Resum

Aquest projecte té com a objectiu crear un software per PC que permeti programar mitjançant el bus CAN *(Controller Area Network)* microcontroladors de la família *PIC18FXX8* de *Microchip*.

Per aquests microcontroladors, *Microchip* subministra un firmware (programa fix al microcontrolador) anomenat *Bootloader*. Aquest firmware permet llegir o escriure qualsevol regió de memòria del microcontrolador, interpretant els missatges CAN que se li envien. Abans de poder usar el *Bootloader*, cal activar-lo. Per fer-ho, cal escriure una dada a l'última posició de la memòria EEPROM *(Electrically Erasable Programmable Read-Only Memory)*.

Aquesta característica és molt útil per programar sistemes en funcionament, ja que enviant missatges podem transferir nou software per a cada microcontrolador connectat a la xarxa CAN. També permet programar al mateix temps diversos microcontroladors amb el mateix software, si aquests tenen el *Bootloader* amb la mateixa configuració. Això redueix els temps de reprogramació, ja que no és necessari desmuntar un microcontrolador per ser reprogramat. A més, com que només deixa de funcionar el microcontrolador que estem programant, la resta de la xarxa pot seguir funcionant.

També és útil en sistemes nous. Les empreses d'automòbils usen aquest sistema per programar les centraletes de vehicles un cop connectades al bus. Així s'estalvien manejar grans estocs de centraletes preprogramades segons les opcions escollides pel client.

Aquest projecte és integrament de software. Les proves de funcionament s'han fet amb hardware subministrat pel Departament d'Enginyeria Electrònica. Aquest hardware consta de: plaques amb microcontroladors *PIC18FXX8* amb connexió CAN, un hardware comercial de l'empresa *Kvaser* que permet al PC accedir al bus i un programador de *Microchip.*

Primerament, s'ha modificat el *Bootloader* subministrat per *Microchip* perquè es pugui configurar més fàcilment, mitjançant un programa de PC a tal fi. També s'ha canviat el mètode d'arrencada perquè s'activi al rebre un missatge CAN, sense escriure a la EEPROM. Un cop el *Bootloader* ha rebut el missatge d'activació, es pot començar a programar. Aquest nou firmware és compatible amb els microcontroladors de la família PIC18FXX8.

Per enviar les dades de programació, s'ha desenvolupat un altre programa per PC. Aquest programa activa el *Bootloader*, envia el nou programa, comprova les dades escrites i treu del mode *Boot* el microcontrolador. El programa és compatible amb qualsevol hardware de PC d'accés a bus CAN de *Kvaser*. Els dos programes per PC s'han desenvolupat en Visual Basic, ja que és un llenguatge que facilita la programació en entorn gràfic.

Sumari

RESUM 1

SUMARI 3

1. GLOSSARI 7

2. ÍNDEX DE FIGURES 9

3. ÍNDEX DE TAULES 10

4. PREFACI 12

4.1. Origen del projecte 12
4.2. Motivació 12
4.3. Requeriments previs 12

5. INTRODUCCIÓ 13

5.1. Objectius del projecte 13
5.2. Abast del projecte 13

6. ANTECEDENTS I ESTAT DE L'ART 14

6.1. Descripció del conflicte 14
6.2. Avantatges i inconvenients 15
6.3. Tipus de connexions de PC a CAN 15
6.4. Exemples de sistemes de programació via bus 16

7. ALTERNATIVES 17

7.1. Microprocessador amb *Bootloader* CAN 17
7.2. Targeta de PC 18
7.3. Llenguatge de programació 18
7.4. Sistema operatiu 19

8. SOLUCIÓ ADOPTADA 20

8.1. *Microchip* PIC18F458 20
8.2. *Kvaser Leaf light HS* 20
8.3. *Microsoft Visual Basic 6.0* 20
8.4. Sistema operatiu de PC 20
8.5. Sistema complet 21

9. EL *BOOTLOADER* MODIFICAT 22

9.1. Missatges enviats per la font de programació 22
9.2. Missatges enviats pel *Bootloader* a la font de programació 23

9.3. Significat de les dades dels missatges CAN 24
9.3.1. Adreces (ADDRL, ADDRH i ADDRU) 25
9.3.2. *Control bits* 25
9.3.3. *Comandaments especials* 26
9.3.4. Dades dels *Comandaments especials* (CPDTL i CPDTH) 27
9.3.5. Dades (DATA 0-7) 27
9.4. Funcions del *Bootloader* 27
9.5. Inici del *Bootloader* 28
9.6. Diagrama de flux del *Bootloader* en mode *PUT* 29
9.7. Diagrama de flux del Bootloader en mode PUTGET 30
9.8. Opcions de compilació del *Bootloader* 31

10. CONFIGURACIÓ DEL *BOOTLOADER* 32

10.1. Fitxer *BOOTDATA* 32
10.2. Formulari principal del programa de definicions 33
10.3. Configuració dels Paràmetres del CAN 34
10.3.1. Registres CAN del microcontrolador 35
10.3.1.1. *BRGCON1* 36
10.3.1.2. *BRGCON2* 36
10.3.1.3. *BRGCON3* 36
10.3.2 Combinacions sense error 36
10.3.3. Paràmetres CAN per defecte 37
10.4. Configuració del mode de funcionament del *Bootloader* 37
10.5. Configuració de la comunicació entre *Bootloader* i PC. 37
10.5.1Selecció del tipus de missatge 38
10.5.2. Selecció dels registres dels bits *Put-get* i *Control-data* 38
10.5.3. Selecció del filtre de recepció i de l'identificador de transmissió 39
10.5.4. Selecció de la màscara de recepció 40
10.6. Configuració de l'arrencada del *Bootloader* 40
10.7. Botons per executar comandes 41
10.8. Aspecte dels fitxers 41

11. PROGRAMADOR CAN 43

11.1. Finestra principal del programador 43
11.2. Menús i submenús 44
11.3. Barra d'eines 47
11.4. Paràmetres de configuració de la targeta de PC 47
11.5. Dades del fitxer *BOOTDATA* carregat 48
11.6. Dades del fitxer *HEX* carregat 48
11.7. Barres de progressió 48
11.8. Barra d'estat 48

11.9. El fitxer *.HEX* 49
11.10. Entrada en mode *Boot* 49
11.11. Mètode de Programació 50
11.12. Mètode d' esborrament 53
11.13. Mètode de lectura 53
11.14. Sortir del mode *Boot* 54
11.15. Gestió d'errors 54
11.16. Usabilitat 55

12. TEST DEL SISTEMA 56
12.1. Proves 56
12.2. Taula de temps per les diferents operacions 57

13. PROPOSTES I RECOMANACIONS 59
13.1. Propostes 59
13.2. Recomanacions 59
13.2.1. Elecció dels paràmetres CAN. 59
13.2.2. Elecció del mode de funcionament del *Bootloader* 59
13.2.3. Consells d'utilització del programador 59

14. COSTOS 60
14.1.Cost del projecte 60
14.2. Cost per l'empresa d'automòbils 61
14.2.1. Inversió inicial empresa 61
14.2.2. Estalvi empresa 62
14.2.3. Rendibilitat de la inversió 63

15. IMPACTE AMBIENTAL 64

CONCLUSIONS 65

AGRAÏMENTS 67

BIBLIOGRAFIA 69
Referències bibliogràfiques 69
Bibliografia complementària 69

1. Glossari

CAN(Controller area Network): Protocol de comunicació creat per BOSCH per comunicar les unitats de control electròniques dels vehicles.

Bootloader: Programa que resideix en un microcontrolador i que permet autoprogramar-lo.

Microcontrolador: Circuit integrat que conté CPU, memòria i perifèrics.

ECU(Electronic control unit) o Centraleta: Computadors encarregats de gestionar diferents sistemes electrònics del automòbil.

EEPROM(Electrically Erasable Programmable Read-Only Memory): És un tipus de memòria ROM que pot ser programat, esborrat i reprogramat elèctricament. No perden la informació si es desconnecten de l'alimentació.

Memòria flash: Tipus de memòria EEPROM que permet que múltiples posicions de memòria siguin escrites o esborrades en una mateixa operació, són més ràpides que la resta d'EEPROM.

SDK(Software Development Kit): Eines de desenvolupament que permet a un programador crear aplicacions.

PCMCIA(Personal Computer Memory Card International Association): Estàndard de hardware per l'expansió de funcionalitats d' ordinadors portàtils.

Express-card: Estàndard de hardware que substitueix les targetes PCMCIA.

Wifi(Wireless Fidelity): Protocol normalitzat de comunicació sense fils per a ordenadors portàtils.

Bluetooth: Protocol normalitzat de comunicació sense fils per aparells electrònics portàtils.

IDE(Integrated Development Environment): Entorn de desenvolupament integrat, són un conjunt d'eines que faciliten la creació de software als programadors.

Clock: Senyal de rellotge del Microcontrolador. La velocitat d'execució de les instruccions depèn de la seva freqüència.

ICSP(In Circuit Serial Programming): Tècnica implementada per *Microchip* per programar dispositius un cop connectats al circuit.

Compilar: Traduir un programa del llenguatge de programació a codi màquina.

ACK(Acknowledge): Tipus de missatge del *Bootloader* que serveix per sincronitzar-se amb la font de programació.

Apuntador: És una variable que fa referència a una regió de memòria.

Checksum: Suma de dades amb intenció de comprovar la seva integritat.

Reset: Estat inicial d'arrencada d'un microcontrolador.

Flag: Bit que s'utilitza per senyalar algun estat del microcontrolador.

Llenguatge assemblador: Llenguatge de baix nivell que permet aprofitar al màxim les capacitats d'un microcontrolador.

T_Q*(Time Quantum):* Unitat de temps equivalent a un senyal de rellotge del controlador CAN.

BRP(Baud Rate Prescaler): Relació entre la senyal de rellotge del microcontrolador i la del controlador CAN.

C/D Rx: Bit *Control-data* de l'identificador de recepció del *Bootloader*.

P/G Rx: Bit *Put-get* de l'identificador de recepció del *Bootloader*.

C/D Tx: Bit *Control-data* de l'identificador de transmissió del *Bootloader*.

2. ÍNDEX DE FIGURES

Fig. 6.1. Esquema de la programació de centraletes a través del bus CAN 14

Fig. 8.1. Esquema del sistema complet 21

Fig. 9.1. Diagrama de flux del *Bootloader* en mode *PUT* 29

Fig. 9.2. . Diagrama de flux del *Bootloader* en mode *PUTGET* 30

Fig. 10.1. Formulari principal del programa de configuració del *Bootloader* 33

Fig. 10.2. Transmissió d'un bit CAN [3] 35

Fig. 10.3. Esquema del mòdul CAN del *PIC18FXX8* [2] 38

Fig. 10.4. Imatge del fitxer *BOOTDATA*.txt 41

Fig. 10.5. Imatge del fitxer *CANBOOT*.def 42

Fig. 11.1. Formulari principal del programador CAN 43

Fig. 11.2. Diagrama de flux de l'habilitació dels menús 46

Fig. 11.3. Diagrama de flux de l'entrada en mode *Boot* en mode *PUTGET* 50

Fig. 11.4. Diagrama de flux d' escriure una posició de memòria en mode *PUTGET* 51

Fig. 11.5. Diagrama de flux d' escriure una posició de memòria en mode *PUT* 52

Fig. 11.6. Diagrama de flux de llegir una posició de memòria 54

3. ÍNDEX DE TAULES

Taula 7.1. Microcontroladors de 8 bits que compleixen les especificacions 17

Taula 7.2. Targetes PC-CAN que compleixen les especificacions 18

Taula 9.1. Tipus de missatges CAN que s'envien al *Bootloader* 22

Taula 9.2. Dades dels missatges enviats al *Bootloader* 23

Taula 9.3. Tipus de missatges CAN que envia el *Bootloader* 23

Taula 9.4. Dades enviades pel *Bootloader* 24

Taula 9.5. Significat de les dades 24

Taula 9.6. Mapa de les memòries 25

Taula 9.7. *Control bits* 26

Taula 9.8. *Comandaments especials* 26

Taula 9.9. Funcions realitzades pel *Bootloader* 27

Taula 9.10. Paràmetres de configuració del *Bootloader* 31

Taula 10.1. Informació continguda en el fitxer *BOOTDATA*.txt 32

Taula10. 2. Registres del mòdul CAN del microcontrolador [2] 35

Taula 10.3. Paràmetres possibles de configuració 37

Taula 10.4. Paràmetres CAN per defecte 37

Taula 10.5. Registres d'identificadors de missatges estàndard 39

Taula 10.6. Registres d'identificadors de missatges estesos 39

Taula 11.1. Valors predefinits de la targeta *Kvaser* 47

Taula 12.1. Temps de les operacions en mode *PUT* 57

Taula 12.2. Temps de les operacions en mode *PUT* 58

Taula 14.1. Cost de les amortitzacions del material 60

Taula 14.2. Cost del personal 60

Taula 14.3. Cost total del projecte 61

Taula 14.4. Inversió inicial empresa 61

Taula 14.5. Estalvi anual de l'empresa 62

Taula 14.6. Estalvi total de l'empresa 62

Taula 14.7. Rendibilitat de la inversió 63

4. PREFACI

4.1. Origen del projecte

Aquest projecte sorgeix de la necessitat de programar les centraletes de vehicles connectades a un bus CAN durant l'assemblatge del vehicle. Un dels mètodes més usat pels fabricants és la programació a través del bus. El programa és enviat pel bus i la centraleta s'encarrega d'escriure'l a la seva pròpia memòria.

L'objectiu d' aquest projecte és la programació de les centraletes de control de la família PIC18FXX de *Microchip* a través del bus CAN. Per aconseguir-ho s'han creat 2 programes per a PC, que permeten programar les centraletes ràpidament. D'aquesta manera, igual que en les fàbriques de vehicles, no cal programar la centraleta abans de muntar-la si no que es programa un cop muntada i connectada al bus.

Actualment, en el mercat ja existeixen programes comercials que realitzen aquestes funcions, però tenen un cost molt elevat. És per aquest motiu, que s'ha decidit desenvolupar un software a mida de les necessitats.

4.2. Motivació

La motivació d'aquest projecte apareix de la possibilitat d'adquirir nous coneixements sobre electrònica de l'automòbil, programació de microcontroladors i programació de PC. Aquests coneixements adquirits durant la realització del projecte s'utilitzen en diferents camps de l'enginyeria. Per tant, seran molt útils en la propera vida laboral com enginyer.

