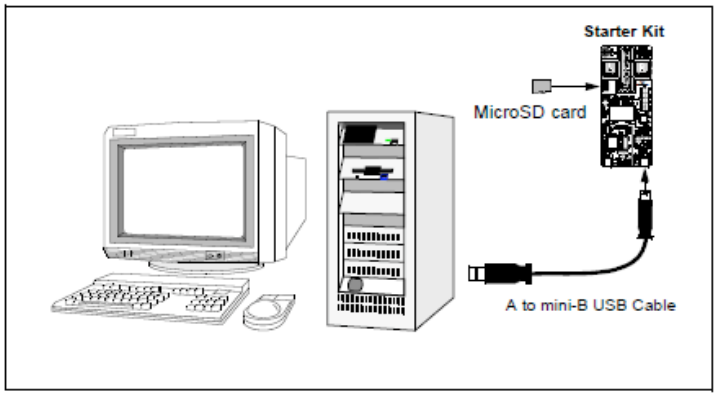
1.3.2 Connecting the Hardware

Once the starter kit software is installed, connect the provided USB cable (A to mini-B) to any available USB port on the PC or powered hub, then to the starter kit at the mini-B receptacle on the application side of the board (on the right side of the board's lower edge, as shown in Figure 1-1). The PC USB connection provides communication and power to the board. The MicroSD™ Flash card, used to store the demo applications, may be inserted into the MicroSD slot at any time.

If the cable is connected correctly, the green power LED is lit. The OLED display will display the "Microchip PIC18F Starter Kit" main menu. At this point, the application waits for the bootloader Reset switch to be pressed. When it is pressed, the bootloader application is loaded from the MicroSD card and executed. At this point, you can use the "Up" and "Down" touch pads to scroll through the available menus, and the "Cancel" and "Accept" pads to select menu items. Refer to **Chapter 2. "The Demonstration Application"** for a complete description of how to use the bootloader application.

When one of the precompiled applications is executed, a sequence of pop-up balloons in the system tray (lower right of desktop) should appear, stating that (1) new hardware has been found, (2) drivers are being installed and (3) the new hardware is ready for use. If you do not see these messages and the starter kit does not work, try unplugging and reconnecting the USB. If this does not work, refer to **Section Chapter 3. "Developing an Application"**.

FIGURE 1-1: STARTER KIT SETUP



9.11 Annex 11. Codi de Recepció