4.3. Requeriments previs

Per la realització d'aquest projecte es requereixen coneixements previs sobre:

- El funcionament del bus CAN [3], [6] i [7].
- Les característiques tècniques de la centraleta i del seu *Bootloader* [1] i [2].
- L' entorn de programació Visual Basic 6.0 [4].
- Les característiques tècniques del hardware que s'usarà per connectar a la xarxa CAN al PC.[8]
- Les funcions de les eines de desenvolupament de software (SDK) del hardware usat per connectar el PC a la xarxa CAN[5].

5. INTRODUCCIÓ

5.1. Objectius del projecte

L'objectiu principal del projecte és poder programar centraletes de la família *PIC18FXX8* de *Microchip* des d'un PC a través del bus CAN que les connecta. Per poder aconseguir aquest objectiu s'han de realitzar les següents tasques:

- Realitzar un anàlisi de les diferents alternatives disponibles actualment per a programar a través del bus CAN.
- Modificar el *Bootloader* subministrat per *Microchip* perquè permeti accedir al mode de programació a través d'un missatge CAN i es pugui configurar fàcilment. Per fer-ho s'utilitzarà l'eina MPLAB subministrada per *Microchip*.
- Crear un programa de PC en Visual Basic per definir les opcions del *Bootloader*.
- Crear un programa de PC en Visual Basic per programar els microcontroladors a través del Bus.
- Realitzar les proves de funcionament del *Bootloader* i dels programes de PC.
- Anàlisi temporal i de limitacions.

5.2. Abast del projecte

Aquest projecte només permet programar centraletes de la família *PIC18FXX8* de *Microchip* amb hardware de PC de l'empresa *Kvaser*. Altres limitacions són:

- El *Bootloader* ha d'estar confinat dins del *Boot Block* del microcontrolador, ja que així queda protegit i no pot ser modificat involuntàriament. Per modificar-lo, s'ha d'utilitzar un programador i el software de PC que subministra *Microchip*. Si es modifiquen alguns dels paràmetres de configuració del microcontrolador amb el *Bootloader*, aquest pot deixar de funcionar.
- El programa de PC per configurar el *Bootloader* permet escollir entre 4 velocitats de bus i 4 de Microcontrolador. Les opcions de configuració dels paràmetres de CAN s'han limitat als òptims del microcontrolador. Aquest programa, només es pot usar en un PC amb sistema operatiu Windows.
- El programa de PC per programar les centraletes només accepta programes de microcontrolador en format *HEX* creats amb l' MPLAB de *Microchip*. Per funcionar correctament necessita saber la configuració del *Bootloader* de la centraleta a programar. Aquest programa, només es pot usar en un PC amb sistema operatiu Windows.

6. ANTECEDENTS I ESTAT DE L'ART

6.1. Descripció del conflicte

Actualment, les centraletes dels vehicles estan connectades entre elles mitjançant un bus CAN, que els hi serveix per compartir informació entre elles a través de missatges. El bus consta físicament de dos cables que interconnecten totes les centraletes, el que permet estalviar cablejat i pes. Aquesta interconnexió dóna flexibilitat, ja que permet afegir o treure algunes de les centraletes segons les opcions escollides en el vehicle.

Per programar cadascuna d'aquestes centraletes es pot usar el bus del vehicle o bé programar cadascuna per separat. Degut al nombre creixent de centraletes que contenen els cotxes i les diferents opcions de programació que tenen cadascuna, és molt difícil gestionar un estoc tan ampli de centraletes preprogramades. És molt més senzill muntar les mateixes centraletes a tots els vehicles i canviar el programa que contenen segons les demandes del client a la mateixa cadena de muntatge.

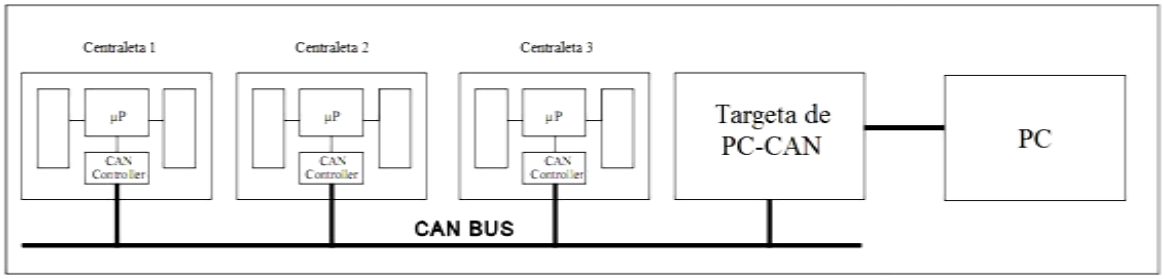

Figura 6.1. Esquema de la programació de centraletes a través del bus CAN

Per poder programar cada centraleta des del bus, es necessita que aquestes tinguin un programa preinstal·lat a fi de poder autoprogramar-se amb les dades rebudes a través del bus. Aquest programa és el *Bootloader* i resideix a les primeres posicions de memòria del microcontrolador, s'encarrega de llegir els missatges del bus, interpretar-los i finalment, posar cada dada a la posició de memòria demanada.

També és necessari utilitzar un PC per enviar les dades que cal programar. Els programes de les centraletes estan emmagatzemats al PC i es selecciona el més adient segons les opcions escollides a cada vehicle. Per transferir les dades de programació del PC al microcontrolador s'utilitza una targeta de PC que permet enviar i rebre missatges CAN. A més, cal un programa de PC que enviï les dades de programació de tal forma que el *Bootloader* de la centraleta les interpreti correctament. Aquest programa és específic pel *Bootloader* i la targeta de PC utilitzada.

Els passos per programar un centraleta a través del bus són:

1. Seleccionar el programa adequat segons les opcions del vehicle.
2. Enviar les dades del programa a través de missatges CAN de tal forma que puguin ser interpretats pel *Bootloader*.
3. El *Bootloader* interpreta els missatges i escriu les dades a la posició de memòria demanada.
4. Comprovar que les dades escrites al microprocessador són les mateixes que s'han enviat.

6.2. Avantatges i inconvenients

Resumidament, usar el bus del vehicle per programar les centraletes té els següents avantatges:

- Redueix el nombre de centraletes en estoc, ja que no cal emmagatzemar-les amb programes diferents.
- Permet programar les centraletes un cop muntades.
- Només és necessari un accés al bus per programar qualsevol centraleta.
- Es pot canviar el programa d'una centraleta un cop fabricat el cotxe.
- La programació és completament automatitzada.

I també té els següents inconvenients:

- La programació dura més temps, ja que totes les dades han de passar pel mateix bus.
- Tot i la fiabilitat de la transmissió de dades a través del bus CAN, cal comprovar que les dades programades a cada centraleta siguin correctes.

6.3. Tipus de connexions de PC a CAN

Es poden connectar les targetes de PC a CAN de diverses maneres. Una de les més comunes és a través d'un port USB. També n'hi ha d'internes per ordinadors portàtils (*Express Card* o PCMCIA) i sobretaula (PCI). Les més recents estan connectades sense fils, ja sigui amb estàndard *Wifi* o *Bluetooth*. Aquesta opció és interessant perquè evita realitzar connexions fixes amb el PC, i un sol ordinador a qualsevol lloc de la fàbrica pot programar diverses centraletes alhora.

6.4. Exemples de sistemes de programació via bus

Hi ha vàries empreses que es dediquen a muntar sistemes complets de programació a través de bus. Aquestes subministren el *Bootloader*, la targeta de PC i el software d'ordinador necessari per poder programar les ECUS(Electronic control units).

A destacar dues d'aquestes empreses:

1. *Vector.* Subministren els programes de PC *CANditoflash* per enviar les dades, *CANdelastudio* per definir els paràmetres de comunicació i el *CANdelaflash* per crear el programa de l' ECU. A més, subministra el hardware per comunicar el PC amb el bus del vehicle i el *Bootloader* a mida del client. El preu d'un sistema complet val uns 19000€ segons el pressupost demanat a l'empresa.

2. *Softing*. Subministra un sol programa per programar les centraletes a través del bus des del PC, el *DTS-flash*. També fabrica hardware per connectar el PC al Bus, un dels quals té interfície *Wifi*, l'EDICNET. Com Vector realitza projectes segons les especificacions del client, incloent el *Bootloader*.

7. ALTERNATIVES

Per poder muntar una xarxa de microcontroladors connectats per bus CAN i que siguin programables des del bus amb un PC, cal seleccionar el següent:

- Microprocessador amb regió *Boot* i *Bootloader* CAN.
- Targeta CAN per a PC.
- Llenguatge de programació pel programa de PC.
- Sistema operatiu.

7.1. Microprocessador amb *Bootloader* CAN

El microprocessador ha de complir les següents especificacions:

- Memòria de tipus flash. Per permetre l'autoprogramació del dispositiu.
- Amplada de bus de dades de 8bits, suficient per realitzar operacions complexes.
- Controlador CAN integrat d'un canal o possibilitat de connexió a un controlador CAN extern.
- Velocitat de rellotge (*Clock*) prou gran per poder fer servir taxes de transferència CAN altes (1Mbit/s).
- Disponibilitat d'una area de memòria dissenyada especialment per un programa de *Boot*, amb protecció contra l'escriptura per hardware. Així el programa d'usuari no pot escriure sobre el programa de *Boot*.
- Hardware i software de desenvolupament que permetin modificar el *Bootloader*.

Els Microcontroladors de 8bits que compleixen aquestes característiques són:

Microcontrolador	**Program Mem[Kbyte]**	**RAM [byte]**	**EEPROM [byte]**	***Clock* [MHz]**
***Microchip* PIC18FXX8**	16/32	768/1536	256	40
***Atmel* AT90CAN**	32/64/128	2048/4096	1024/2048/4096	16
***Freescale* MC9S08DZ**	17/34/48/60	1024/1536/2048	512/1024/1536/2048	40
***NEC* 78K0/FC2**	32/48/60	2048/3072	NO	20

Taula 7.1. Microcontroladors de 8 bits que compleixen les especificacions

Microchip i Atmel disposen de *Bootloader* CAN, en canvi NEC i Freescale no. Tot i que és possible crear un *Bootloader* CAN per aquests dos últims, això requeriria més temps per realitzar el projecte.

7.2. Targeta de PC

Pel que fa a la targeta de PC s'ha optat per un dispositiu USB, ja que permet una connexió senzilla a un ordinador portàtil o de sobretaula.

- Disposar d' 1 canal CAN.
- Complir la norma ISO11898-2.
- Disposar de llibreria i *driver* per PC que permeti accedir a les funcions de la targeta des del sistema operatiu.
- Eines de desenvolupament per poder monitoritzar el bus.

A la taula 7.2 es mostren les targetes que compleixen les especificacions.

Targeta de PC	**Llibreries**	**S.O. compatibles**	**Eines monotorització**
Kvaser Leaf Light HS	CANlib32	Windows/Linux	*CanKing*
Softing CANUsb	CAN-API	Windows	*X-Analyser Appetizer*
RM canviewUSB	No gratuït	Mac/Linux/Windows	*RM CAN Device Monitor*
Sonthiem CANUSBlight	CANapi	Windows	*CANexplorer*
Ixxat USB to CAN	No gratuït	Windows	*miniMon*

Taula 7.2. Targetes PC-CAN que compleixen les especificacions

Tots ells disposen d'un sol canal CAN i software de desenvolupament pels sistemes operatius citats a dalt.

7.3. Llenguatge de programació

Actualment, es poden crear programes en entorn gràfic usant diferents tipus de llenguatges. Els més comuns són els orientats a esdeveniments i els orientats a objectes. Els orientats a esdeveniments són més fàcils d'aprendre, però cal controlar el flux del programa perquè aquest funcioni correctament. En canvi, els orientats a objectes són més difícils d'aprendre però més senzills per controlar del flux del programa.

Per cada llenguatge existeixen eines per desenvolupar programes amb més facilitat, s'anomenen els Entorns de Desenvolupament Integrats (IDE). Els més usats són els següents:

IDE per llenguatges orientats a objectes:

- *Microsoft Visual Studio* per llenguatge *VB.NET*, només vàlid per Windows.
- *NetBeans* per llenguatge *Java*, vàlid per qualsevol sistema operatiu.
- *Delphi* per llenguatge *Pascal*, vàlid per qualsevol sistema operatiu.

IDE per llenguatges orientats a esdeveniments:

- *Microsoft Visual Basic 6.0* per llenguatge *Basic*, només per a Windows.
- *Gambas* per llenguatge *Basic*, només per a LINUX.

7.4. Sistema operatiu

Actualment, en el món dels sistemes operatius per a PC hi ha 2 tendències clares. Una és el LINUX i l'altre Windows. LINUX és software lliure i permet modificar totes les seves parts sense pagar cap llicència. Windows en canvi és software propietari, i només permet la seva utilització com a usuari.

Cal destacar que el sistema LINUX pot funcionar en diferents tipus de màquines i és gratuït. Tot i això, el sistema Windows continua essent el més utilitzat degut a la seva facilitat d'ús.

8. SOLUCIÓ ADOPTADA

Finalment, s'ha decidit usar el següent hardware i software:

- Microcontrolador: *Microchip* PIC18F458
- Targeta de PC a CAN: *Kvaser Leaf light HS*
- Entorn de desenvolupament integrat (IDE): *Microsoft Visual Basic 6.0*
- Sistema operatiu: Windows 98/Me/NT/2000/XP/Vista

8.1. *Microchip* PIC18F458

S'ha escollit aquest microcontrolador perquè el Departament d'Enginyeria Electrònica disposava de plaques amb aquest microcontrolador, utilitzades en les pràctiques de l' ALE d'Electrònica de l'Automòbil. Aquesta placa disposa de connexions de bus CAN i de programador ICSP *(In Circuit Serial Programming*) per poder modificar el *Bootloader*. Les característiques de la placa i els seus components es detallen al l'annex D.

8.2. *Kvaser Leaf light HS*

S'ha escollit aquesta targeta de PC perquè el Departament d'Enginyeria Electrònica en disposava d'una unitat. És econòmica i disposa de les eines de desenvolupament necessàries per realitzar un programa en entorn gràfic. Les característiques d'aquesta targeta de PC i la seva llibreria de funcions es detallen a l'annex E.

8.3. *Microsoft Visual Basic 6.0*

Tot i que és un entorn de programació antic, data de l'any 1998, s'ha escollit per la seva facilitat d'ús i perquè permet obtenir bons resultats en poc temps. També perquè és compatible amb els sistemes operatius Windows antics i actuals. A més, el Departament d'Enginyeria Electrònica disposava d'una llicència pel seu ús. Les característiques d'aquest entorn de programació es detallen a l'annex F.

8.4. Sistema operatiu de PC

Com que s'ha escollit un entorn de programació que funciona únicament sobre Windows, els programes només seran compatibles amb aquest sistema operatiu.

8.5. Sistema complet

Aquest és l'esquema del sistema complet, amb tot el hardware i software escollit. Les dades flueixen del programa de Visual Basic al programa *Bootloader* i a l'inversa, ja que pot haver-hi comunicació bidireccional entre els 2.

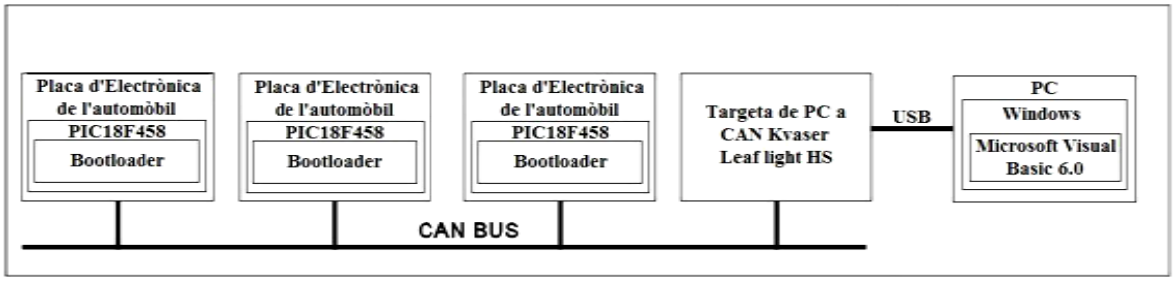

Figura 8.1. Esquema del sistema complet

9. El *BOOTLOADER* MODIFICAT

El *Bootloader* pot funcionar en dos modes, el mode *PUT* o el *PUTGET*. En el mode *PUT* la comunicació és unidireccional, el microcontrolador no envia missatges al Bus. En canvi, en el mode *PUTGET* el microcontrolador envia missatges, permetent operacions de lectura d'informació i sincronització durant la programació. Per referències d'aquest capítol consultar [1] . Per entendre millor el seu funcionament consultar els apartats 9.6 i 9.7 on es mostren els diagrames de flux complets del *Bootloader* en mode *PUT* i en mode *PUTGET*. També es pot consultar l'annex G per consultar el codi font del *Bootloader*.

9.1. Missatges enviats per la font de programació

El tipus de missatge enviat per la font de programació es determina mitjançant dos bits de l'identificador del missatge CAN, el bit *Control-data* de recepció i el bit *Put-get* de recepció. Aquests bits es poden escollir quan es compila el *Bootloader*. Per més informació sobre els missatges CAN consultar l'annex C.

Hi ha quatre tipus de missatges que es poden enviar al *Bootloader* per accedir a les seves funcions.

Bit *Put-get* de recepció	**Bit *Control-data* de recepció**	**Tipus de missatge**	**Vàlid per mode**
0	**0**	Put control	*PUT* I *PUTGET*
0	**1**	Put data	*PUT* I *PUTGET*
1	**0**	Get control	*PUTGET*
1	**1**	Get data	*PUTGET*

Taula 9.1. Tipus de missatges CAN que s'envien al *Bootloader*

A part de l'identificador, les dades enviades en el mateix missatge poden configurar algunes funcions a realitzar. Segons el tipus de missatge les dades tenen un significat diferent. Cada missatge CAN disposa de com a màxim 8 bytes d'informació. El significat de les dades de la taula 9.2 s'expliquen en l'apartat 9.3.

Tipus missatge	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Put control	ADDRL	ADDRH	ADDRU	RESVD	CTLBT	SPCMD	CPDTL	CDPTH
Put data	DATA0	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7
Get control	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Get data	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Taula 9.2. Dades dels missatges enviats al *Bootloader*

9.2. Missatges enviats pel *Bootloader* a la font de programació.

El tipus de missatge enviat pel *Bootloader* es determina mitjançant 1 bit de l'identificador del missatge CAN, el bit *Control-data* de transmissió. Aquest bit es pot escollir quan es compila el *Bootloader*.

El *Bootloader* pot enviar missatges amb informació de control o de dades. També pot enviar missatges d'*Acknowledge* (ACK), que són respostes als missatges *Put* i no tenen dades. Els ACK serveixen per sincronitzar la font de programació i el *Bootloader*.

Bit *Control-data* de transmissió	Tipus de missatge	Vàlid per mode
0	INFORMACIÓ DE CONTROL	*PUTGET*
1	INFORMACIÓ DE DADES	*PUTGET*
0	*ACKNOWLEDGE* DE CONTROL	*PUTGET*
1	*ACKNOWLEDGE* DE DADES	*PUTGET*

Taula 9.3. Tipus de missatges CAN que envia el *Bootloader*

La resposta del *Bootloader* està determinada pel missatge previ enviat per la font de programació. Les dades enviades segons el missatge rebut es citen a la taula 9.4. En l'apartat 9.3. s'expliquen amb detall el seu significat.

Tipus rebut	Tipus enviat	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Get control	Info Control	ADDRL	ADDRH	ADDRU	RESVD	STATS	RESVD	RESVD	RESVD
Get data	Info Data	DATA0	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7
Put control	ACK Control	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Put data	ACK Data	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Taula 9.4. Dades enviades pel *Bootloader*

9.3. Significat de les dades dels missatges CAN

La taula 9.5 és un resum del significat de les dades aparegudes en els apartats 9.1 i 9.2. En els apartats següents es farà una descripció més estesa de cadascuna d'elles.

Abreviació	Significat
ADDRL	És el byte menys significatiu de l'adreça de l'apuntador.
ADDRH	És el byte d' entremig de l'adreça de l'apuntador.
ADDRU	És el byte més significatiu de l'adreça de l'apuntador.
RESVD	Byte sense usar, reservat per a futurs usos.
CTLBT	Bits de control, serveixen per configurar les funcions del *Bootloader*.
SPCMD	*Comandaments especials*, serveixen per demanar que s'executi una comanda.
CPDTL	És el byte menys significatiu de les dades dels Comandaments especials
CPDTH	És el byte més significatiu de les dades dels *Comandaments especials*
DATA 0-7	Són bytes de dades, ja siguin llegides o enviades per ser escrites.
STATS	Aquest byte ens indica l'estat dels *Control bits*.
N/A	No aplica, són buits.

Taula 9.5. Significat de les dades

9.3.1. Adreces (ADDRL, ADDRH i ADDRU)

Com que tota la memòria del *PIC18FXX8* està mapajada, es pot escriure a qualsevol tipus de memòria mitjançant apuntadors. Tot i això, alguns bits de configuració només es poden canviar amb un programador extern. A la taula 9.6 hi ha la correspondència entre l'adreça en hexadecimal i la memòria accedida.

Adreça d'inici	Adreça final	Memòria accedida	Operacions
000000h	0001FFh	*Boot program*	Llegir
000200h	007FFFh	*Program memory*	Llegir i escriure
200000h	200008h	*User Identification*	Llegir i escriure
300000h	3FFFFFh	*Configuration bits*	Depèn de la funció
F00000h	F000FFh	EEPROM	Llegir i escriure

Taula 9.6. Mapa de les memòries

Com es pot observar les adreces estan formades per 3 bytes, sis xifres hexadecimals. Per tant, per enviar una adreça cal usar els tres bytes: ADDRL, ADDRH i ADDRU. Per exemple, l'adreça 200008h s'enviaria posant ADDRL=08h, ADDRH=00h i ADDRU=20.

9.3.2. Bits de Control

Els Bits de Control ens permeten controlar el funcionament del *Bootloader*. Hi ha 5 funcions que es poden habilitar o deshabilitar canviant el valor dels primers bits del byte CTLBT. A la taula 9.7 es descriuen les funcions d'aquests 5 bits.

Bit CTLBT	Funció	Comportament si val 1	Comportament si val 0
0	**WRITE UNLOCK**	Permet escriure a qualsevol memòria.	No deixa escriure a qualsevol memòria.
1	**ERASE ONLY**	No escriu noves dades després d'esborrar.	Escriu dades noves després d'esborrar.
2	**AUTO ERASE**	Esborra 64 bytes consecutius abans d'esborrar.	No esborra abans d'escriure les dades.
3	**AUTO INC**	Incrementa el punter automàticament.	No incrementa el punter, s'ha de fer manualment.
4	**ACK**	Envia missatges d'*Acknowledge*, només si està en mode *PUTGET*.	No envia missatges d'*Acknowledge*.

Taula 9.7. Bits de Control

9.3.3. *Comandaments especials*

Els *Comandaments especials* són unes comandes usades per posar a 0 i comprovar el *Checksum* i per fer un *Reset* al microcontrolador.

SPCMD	Comanda	Descripció
00h	***NOP***	No realitza cap operació.
01h	***RESET***	Fa un *Reset* de software a l'aparell.
02h	***INIT CHECKSUM***	Inicialitza el *Checksum* i esborra el *Flag* d'error d'escriptura.
03h	***CHECK AND RUN***	Comprova el *Checksum* i el *Flag* d'error d'escriptura. Si és correcte fa un *Reset* software.

Taula 9.8. Comandaments especials

La comanda *Check and run* en la versió original de *Microchip*, enlloc de fer un *Reset* escrivia FFh a l'ultima posició de la EEPROM.

9.3.4. Dades dels *Comandaments especials* (CPDTL i CPDTH)

Aquests dos bytes s'utilitzen per enviar els valors del *Checksum* calculats a la font de programació per poder-los comparar amb els calculats pel *Bootloader*. S'envien juntament amb una comanda *Check and Run* perquè es pugui realitzar la comanda correctament.

9.3.5. Dades (DATA 0-7)

Si es tracta d'un missatge Put data, aquestes dades s'escriuran a la posició de memòria consecutives a l'apuntador. Si es un missatge d'informació de dades, són les dades escrites a continuació de l'apuntador.

9.4. Funcions del *Bootloader*

Les funcions que realitza el *Bootloader* segons els missatges rebuts es mostren en la taula 9.9.

Tipus de missatge	Funcions
Put control	Posiciona el punter a l'adreça. Actualitza els *Bits de Control*i executa la comanda, si és necessari fa servir les dades dels *Comandaments especials*. Si està en mode *PUTGET* i ACK està activat, envia un missatge d'*Acknowledge* de control quan acaba l'operació.
Put data	Escriu les dades a on està situat el punter. Depenent del tipus de memòria escriu 1 o 8 bytes. Si està en mode *PUTGET* i ACK està activat, envia un missatge d'*Acknowledge* de dades quan acaba l'operació.
Get control	Llegeix el punter i l'estat dels *Configuration bits* i envia un missatge d'informació de control.
Get data	Llegeix els bytes escrits on hi ha el punter. Segons el tipus de memòria llegeix 1 o 8 bytes. Després envia un missatge d'informació de control.

Taula 9.9. Funcions realitzades pel *Bootloader*

9.5. Inici del *Bootloader*

El *Bootloader* subministrat per *Microchip* s'ha modificat perquè es pugui activar amb un missatge CAN, just després d'iniciar el microcontrolador. Aquest missatge d'inici ha de ser del tipus Put-control. Per entrar en mode *Boot* s'ha de rebre dins d'un període de temps determinat (*Timeout*), si no salta al programa d' usuari. Aquest temps és controlat pel TIMER 0 del microcontrolador. Si el *Bootloader* està en mode *PUTGET* al rebre correctament el missatge d'inici envia un *Acknowledge* de control, abans d'entrar en mode *Boot*.

Per entendre més fàcilment aquestes modificacions, en els apartats 9.6 i 9.7 es mostren els diagrames de flux complets del *Bootloader* en mode *PUT* i en mode *PUTGET*. També es pot consultar l'annex G per consultar el codi font del *Bootloader*.

9.6. Diagrama de flux del *Bootloader* en mode *PUT*

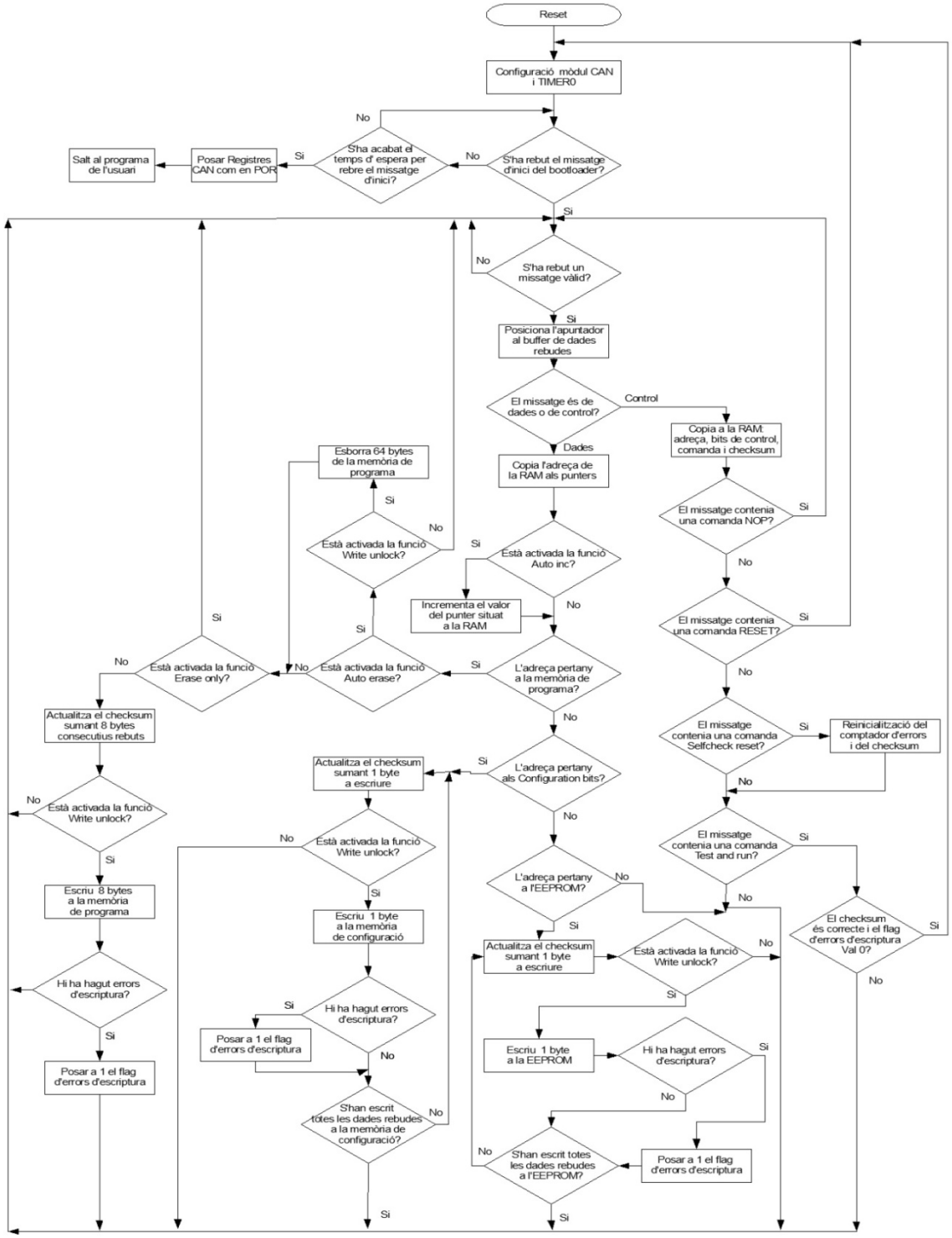

Figura 9.1. Diagrama de flux del *Bootloader* en mode *PUT*

9.7. Diagrama de flux del *Bootloader* en mode *PUTGET*

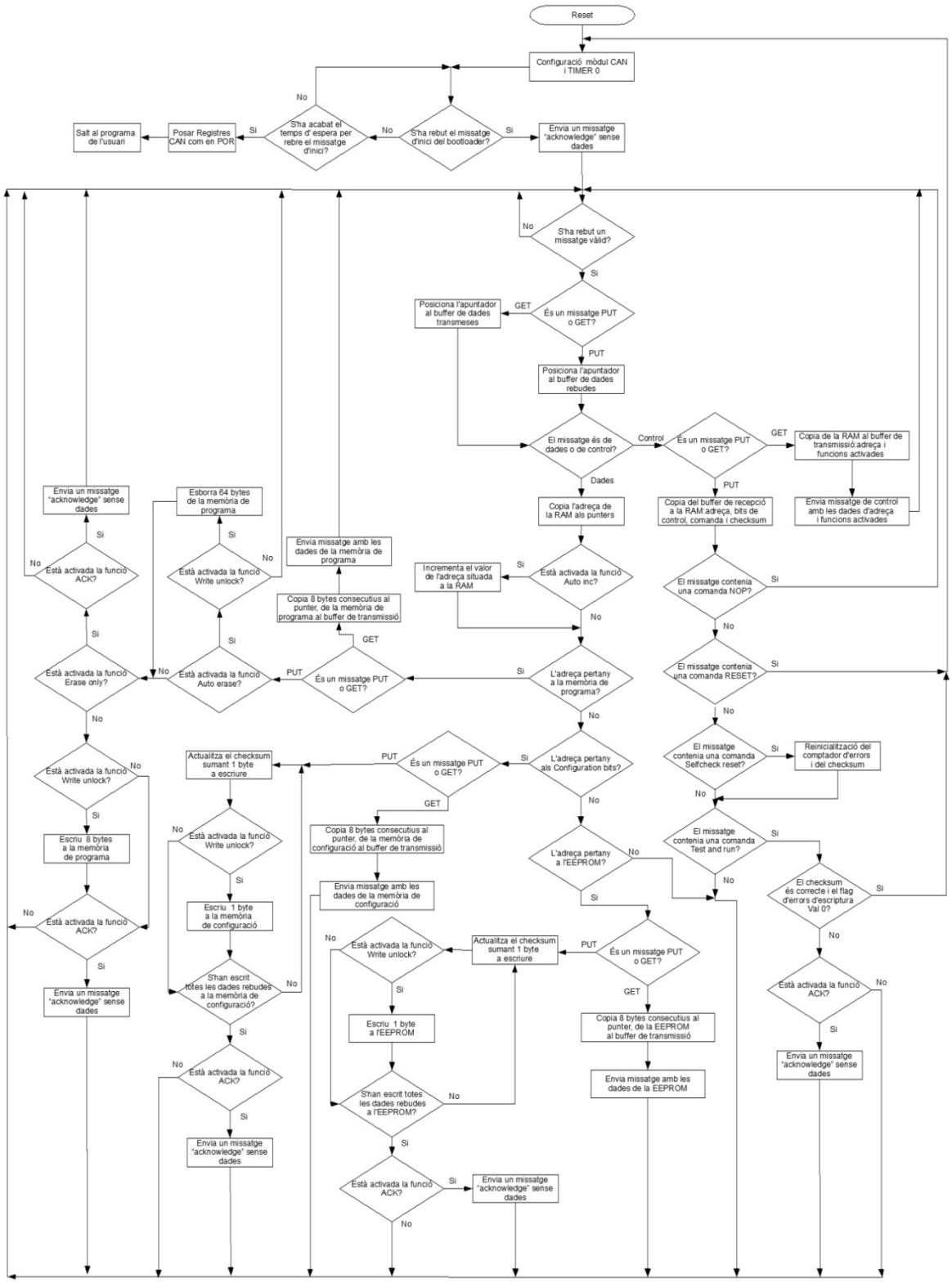

Figura 9.2. Diagrama de flux del *Bootloader* en mode *PUTGET*

9.8. Opcions de compilació del *Bootloader*

El *Bootloader* és un programa escrit en llenguatge assemblador, per modificar fàcilment alguns paràmetres i no canviar el codi s'usa un fitxer de definicions. Els paràmetres del *Bootloader* que es poden modificar amb el fitxer són explicats a la taula 9.10.

Paràmetre	**Descripció**
Boot mode	Permet escollir entre el mode *PUT* o el *PUTGET*.
***Reset vector* remapejat**	És l'adreça de memòria de programa on comença el codi de l'usuari.
Vector d' interrupció d'alta prioritat remapejat	És l'adreça de memòria de programa on comença el codi de les interrupcions d'alta prioritat definit per l'usuari.
Vector d' interrupció de baixa prioritat remapejat	És l'adreça de memòria de programa on comença el codi de les interrupcions de baixa prioritat definit per l'usuari.
Bit *Control-data* de recepció (C/D Rx)	És el bit de l'identificador del missatge CAN rebut pel *Bootloader* que determina si és de dades o de control.
Bit *Put-get* de recepció (P/G Rx)	És el bit de l'identificador del missatge CAN rebut pel *Bootloader* que determina si un missatge és Put o Get.
Bit *Control-data* de transmissió (C/D Tx)	És el bit de l'identificador del missatge CAN enviat pel *Bootloader* que determina si és de dades o de control.
Tipus de missatges CAN	Els missatges CAN poden ser de tipus estàndard o estès. Per més informació consultar l'annex C.
Identificador CAN de transmissió	És l'identificador que usa el *Bootloader* per enviar missatges a la font de programació.
Filtre CAN de recepció	És l'identificador CAN dels missatges que envia el PC.
Màscara CAN de recepció	Permet rebre missatges diferents del filtre CAN, és el que permet rebre missatges amb diferents identificadors.
Paràmetres de configuració del CAN	Són els paràmetres de configuració del mòdul CAN del microcontrolador.
Timeout* de l'inici del *Bootloader	És el temps que espera el *Bootloader* per rebre el missatge d'inici de programació. Si durant aquest temps no rep el missatge salta al programa de l'usuari.
Filtre CAN de programació	És un filtre CAN alternatiu poder canviar l'inici del *Bootloader* en una futura versió.

Taula 9.10.Paràmetres de configuració del *Bootloader*

10. CONFIGURACIÓ DEL *BOOTLOADER*

El programa de configuració del *Bootloader* crea dos fitxers, un serveix per compilar el *Bootloader (CANBOOT.def)* i l'altre per definir els paràmetres de comunicació amb el PC *(BOOTDATA.txt).* Per veure el seu codi font consultar l'annex H. El primer, és l'arxiu de definicions comentat en el capítol del *Bootloader*, i el segon, s'explica en l'apartat 10.1.

10.1. Fitxer *BOOTDATA*

El fitxer *BOOTDATA* emmagatzema la informació necessària per establir la comunicació entre el PC i el *Bootloader* a través del bus CAN. La informació emmagatzemada pel fitxer és:

Paràmetre	**Descripció**
Boot mode	Permet escollir entre el mode *PUT* o el *PUTGET*.
Reset vector	És l'adreça de memòria de programa on comença el codi de l'usuari.
Bit *Control-data* de recepció (C/D Rx)	És el bit de l'identificador del missatge CAN rebut pel *Bootloader* que determina si és de dades o de control.
Bit *Put-get* de recepció (P/G Rx)	És el bit de l'identificador del missatge CAN rebut pel *Bootloader* que determina si un missatge és Put o Get.
Bit *Control-data* de transmissió (C/D Tx)	És el bit de l'identificador del missatge CAN enviat pel *Bootloader* que determina si és de dades o de control.
Tipus de missatges	Els missatges CAN poden ser de tipus estàndard o estès. Per més informació consultar l'annex C.
Identificador CAN de transmissió	És l'identificador que usa el *Bootloader* per enviar missatges al PC.
Filtre CAN de recepció	És l'identificador CAN dels missatges del PC.
Màscara CAN de recepció	Permet rebre missatges diferents del filtre CAN, és el que permet rebre missatges amb diferents identificadors.
Velocitat del bus (kbit/s)	És la velocitat de transferència d' informació del bus CAN.
Timeout* de l' inici del *Bootloader	És el temps que espera el *Bootloader* per rebre el missatge d'inici del *Bootloader*. Si durant aquest temps no rep el missatge salta al programa de l'usuari.
Filtre CAN de programació	És un filtre CAN alternatiu per poder canviar l'inici del *Bootloader* en una futura versió.

Taula 10. 1.Informació continguda en el fitxer *BOOTDATA*.txt

10.2. Formulari principal del programa de definicions

La finestra principal del programa es mostra en la figura 10.1. S'han marcat amb colors els controls segons la seva funció.

- *Vermell:* Configuració dels paràmetres del CAN.
- *Groc:* Tria del mode de funcionament del *Bootloader*.
- *Taronja:* Configuració dels vectors de *Reset* i interrupció.
- *Verd:* Configuració de la comunicació entre el PC i el *Bootloader*.
- *Negre:* Configuració de l'arrencada del *Bootloader*.
- *Blau:* Botons per executar diferents comandes.

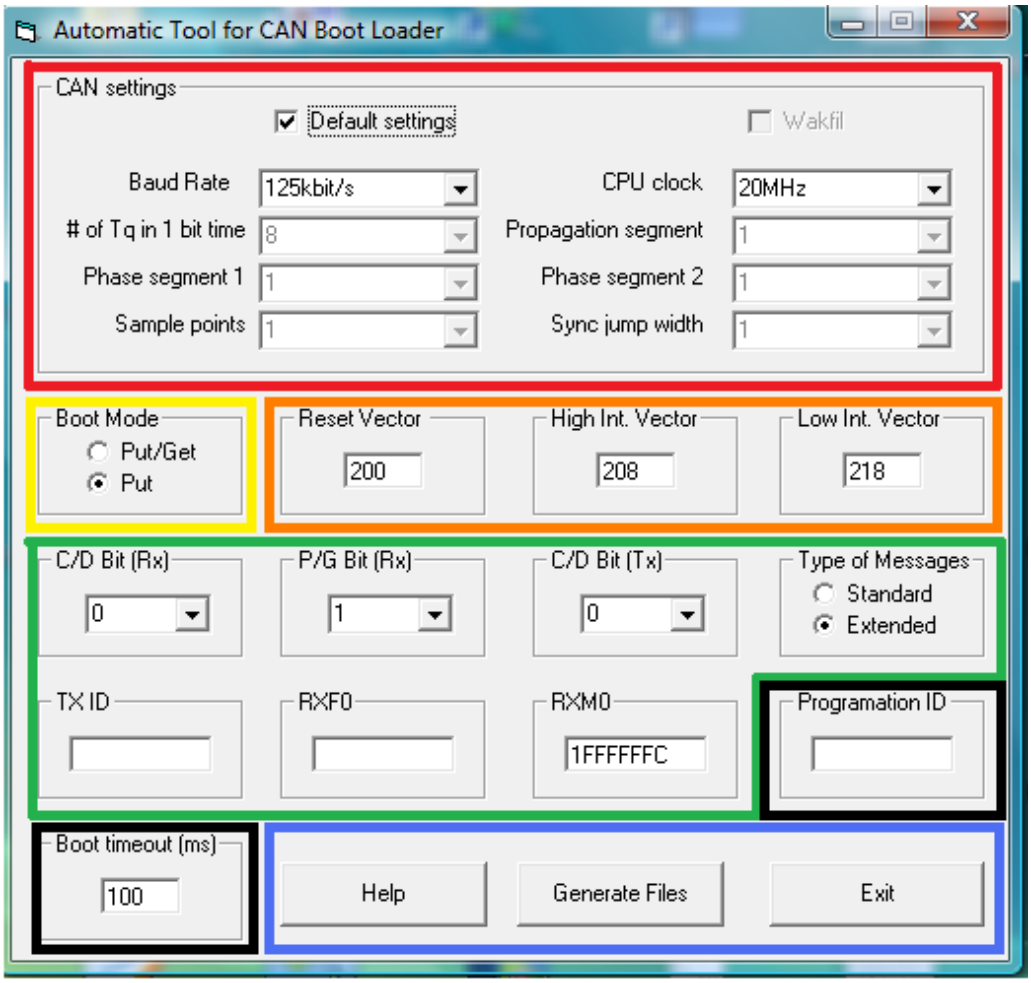

Figura 10.1. Formulari principal del programa de configuració del *Bootloader*

10.3. Configuració dels Paràmetres del CAN

Les opcions dels paràmetres CAN s'han limitat a aquelles configuracions on la velocitat del bus *(Baud Rate)* és exacte. Es pot escollir entre 4 velocitats de CPU i 4 de Bus. Les velocitats de CAN disponibles són: 125, 250, 500 i 1000kbit/s, que són les més usades, en diferents CAN *Higher Layer protocols* (consultar annex C). Pel que fa les velocitats de processador s'ha optat per: 5, 10, 20 i 40MHz, ja que són els més habituals per aquest tipus de microprocessador [2]. És important la freqüència en que treballa el processador ja que la freqüència del mòdul CAN depèn d'aquesta.

L'enviament d'un bit a través del bus requereix un temps, aquest temps ve determinat per la velocitat del Bus. El temps de que es disposa és el Nominal Bit time(t_{NBT}), que ve donat per l'equació 10.1[3].

$$t_{NBT} = \frac{1}{f_{NBT}} \qquad \text{(Eq. 10.1)}$$

On f_{NBT} és el Baud rate. Aquest temps es divideix en 4 parts: *Synchronisation segment*, *Propagation segment , Phase segment 1 i Phase segment 2*. El temps que dura cada part es mesura amb una unitat de temps anomenada T_Q *(Time Quantum).* Un T_Q és el període d'una senyal de rellotge del mòdul CAN.

- *Synchronisation segment*: És la part del temps de bit que serveix per sincronitzar els diferents nodes del bus.
- *Propagation segment*: És la part del temps de bit que s'usa per compensar el retard de la xarxa. Inclou el retard degut al cable i l'intern del node.
- *Phase segment 1 i 2*: S'usen per localitzar amb precisió els *Sample Points.*

El controlador llegeix el valor del bit en els *Sample points*. Pot haver-n'hi un o tres, si n'hi ha un està situat justament quan acaba el *Phase segment 1*. Si n'hi ha tres, el primer està 1 T_Q abans de la finalització del *Phase segment 1,* el segon està situat mig T_Q abans de la fi del *Phase segment 1* i el tercer quan acaba justament el *Phase segment 1*. Si es fan servir 3 *Sample Points*, el controlador pren com a valor de bit el que ha aparegut més vegades.

El nombre de T_Q's que duren les parts és variable, menys pel *Synchronisation segment,* que sempre és d'un T_Q. El *Propagation segment* i el *Phase segment 1* poden valdre entre 1 i 8T_Q. Pel que fa el valor del *Phase segment 2* ha d'estar entre 2 i 8T_Q's, ja que 2 T_Q's és el temps mínim necessari perquè el controlador CAN processi l' informació [2]. El nombre màxim de T_Q's per bit és 25.

A la figura 10.2 s'observa que la relació entre la freqüència del processador i la freqüència del controlador CAN és el *Baud Rate Prescaler* . També s'observa les parts en què està dividit el temps d'un bit.

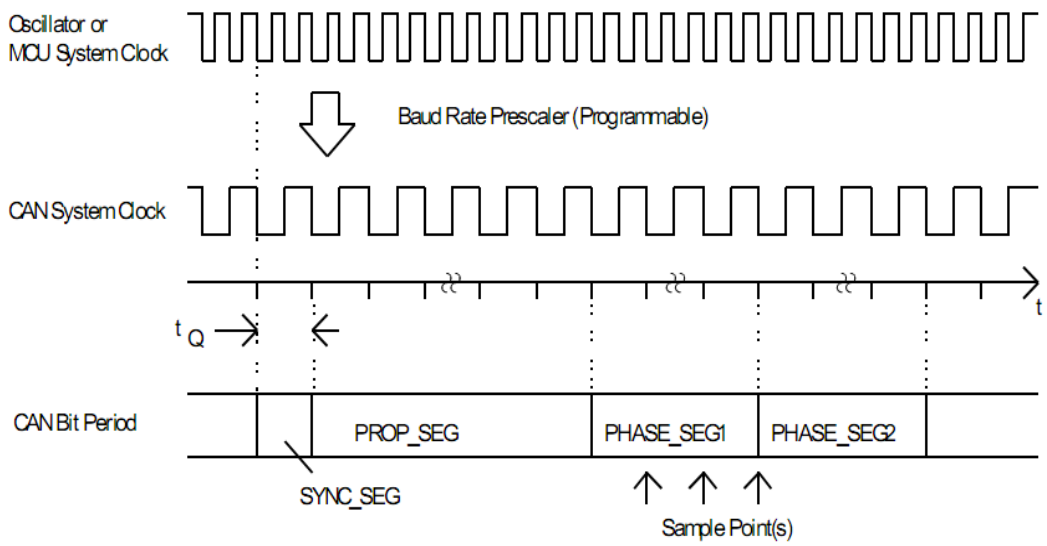

Figura 10. 2. Transmissió d'un bit CAN [3]

Per resincronitzar el sistema degut a variacions dels rellotges dels diferents nodes, s'usa el *Synchronisation jump width* (SJW). Aquest permet escurçar o allargar el *Phase segment 1* allargant o escurçant el *Phase segment 2*, la variació entre els 2 P*hase segments* sempre és del valor del SJW.

10.3.1. Registres CAN del microcontrolador

Per configurar correctament el mòdul CAN del microcontrolador *PIC18FXX8* s'han de configurar 3 registres: BRGCON1, *BRGCON2* i *BRGCON3* [2]. A la taula 10.2 es mostren quins paràmetres configuren cadascun d'aquests registres.

Registre	Paràmetres que configura
BRGCON1	*Synchronized Jump Width time i Baud Rate Prescaler bits.*
BRGCON2	Nombre de *Sample points, Phase segment 1 time i Propagation time*. L'opció de programació lliure del *Phase segment 2.*
BRGCON3	El *Phase segment 2* time i *l'opció Wakfil.*

Taula 10.2. Registres del mòdul CAN del microcontrolador [2]

10.3.1.1. *BRGCON1*

- *Baud Rate prescaler (BRP)*: Per calcular el valor del BRP en funció de la freqüència del microcontrolador (F_{OSC}), la velocitat de bus desitjada (BR) i el nombre de T_Q's (#T_Q) per bit time, s'usa l'equació 10.2. El BRP és adimensional i ha de ser un enter entre 0 i 63 [2].

$$BRP = \frac{F_{osc}}{BR \cdot \#\mathrm{T_Q} \cdot 2} - 1 \qquad \text{(Eq. 10. 2.)}$$

- *Syncrhonisation jump width*: S'usa l' introduït en la casella a tal efecte. Es pot escollir valors entre 1 i 4 T_Q's.

10.3.1.2. *BRGCON2*

Per configurar aquest registre s'usen els valors introduïts en cadascuna de les caselles. L'opció de programació lliure del *Phase segment 2* es deixa sempre habilitada per poder entrar el valor en el registre *BRGCON3*.

10.3.1.3. *BRGCON3*

Per configurar aquest registre s'usa el valor posat a la casella del *Phase segment 2* i el la casella de verificació del *Wakfil*. El *Wakfi*l és una opció que permet sortir del mode S*leep* del microcontrolador si es rep un missatge CAN [2].

10.3.2. Combinacions sense error

Al limitar el nombre de velocitats i Baud rates s'obté la taula 10.3 de combinacions sense error segons el nombre de T_Q's per bit [9]. Les combinacions que donen un BRP no vàlid són les marcades amb "NO", les marcades amb impossible tenen un senyal de rellotge massa baix per la velocitat de bus.

	125 [kbit/s]				250 [kbit/s]				500 [kbit/s]				1000 [kbit/s]			
# T_Q'S	20	16	10	8	20	16	10	8	20	16	10	8	20	16	10	8
5 [MHz]	Ok	No	Ok	No	No	No	Ok	No	Impossible				Impossible			
10 [MHz]	Ok	No	Ok	Ok	Ok	No	Ok	No	No	No	Ok	No	Impossible			
20 [MHz]	Ok	Ok	Ok	Ok	Ok	No	Ok	Ok	Ok	No	Ok	No	No	No	Ok	No
40 [MHz]	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	No	Ok	Ok	Ok	No	Ok	No

Taula 10.3. Paràmetres possibles de configuració

10.3.3. Paràmetres CAN per defecte

S'han escollit uns paràmetres CAN per defecte corresponents a l'opció de 10 T_Q's per bit, ja que és vàlid per totes les configuracions possibles i no té error. Per cada opció de velocitat de bus i de rellotge s'ha canviat únicament el valor del *Baud Rate Prescaler*.

Nombre de T_Q's per bit [T_Q]	Propagation segment [T_Q]	Phase segment 1 [T_Q]	Phase segment 2 [T_Q]	SJW	Sample points	WAKFIL
10	1	5	3	1	1	NO

Taula 10. 4.Paràmetres CAN per defecte

10.4. Configuració del mode de funcionament del *Bootloader*

Aquest control permet escollir entre els 2 modes de funcionament del *Bootloader*, *PUT* o *PUTGET*.

10.5. Configuració de la comunicació entre *Bootloader* i PC.

El controlador CAN del PIC18 disposa de 2 *buffers* de recepció (RXB0 i RXB1) i 3 de transmissió (TXB0,TXB1 i TXB2) [2]. Els *buffers* de recepció acullen les dades dels missatges rebuts si el missatge ha estat acceptat. Per seleccionar els missatges a acceptar segons el seu identificador, els *buffers* de recepció disposen de filtres i màscares. Pel que fa al *buffer* de transmissió cal assignar-li un identificador pels missatges que s'enviïn al PC. S'ha escollit usar el *buffer* de recepció RXB0 i el *buffer* de transmissió TXB0. La figura 10.3 mostra l'arquitectura del mòdul CAN de microcontrolador.

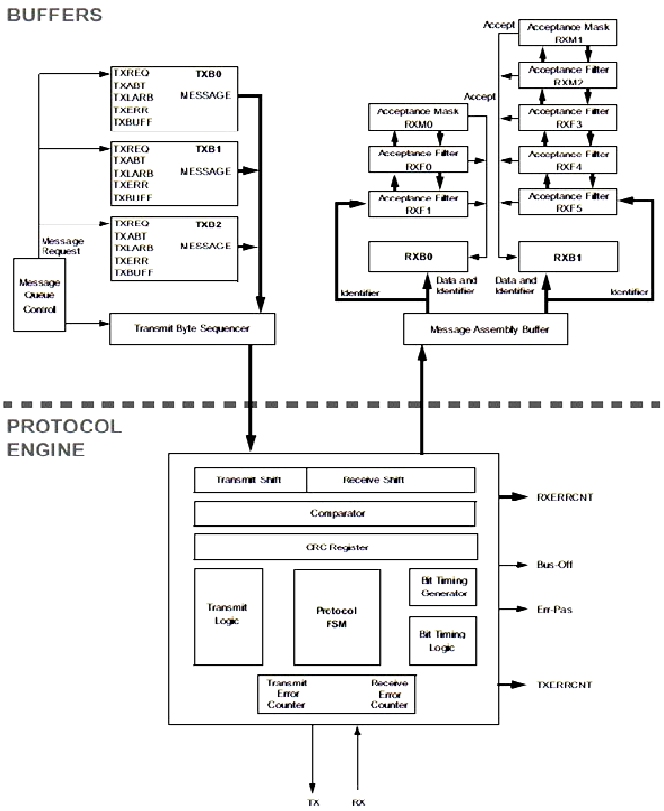

Figura 10.3. Esquema del mòdul CAN del *PIC18FXX8* [2]

Per seleccionar els missatges que seran emmagatzemats al *buffer* RXB0, s'usa el filtre RXF0, i la màscara RXM0. Per poder enviar missatges amb TXB0 cal configurar l'identificador de transmissió.

10.5.1. Selecció del tipus de missatge

Segons les especificacions de la norma ISO11898-2 es poden usar dos tipus de missatges CAN, estàndard o estès [10]. Els 2 sistemes es poden usar segons les necessitats de l'aplicació si el controlador CAN compleix aquesta norma. La diferència principal entre els dos sistemes és el nombre de bits de l'identificador.

10.5.2. Selecció dels registres dels bits *Put-get* i *Control-data*

Els registres són diferents segons l'índex del bit i del tipus de missatges usat. El programa selecciona automàticament el registre que ocupen els bits de *Put-get* (P/G Rx), en el programa) i *Control-data* (C/D Rx) en el filtre de recepció. També ho fa pel bit *Control-data*

(C/D TX) de l'identificador de transmissió. Els dos bits del filtre de recepció han de tenir índexs diferents, si no és així apareix un missatge d'error.

Si els missatges són de tipus estàndard, l'identificador de missatge disposa d' 11 bits [10]. Segons l'índex del bit escollit, aquest pertanyerà a un registre o altre. La taula 10.5 mostra els registres implicats[2].

	Índex del bit entre 10 i 3	**Índex del bit entre 2 i 0**
Filtre de recepció	RXF0SIDH	RXF0SIDL
Identificador de transmissió	TXB0SIDH	TXB0SIDL

Taula 10. 5.Registres d'identificadors de missatges estàndard

Si s'usen missatges estesos, l'identificador disposa de 29 bits [10]. Segons l'índex del bit escollit, pertanyerà a un dels 4 registres. La taula 10.6 mostra els registres implicats[2].

	Índex del bit entre 29 i 21	**Índex del bit entre 20 i 16**	**Índex del bit entre 15 i 8**	**Índex del bit entre 8 i 0**
Filtre de recepció	RXF0SIDH	RXF0SIDL	RXF0EIDH	RXF0EIDL
Identificador de transmissió	TXB0SIDH	TXB0SIDL	TXB0EIDH	TXB0EIDL

Taula 10. 6. Registres d'identificadors de missatges estesos

10.5.3. Selecció del filtre de recepció i de l'identificador de transmissió

Per seleccionar el filtre i l'identificador de transmissió cal posar un identificador CAN vàlid a cada casella. Si es seleccionen missatges estàndard s'han d'introduir 3 xifres hexadecimals, si són estesos 8 xifres per identificador. El programa comprova que hi son totes i que estan dins del rang possible. Els dos identificadors han de ser diferents pel correcte funcionament del *Bootloader* [1].

10.5.4. Selecció de la màscara de recepció

La màscara permet rebre missatges amb identificador diferent al del filtre de recepció, emmascarant els bits no coincidents. Per tant, és indispensable configurar correctament la màscara pel correcte funcionament del *Bootloader*, ja que ha de poder rebre missatges amb identificadors diferents per realitzar les diferents funcions [1]. Els missatges rebuts són acceptats si compleixen l'equació 10.3 [7].

$$(Filtre\ XOR\ Identificador\ missatge\ rebut)\ AND\ Màscara = 0 \qquad \text{(Eq.10. 3)}$$

Segons els bits seleccionats el programa calcula una màscara de recepció automàticament, encara que es pot modificar a voluntat un cop escollits els bits. La màscara es calcula de manera diferent segons el tipus de missatge.

Si el missatge és estàndard es calcula amb l'equació 10.4.

$$Màscara = 7FFh - 2^{index\ PG\ RX} - 2^{index\ CD\ RX} \qquad \text{(Eq.10. 4)}$$

I si és de estès amb l'equació 10.5.

$$Màscara = 1FFFFFFFh - 2^{index\ PG\ RX} - 2^{index\ CD\ RX} \qquad \text{(Eq.10. 5)}$$

Amb aquestes màscares s'assegura que l'equació 10.3 es compleix per a tots els missatges enviats al *Bootloader* des de la font de programació. Permetent realitzar totes les seves funcions.

10.6. Configuració de l'arrencada del *Bootloader*

Per configurar l'arrencada del *Bootloader* cal definir l'identificador del missatge d'inici i el temps que espera el microcontrolador per rebre aquest missatge. En la versió actual del *Bootloader* el missatge d'inici té el mateix identificador que el del filtre de recepció. S'ha deixat l'opció de posar identificadors diferents per a futures versions del *Bootloader*.

Per configurar el temps d'espera (*Timeout*) es carrega amb un valor inicial el comptador del TIMER 0 del microcontrolador [2]. L'equació 10.6 calcula d'aquest valor.

$$Valor\ d'inici\ TIMER0 = FFFFh - \left(\left(\frac{F_{osc}[KHz]}{4 * 256}\right) * timeout[ms]\right) \qquad \text{(Eq.10. 6)}$$

Quan el comptador del TIMER arriba a 0 el *Bootloader* salta al codi de l'usuari. S'ha limitat el temps màxim d'espera a un segon, per evitar esperes excessives cada vegada que s'inicialitza el microcontrolador.

10.7. Botons per executar comandes

Hi ha tres botons de comandes, cadascun té les funcions següents:

- *Help:* Mostra el fitxer d'ajuda del programa
- *Generate files:* Genera els fitxers i obre un quadre de diàleg per escollir la carpeta on guardar cada fitxer.
- *Exit:* Surt del programa

10.8. Aspecte dels fitxers

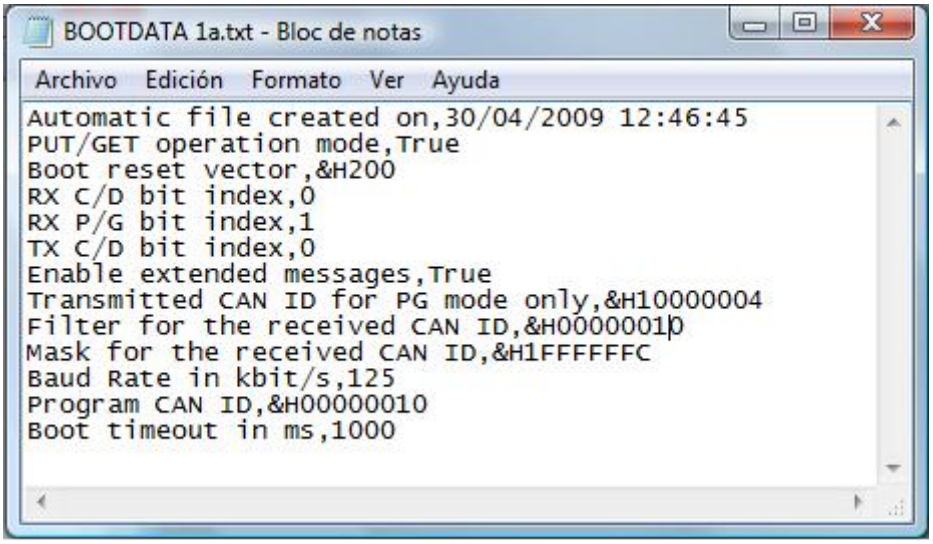

Figura 10.4. Imatge del fitxer *BOOTDATA*.txt

CANBoot.def - Bloc de notas

Archivo Edición Formato Ver Ayuda

```
; Automatic File Created on 30/04/2009 12:46:45
;
; Set the desired mode of operation. BOOT_MODE = 0 -> PG = PUT and GET, BOOT_MODE = 1 ->P = Put
#define BOOT_MODE 0
;
; Set the remapped reset vector.
#define BOOT_RESET_VECT        0x200
;
; Set the remapped high priority interrupt vector.
#define BOOT_HIGH_INT_VECT        0x208
;
; Set the remapped low priority interrupt vector.
#define BOOT_LOW_INT_VECT        0x218
;
; Set the register containing the C/D bit
#define BOOT_CAN_RXCD        RXB0EIDL
;
; Set the C/D bit
#define BOOT_CAN_CD_BIT        0
;
; Set the register containing the P/G bit
#define BOOT_CAN_RXPG        RXB0EIDL
;
; Set the P/G bit
#define BOOT_CAN_PG_BIT        1
;
; Set the register containing the Put bit
#define BOOT_CAN_TXP        TXB0EIDL
;
; Set the Put bit
#define BOOT_CANTX_PG_BIT        0
;
; Set the message type
#define BOOT_CAN_ST_EXT        1
;
; Set the transmitted CAN ID for PG mode only
#define BOOT_CAN_TXB0        0x10000004
;
; Set the filter for the received CAN ID.
#define BOOT_CAN_RXF0        0x00000010

; Set the mask for the received CAN ID.
#define BOOT_CAN_RXM0        0x1FFFFFFC
;
; Set the BRGCON1.
#define _CAN_BRGCON1        0x44
;
; Set the BRGCON2.
#define _CAN_BRGCON2        0x82
;
; Set the BRGCON3.
#define _CAN_BRGCON3        0x02
;
; Constant per determinar el timeout del bootloader
#define BOOT_TIMEOUT        0xB1E0
;
; Set the Program ID
#define BOOT_PROGRAM_ID        0x00000010
```

Figura 10.5. Imatge del fitxer *CANBOOT*.def

11. PROGRAMADOR CAN

Aquest programa serveix per programar, esborrar o llegir els diferents tipus de memòries dels microcontroladors *PIC18FXX8* a través del bus CAN gràcies al *Bootloader* i a la targeta de PC, el seu codi font es pot consultar a l'annex I. Per realitzar qualsevol d'aquestes operacions cal establir comunicació entre el *Bootloader* i el PC. Això es pot fer fàcilment gràcies al fitxer *BOOTDATA*.txt generat amb el programa de definicions.

Una vegada s'hagi establert comunicació, segons el mode del *Bootloader* es poden realitzar diferents operacions. Si està en mode *PUTGET* es pot llegir els diferents tipus de memòries o bé esborrar-les, si està en mode *PUT* només es poden esborrar. En els dos modes es pot programar si s'ha carregat prèviament algun fitxer **.HEX* , el qual s'ha de crear amb l'MPLAB de *Microchip*. Un cop realitzades les operacions cal sortir del mode *Boot* perquè el microcontrolador executi el programa acabat de programar.

11.1. Finestra principal del programador

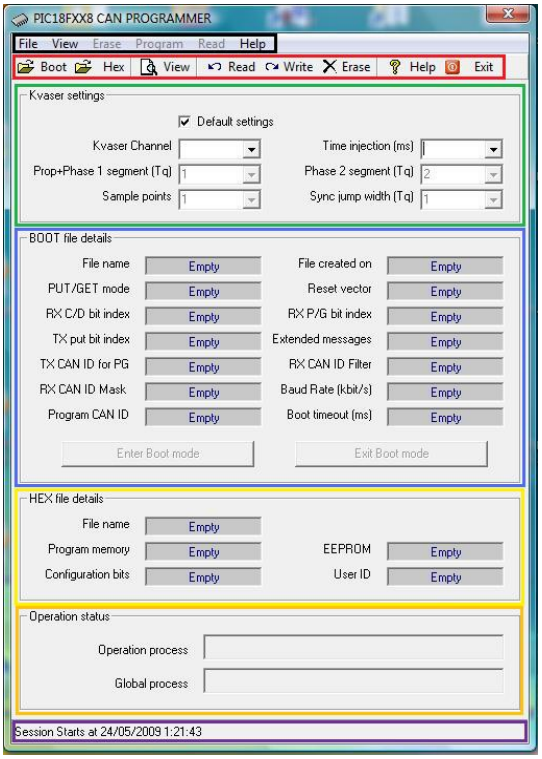

Figura 11.1. Formulari principal del programador CAN

A la figura 11.1 podem veure diferents zones remarcades en colors, aquestes zones són:

- *Negra:* Menús per accedir a totes les funcions del programa.
- *Vermella:* Barra d'eines per accedir ràpidament a les funcions més habituals.
- *Verda:* Paràmetres de configuració de la targeta de PC *Kvaser*.
- *Blava:* Dades del fitxer *BOOTDATA* carregat.
- *Groga:* Detalls del fitxer *HEX* carregat.
- *Taronja:* Barres de progressió dels processos i general.
- *Lila:* Barra d'estat on es dóna informació rellevant.

11.2. Menús i submenús

Els menús i submenús donen accés a totes les funcions següents:

- *File*
    - *Open BOOTDATA File:* Obre un quadre de diàleg que permet escollir el fitxer *BOOTDATA* desitjat.
    - *Open HEX file:* Obre un quadre de diàleg que permet escollir el fitxer *HEX* desitjat.
    - *Exit: Surt del programa.*
- *View*
    - *Memory regions:* Permet veure el contingut de cadascuna de les memòries presents al submenús. Està disponible si s'està en mode *PUTGET* i s'ha llegit la memòria.
        - *Program memory*
        - *EEPROM*
        - *User ID (User Identification)*
        - *Configuration bits*
    - *Bus Activity:* Permet veure les operacions realitzades i els missatges enviats i rebuts durant la sessió.
- *Erase*: Permet esborrar qualsevol de les memòries del submenú.
    - *All*
    - *Program memory*
    - *EEPROM*
    - *User ID*
    - *Configuration bits*
- *Program:* Permet programar les dades a les memòries del submenú, si el fitxer *HEX* conté informació per aquestes.
    - *All available*

  - *Program memory*
  - *EEPROM*
  - *User ID*
  - *Configuration bits*

- *Read:* Permet llegir qualsevol de les memòries del submenú, si s'està en mode *PUTGET*.
  - *All*
  - *Program memory*
  - *EEPROM*
  - *User ID*
  - *Configuration bits*
- *Help*
  - *About*: Permet veure la informació del programa i del sistema.
  - *Help file:* Permet veure el fitxer d'ajuda del programa.

Perquè el programa funcioni correctament, cal controlar en cada moment quins menús i submenús estan activats. És un dels inconvenients de la programació per esdeveniments, és indispensable controlar el flux del programa [4]. Per exemple, és impossible esborrar la memòria si no s'ha entrat primer en mode *Boot* i si s'intentés es donarien un seguit d'errors que l'usuari no entendria. La figura 11.2 mostra el diagrama del comportament dels menús.

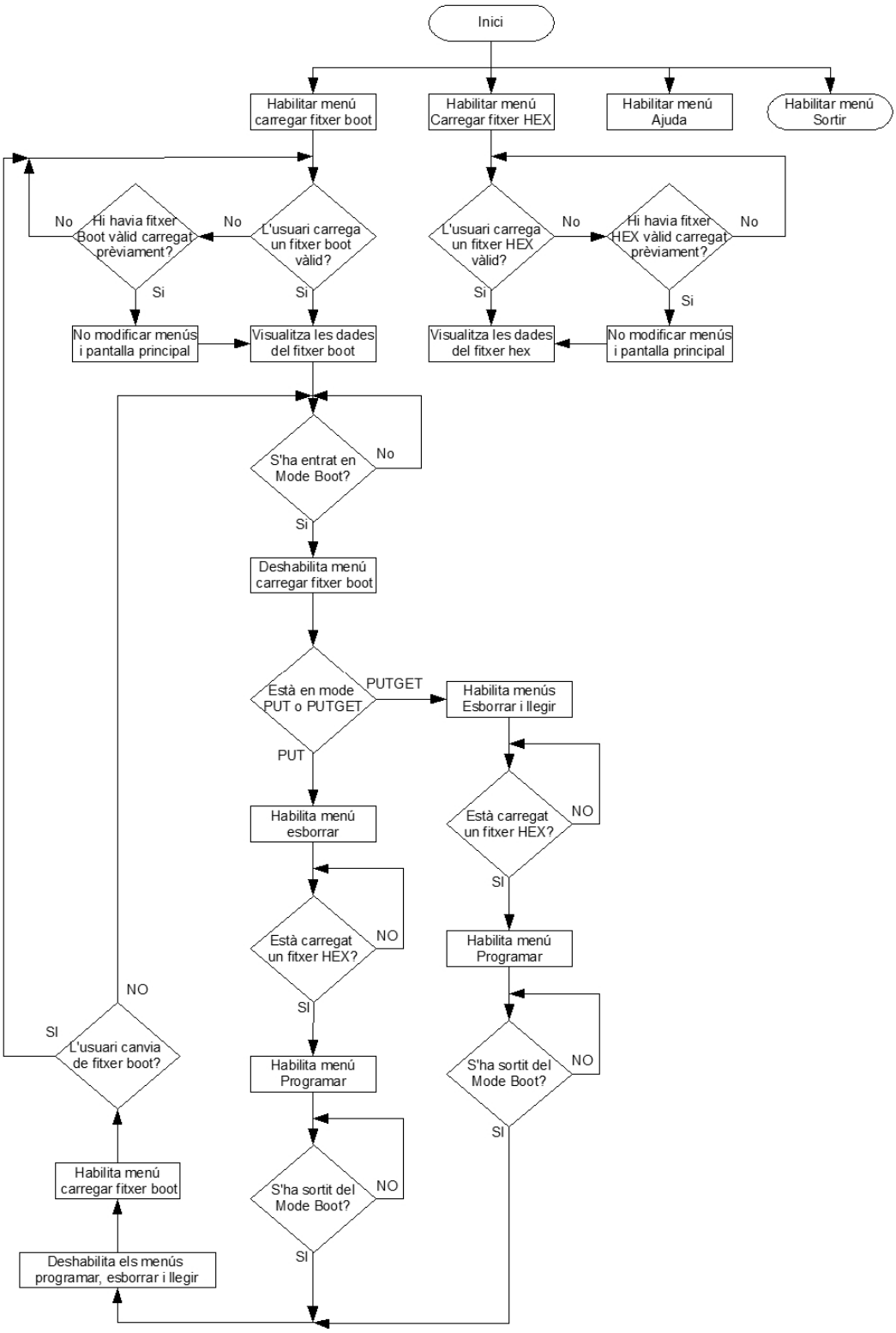

Figura 11. 2. Diagrama de flux de l'habilitació dels menús

Mentre es realitzen funcions de programar, llegir o escriure es deshabiliten tots els menús menys el de sortir, per evitar errors de l'usuari i assegurar el funcionament del programa.

11.3. Barra d'eines

La barra d'eines està formada pels següents botons:

- *Botó Boot*: Permet obrir un fitxer *BOOTDATA*.
- *Botó HEX*: Permet obrir un fitxer *HEX*.
- *Botó View*: Obre els fitxers de lectura de memòria i el d'activitat de bus.
- *Botó Read:* Llegeix totes les regions de memòria, si està en mode *PUTGET*.
- *Botó Write:* Programa al microcontrolador totes les dades del fitxer *HEX* carregat.
- *Botó Erase: Esborra totes les dades del microcontrolador.*
- *Botó Help: Obre el fitxer d'ajuda.*
- *Botó Exit: Surt de l'aplicació.*

Els botons de la barra d'eines estan habilitats si els menús que fan les mateixes funcions estan també habilitats. En cas de que una funció no estigui disponible apareixerà un missatge d'informació informant-ne.

11.4. Paràmetres de configuració de la targeta de PC

La targeta *Kvaser* ofereix unes configuracions de bus predeterminades que serveixen per la majoria de sistemes, si més no en proves de laboratori. Si està marcada la casella de "*Default settings*", s'usen aquests paràmetres per la velocitat de bus carregada pel fitxer *BOOTDATA*. El valor dels paràmetres predeterminats s'indica a la taula 11.1[5].

Baud rate [Kbit/s]	***# T_Q's per bit***	***Propagation+Phase 1 segment [T_Q]***	***Phase 2 segment [T_Q]***	**SJW**	***Sample points***
125	16	10	5	1	1
250	8	4	3	1	1
500	8	4	3	1	1
1000	8	4	3	1	1

Taula 11.1. Valors predefinits de la targeta *Kvaser*

La casella "*Channel*" serveix per seleccionar el canal, i és útil si es disposa de més d'un dispositiu *Kvaser* connectat a l'ordinador. Per saber quin dispositiu correspon a cada canal cal consultar "*Kvaser* hardware" al Panel de control de Windows[5].

La casella "*Time injection*" serveix per deixar el bus lliure durant un temps després d' enviar missatges. Això facilita la comunicació entre els altres nodes i permet que el sistema continuï funcionant, ja que normalment els missatges usats en la programació són d'alta prioritat i guanyen l'arbitratge a la resta de missatges[1]. Per més informació sobre l'arbitratge dels missatges CAN, consultar l'annex C.

Les altres caselles serveixen per configurar els paràmetres temporals de la transmissió del bus CAN, els quals ja s'han explicat en el capítol 9.

11.5. Dades del fitxer *BOOTDATA* carregat

En aquesta secció es mostren les dades que conté el fitxer *BOOTDATA* carregat. També conté dos botons un per entrar al mode *Boot* (activació del *Bootloader*), "*Enter Boot mode*" i l'altre per sortir-ne, "*Exit Boot mode*".

11.6. Dades del fitxer *HEX* carregat

En aquesta secció es mostren les dades del fitxer *HEX*. Es mostren quines àrees de memòria es poden programar segons el fitxer carregat i l'última posició de les dades a escriure a cada memòria.

11.7. Barres de progressió

Hi ha dues barres de progressió. La barra del dalt mostra l'estat dels processos de llegir, programar o esborrar per cada tipus de memòria. La de baix mostra l'estat del procés global, quan es realitzen vàries operacions seguides. Per exemple, si es prem el menú Erase -> All, la barra de dalt mostrarà l'estat de l'esborrament de cada memòria, i la de baix la del procés total, que inclou les 4 memòries.

11.8. Barra d'estat

Informa dels esdeveniments succeïts durant l'execució del programa. També mostra informació durant les operacions, com l'adreça actual que s'està llegint, programant o esborrant.

11.9. El fitxer *.HEX*

El fitxer *.HEX* conté la informació en llenguatge màquina del programa creat pel PIC18F4XX8 amb l'MPLAB. Aquest programa es pot desenvolupar en llenguatge C o bé en assemblador, per posteriorment exportar-lo en format *HEX*.

En un fitxer *HEX* la informació està codificada, per tant, per poder utilitzar les dades cal descodificar-lo. El programa el descodifica i separa en 4 fitxers, un per cada tipus de memòria. Un cop creats aquests fitxers els utilitza per programar el microcontrolador. Els fitxers es guarden al directori del programa i tenen els següents noms: Pic*Boot*S.ConfigFile.txt, Pic*Boot*S.EEDataFile.txt, Pic*Boot*S.ProgMemFile.txt i Pic*Boot*S.UserIDFile.txt.

Per més informació sobre la codificació dels fitxers *HEX* consultar l'annex K.

11.10. Entrada en mode *Boot*

Per entrar en mode *Boot* primer cal carregar un fitxer *BOOTDATA*.TXT i prémer el botó "*Enter Boot mode*". Un cop premut el botó es disposa de 30 segons per reiniciar el microcontrolador.

Durant aquests 30 segons el programa intenta arrencar el *Bootloader*. Si el *Bootloader* està en mode *PUT* es van enviant missatges mentre no siguin llegits, un cop ha estat llegit un d'ells es considera que la centraleta l'ha rebut i que el *Bootloader* s'ha iniciat. Si està en mode *PUTGET* es van enviant missatges d'inici, quan aquest és llegit pel *Bootloader* aquest ha de respondre amb un missatge d'*Acknowledge*. Si no respon es considera que no s'ha entrat en mode *Boot* i es repeteix l'operació. En la figura 11.3 es mostra com funciona l' inici del *Bootloader* en mode *PUTGET*.

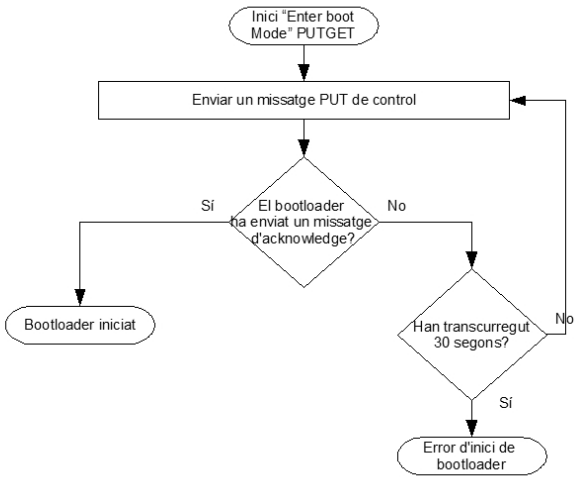

Figura 11. 3. Diagrama de flux de l'entrada en mode *Boot* en mode *PUTGET*

En mode *PUT* no hi ha manera de saber si s'ha entrat correctament en mode *Boot*, ja que la comunicació és unidireccional. Si no s'ha entrat es farà evident quant s'intenti fer executar alguna comanda al *Bootloader* i aquest no la realitzi. Si és així, cal tornar intentar l'entrada al mode *Boot*.

11.11. Mètode de Programació

El mètode de programació és diferent segons el mode de funcionament del *Bootloader*. Si s'està en mode *PUT* no es poden llegir les dades escrites. Per comprovar que s'hagin escrit correctament les dades, el programa i el *Bootloader* realitzen un *Checksum* de totes les dades.

Degut a les diferències físiques de les memòries, no totes es poden escriure de la mateixa manera. La *Program memory* i *l'User id* poden escriure 8 bytes a la vegada, en canvi, la EEPROM i els *Configuration bits* només un byte. Per tant, quan es programin les primeres s'enviaran missatges de dades amb 8 bytes, i amb les segones missatges de dades amb un sol byte.

Cada tipus de memòria necessita d'un temps per ser programada, per l'EEPROM són 4ms i per la resta de memòries 2ms. Per aquest motiu, el programador espera aquest temps després d'enviar les dades a programar, així el *Bootloader* rep sempre els missatges quan està desocupat. D'aquesta manera s'evita que no es llegeixin missatges que s'han rebut mentre es programava la memòria.

Si es volen programar els *Configuration bits* apareix un missatge d'advertiment que avisa que alguns canvis poden provocar un mal funcionament del *Bootloader*. Per prevenir descuits de l'usuari, no es permet modificar els bits de protecció de *Boot* i de rellotge.

A continuació, es mostren les figures 11.4 i 11.5 que són diagrames de l'operació d'escriptura d'una posició de memòria segons el mode de funcionament del *Bootloader*. Per programar tota la memòria s'ha de repetir aquesta operació fins que totes les dades estiguin gravades.

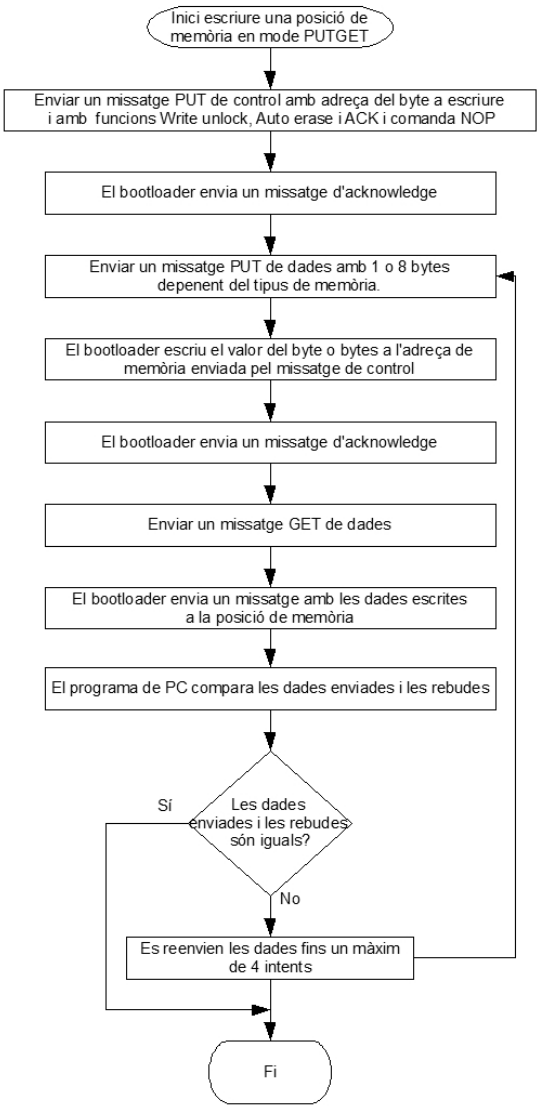

Figura 11 .4. Diagrama de flux d' escriure una posició de memòria en mode *PUTGET*

A la figura 11.4 s'observa que si les dades enviades i rebudes pel PC no són iguals es reenvien. Les dades es poden reenviar fins a 4 vegades, si en cap dels intents coincideixen es mostra un missatge d'error i s'interromp l'operació d'escriptura a la memòria. Aquest fet pot indicar que s'han modificat incorrectament alguns *Configuration bits* o bé que hi ha alguna posició de memòria feta malbé dins del microcontrolador.

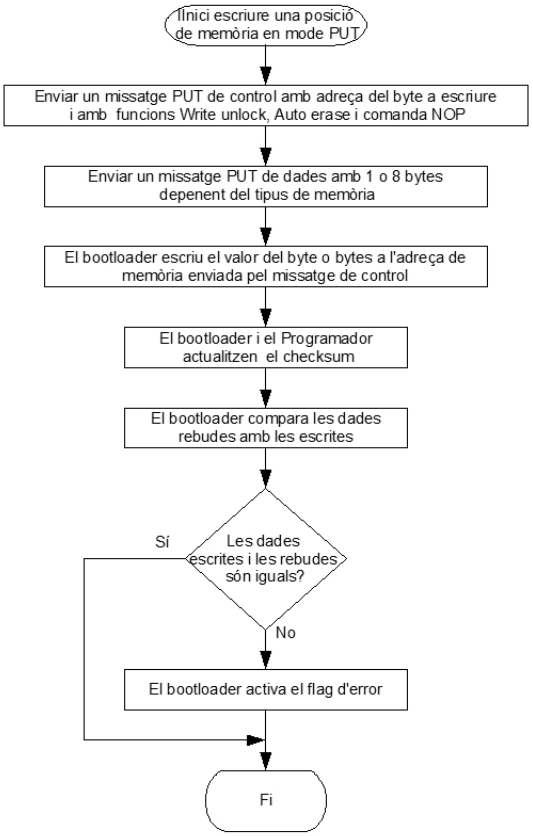

Figura 11.5. Diagrama de flux d' escriure una posició de memòria en mode *PUT*

Com s'observa en la figura 11.5, la font de programació i el microcontrolador calculen un *Checksum* independentment. Aquest *Checksum* s'utilitzarà just abans de sortir del mode *Boot* per comprovar que les dades enviades i les rebudes eren iguals, detectant així algun problema ocorregut durant la transferència de les dades a través del bus

A part hi ha el *Flag* d'error d'escriptura, que s'encarrega de comprovar de que les dades rebudes i escrites són iguals, per detectar així algun problema a la memòria.

Per comprovar el *Checksum* i el *Flag* d'error s'envia un missatge de control amb la comanda "*Check and run*" i el *Checksum* calculat pel PC. Si coincideixen els *Checksum*s i no hi ha hagut errors d'escriptura el *Bootloader* reinicia el microcontrolador. D'aquesta manera s'assegura que les dades enviades i les escrites són iguals i que la programació a reeixit.

Es recomana programar els *Configuration bits* al final, ja que si es modifica algun d'aquests erròniament pot deixar de funcionar correctament el *Bootloader*, però la resta de memòries ja estaran gravades. Es recomana l'ordre següent[1]:

1. Memòria de programa
2. *User id*
3. EEPROM
4. *Configuration bits*

11.12. Mètode d' esborrament

Per esborrar les memòries s'ha d'escriure el valor FFh a totes les seves posicions. Per esborrar més ràpid la memòria de programa, que és la més estesà, s'usa la funció A*uto-erase* del *Bootloader*. Així per cada missatge enviat s'esborren 64 posicions consecutives de la memòria, enlloc de una com en les altres memòries[1].

Alhora d'esborrar els *Configuration bits* es mostra un missatge d'advertiment, ja que si s'esborren els *Configuration bits* el sistema pot deixar de funcionar correctament. Per prevenir aquesta situació, no es permet esborrar els *Configuration bits* vitals pel funcionament del *Bootloader*, com el de protecció del sector de *Boot* i el del rellotge[1].

No hi ha cap funció que asseguri que el microcontrolador està totalment en blanc. Però, com que esborrar és equivalent a escriure a totes les posicions FFh, l'esborrament serà sempre correcte, perquè el mètode de programació assegura que les dades han estat escrites correctament.

11.13. Mètode de lectura

Només es pot llegir la memòria en mode *PUTGET*, es realitza sempre byte a byte per tot tipus de memòria. Com que tota la memòria està mapejada es pot llegir qualsevol memòria només canviant l'adreça. Per fer més fàcil a l'usuari la lectura de les dades, es guarden les dades en fitxers diferents per cada tipus de memòria. Aquests fitxers són guardats al

directori del programa i tenen els noms següents: Configbits.txt, EEPROM.txt, Prog_mem.txt i USERID.txt. Per accedir a aquests fitxers cal prémer el menú *View->Memory regions*.

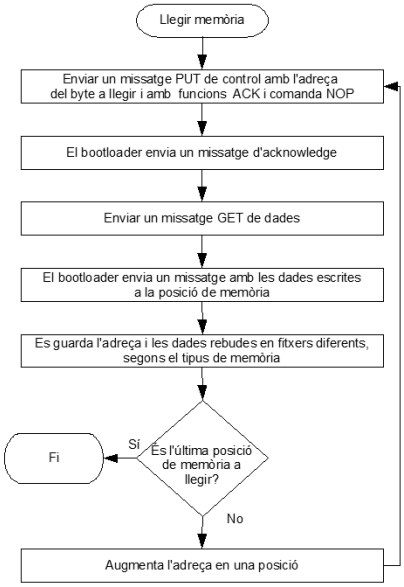

Figura 11.6. Diagrama de flux de llegir una posició de memòria

11.14. Sortir del mode *Boot*

Per sortir del mode *Boot* cal prémer el botó "*Exit Boot mode*". Aquest botó realitza diferents accions segons el mode de funcionament del *Bootloader*.

Si està en mode *PUT* envia un missatge de control amb una comanda "*Check and run*" i els *Checksum*s calculats pel PC. Com s'ha dit en l'apartat mètode de programació, el *Checksum* calculat pel microcontrolador i el del PC han de coincidir. A més, el *Flag* d'error d'escriptura ha d'estar desactivat perquè es reiniciï el microcontrolador. Un cop reiniciat es saltarà al programa d'usuari després del *Timeout* del *Bootloader*.

Si està en mode *PUTGET* només cal enviar un missatge de control amb la comanda *Reset*, perquè ja s'han comprovat les dades escrites durant la programació. Un cop reiniciat saltarà al programa d'usuari després del *Timeout* del *Bootloader*.

11.15. Gestió d'errors

Quan succeeix un error el programa mostra un quadre de diàleg d'error indicant que ha fallat, i seguidament un altre dient a quina acció ha afectat. Els errors ocorreguts durant una

sessió es guarden al arxiu Busactivity.txt, juntament amb altre informació rellevant. Per accedir a aquest fitxer, cal anar al menú *View-> Bus Activity*. Després de que succeeixi un error es torna a l'estat previ de l'acció que no ha funcionat correctament.

11.16. Usabilitat

Segons la norma ISO9126, la usabilitat és la capacitat d'un software de ser comprès, après, usat i ser atractiu per l'usuari en condicions d'us. El programador s'ha dissenyat de tal manera que es pot accedir a qualsevol operació realitzant pocs *clics*, per exemple, per esborrar la memòria són necessaris només 4 *clics*. A més, es deixa sempre visible la informació del fitxer *BOOTDATA* i *HEX* per saber que s'està fent en qualsevol moment.

12. TEST DEL SISTEMA

Per comprovar el funcionament del sistema s'han fet diferents proves, en totes s'ha programat correctament el programa en format *HEX* enviat des del programador. Per fer les proves s'han fet servir tres programes en format *HEX*, el primer fa encendre i apagar 2 leds que té la placa, el segon envia i rep missatges CAN i el tercer fa servir una interrupció de baixa prioritat per encendre els leds. El codi font de cadascun d'aquests programes es pot consultar usant l'annex J.

Per realitzar les proves cal programar el *Bootloader* a la centraleta. Per fer-ho, primer cal executar el programa de definicions i entrar-li els paràmetres desitjats. El programa crearà els fitxers *BOOTDATA*.txt i *CANBOOT*.def. Seguidament, s'usarà el *CANBOOT*.def per compilar amb l'MPLAB el *Bootloader* segons els paràmetres desitjats. Finalment, es gravarà al microcontrolador des del MPLAB usant el programador ICD2. Un cop gravat el *Bootloader* ja es poden començar les proves amb el programador. Pel programador cal fer servir el fitxer *BOOTDATA*.txt creat juntament amb el *CANBOOT*.def per entrar en mode *Boot*. Un cop en mode *Boot* s'esborrarà tota la memòria menys els *Configuration bits* i es programarà el programa en format *HEX*. Després es sortirà del mode *Boot* i es comprovarà el correcte funcionament del programa.

12.1. Proves

En totes les proves s'ha aconseguit programar correctament i el bus ha continuat funcionant. Amb la qual cosa es prova el funcionament de tot el sistema: programa de definicions, *Bootloader* i programador CAN. Les proves realitzades són les següents:

- Proves inicials amb una sola centraleta. Es fan proves en mode *PUT* i *PUTGET* a diferents velocitats de bus i amb formats de missatge estàndard i estès.
- Proves en mode *PUTGET* i 4 centraletes, amb velocitat de bus de 125 kbit/s. En totes s'ha utilitzat el programa de monotorització del bus del *Kvaser*, el CANKing, amb el qual s'ha comprovat que no hi havia *Error frames* al bus. S'ha usat el format de missatge estès. Per més informació sobre els *Error frames* consultar l'annex C.
    - S'han programat tres centraletes amb el programa que envia missatges CAN i la restant amb el *Bootloader* amb l'ICD2. Seguidament, s'ha enviat amb el programador CAN el programa que encén els leds a la centraleta amb *Bootloader*.

- S'han programat dues centraletes amb el programa que envia missatges CAN i dues amb diferents *Bootloader*s amb l'ICD2. A continuació, amb el programador CAN s'ha enviat el programa que envia i rep missatges CAN a les dues centraletes amb *Bootloader*.
- S'han programat totes les centraletes amb *Bootloader*s diferents amb l'ICD2. Després, s'ha enviat el programa que encén els leds amb el programador CAN a cadascuna de les centraletes.
- S'han programat tres centraletes amb el programa que envia missatges CAN i la restant amb un *Bootloader* amb l'ICD2. Tot seguit, s'ha enviat amb el programador CAN el programa que encén els leds amb interrupcions a la centraleta amb *Bootloader*.

- Prova en mode *PUT* amb 4 centraletes. S'ha programat el mateix *Bootloader* a totes les centraletes amb l'ICD2. Seguidament, s'ha enviat el programa de leds amb el programador CAN, quedant totes les centraletes programades d'un sol cop.
- Prova a 250kbit/s amb 4 centraletes i usant el CANking. S'han programat dues centraletes amb el programa que envia missatges CAN i dues amb diferents *Bootloader*s amb l'ICD2. A continuació, s'ha enviat amb el programador el programa que envia i rep missatges CAN a les dues centraletes amb *Bootloader*.

Cal remarcar la importància d'escollir uns paràmetres del temps de bit del CAN correctes, ja que en les proves ha calgut modificar-los pel correcte funcionament del sistema.

12.2. Taula de temps per les diferents operacions

A les taules 13.1 i 13.2 es mostra el temps en segons, que tarda el programador a realitzar cadascuna de les operacions segons el mode de funcionament. Es considera que es programen completament les memòries i que la velocitat de bus es de 125kbit/s. Per tant, aquest és el pitjor cas. Es pot observar que les operacions en mode *PUT* són més ràpides degut a que la comunicació és unidireccional.

Mode PUT	***All [s]***	***Program memory [s]***	***EEPROM [s]***	***User id [s]***	***Configuration bits [s]***
Erase	8	6	1	<1	<1
Program	49	46	2	<1	<1

Taula 12.1. Temps de les operacions en mode *PUT*

Mode PUTGET	***All [s]***	***Program memory [s]***	***EEPROM [s]***	***User id [s]***	***Configuration bits [s]***
Erase	10	6	3	<1	<1
Program	56	52	3	<1	<1
Read	18	16	<1	<1	<1

Taula12.2. Temps de les operacions en mode *PUTGET*

13. PROPOSTES I RECOMANACIONS

13.1. Propostes

Tot i que en la versió actual del sistema l'esborrat és correcte, no té cap funció de *Blank Check*, o sigui de comprovació de que tota la memòria està buida. Això aportaria la seguretat a l'usuari de que inicia la programació en un dispositiu buit. Per poder realitzar aquesta funció s'hauria de modificar el *Bootloader* i el programador. En mode *PUT* GET és senzill, ja que només caldria modificar el programador perquè mentre llegeixi comprovi que les dades han estat correctament esborrades. En el mode *PUT* és més complicat ja que requereix modificar el *Bootloader*.

Una altra proposta és poder identificar el microcontrolador a programar, així tot el sistema es podria utilitzar en altres dispositius de *Microchip*.

13.2. Recomanacions

13.2.1. Elecció dels paràmetres CAN.

Es recomana fer servir els paràmetres predefinits del *Kvaser*, ja que són un bon compromís i funcionen correctament per la majoria de xarxes CAN.

13.2.2. Elecció del mode de funcionament del *Bootloader*

Depenent de com s'hagin de programar les centraletes es recomana el següent:

- Per a programar totes les centraletes amb el mateix programa es recomana el mode *PUT* i gravar a totes el mateix *Bootloader*. D'aquesta manera amb una sola operació d'escriptura es gravarà el mateix programa a totes les centraletes.
- Si es vol programar cada centraleta amb un programa diferent es recomana fer servir el mode *PUTGET* i gravar a cadascuna un *Bootloader* diferent.
- Si es vol llegir la memòria del microcontrolador, només es pot fer en mode *PUTGET*.

13.2.3. Consells d'utilització del programador

Es recomana esborrar abans de programar, ja que quan es programa només s'esborren els 64 blocs següents a la finalització del programa.

14. COSTOS

14.1. Cost del projecte

Al tractar-se d'un projecte integrament de software, no hi ha pràcticament costos de material. Si que hi ha amortitzacions del material usat durant la realització i test del software. A la taula 14.1 es mostra el material usat, del qual es calcula un 50% del cost de compra en amortització durant la realització del projecte.

Quantitat	Material	Preu compra[€]	Amortització [€]
1	PC portàtil amb llicència Windows	700€	350€
4	Plaques amb microcontrolador *PIC18F*	400€	200€
1	Targeta de PC a CAN *Kvaser Leaf ligth HS*	250€	125€
1	Llicència de *Visual Basic 6.0*	500€	250€
1	Programador ICD2 de *Microchip*	120€	60€
Total amortitzacions		**1970€**	**985€**

Taula 14.1. Cost de les amortitzacions del material

Pel que fa als costos de personal, es deu integrament a les hores dedicades per un enginyer especialitzat en programació. Dins de les hores es calcula el temps de documentació, desenvolupament i test de tot el software.

Hores	Concepte	Preu hora [€/h]	Total[€]
400	Hores d'enginyer	50€/h	20000€

Taula 14.2. Cost del personal

Així doncs, unificant els costos d'amortitzacions i personal s'obté:

Cost total del projecte[€]	
Cost total amortitzacions	985€
Cost total personal	20000€
Cost total	**20985€**

Taula 14.3. Cost total del projecte

14.2. Cost per l'empresa d'automòbils

Per una empresa automobilística programar les centraletes dels vehicles a través del bus CAN suposa una reducció dels estocs[11]. A l'empresa no li cal tenir centraletes preprogramades segons les opcions i model de vehicle, totes poden ser iguals i ser programades a la mateixa cadena de muntatge. Només cal que les centraletes disposin del *Bootloader* CAN i instal·lar a la cadena de muntatge diferents PC amb programadors CAN, així es programen les centraletes un cop connectades al bus del vehicle.

14.2.1. Inversió inicial empresa

A part del cost del projecte, l'empresa ha d'invertir al magatzem i a la cadena de muntatge. Els càlculs d'inversions següents són orientatius, ja que per cada empresa suposarà un cost diferent. Per la cadena de muntatge s'estima que fan falta 10 estacions de programació, formades per un PC portàtil i una targeta de PC –CAN. La inversió inicial de l'empresa es mostra a la taula 14.4.

Inversió	**Cost[€]**
Projecte	20985€
Reorganització magatzem	10000€
Estacions de programació	9500€
Cost total	**40485€**

Taula 14.4. Inversió *inicial* empresa

14.2.2. Estalvi empresa

Els càlculs d'estalvi també són orientatius, ja que per cada empresa suposarà un estalvi diferent. Per tal de calcular l'estalvi que suposa aquesta reducció d'estocs, s'estima que l'empresa disposa d'un magatzem adjacent en lloguer al costat, on emmagatzema les centraletes abans de ser utilitzades a la cadena de muntatge. Suposant que pot llogar únicament els metres quadrats que requereixi i que s'estalvia uns 200m^2 de magatzem, l'estalvi seria el mostrat a la taula 14.5. Incloent en la mensualitat el lloguer (6 €/m2·mes) i els serveis del magatzem (10 €/m^2·mes), que són les despeses d'energia i manteniment.

Mesos	**Mensualitat**	**Superfície[**m^2]	**Estalvi anual[€]**
12	$16\,\frac{€}{m^2 \cdot mes}$	200m^2	38400€

Taula 14.5. Estalvi anual de l'empresa

D'aquest estalvi cal descomptar el manteniment de les estacions de programació. Es calcula que el manteniment d'aquestes serà d'uns 5000€ anuals.

Llavors l'estalvi total seria de:

Concepte	**Estalvi[€]**
Magatzem	38400€
Manteniment estacions de programació	-5000€
Total estalvi	33400€

Taula 14.6. Estalvi total de l'empresa

14.2.3. Rendibilitat de la inversió

La rendibilitat d'inversió s'ha calculat a 5 anys, amb una taxa d'interès del diner del 3%. A la taula 14.7 es pot observar els fluxos que genera la inversió.

Any	0	1	2	3	4	5
Inversió	40.485 €	-	-	-	-	-
Estalvi	-	33.400 €	34.402 €	35.434 €	36.497 €	37.592 €
Moviment de fons	-40.485 €	33.400 €	34.402 €	35.434 €	36.497 €	37.592 €
Fluxos acumulats	-40.485 €	-7.085 €	27.317 €	62.751 €	99.248 €	136.840 €

Taula 14.7. Rendibilitat de la inversió

Amb aquests moviments de fons podem trobar el valor actualitzat net (VAN), que és de 121.650€ . A més el PAY BACK és de 1,2 anys. Pel que fa la taxa interna de rendibilitat (TIR) calculada és del 75%, molt superior al interès donat per les entitats financeres.

15. IMPACTE AMBIENTAL

Al tractar-se d'un projecte íntegrament de software no hi ha interacció amb el medi ambient. Els diferent aparells usats per realitzar-lo com el PC, la targeta de PC i la placa del microcontrolador no formen part de l'estudi.

Com que el software creat en aquest projecte pretén reduir costos d'estoc de centraletes en les empreses d'automòbils, es podria reduir l'espai dels magatzems de les fàbriques. D'aquesta manera es reduirien els residus generats durant la construcció dels magatzems i l'energia que s'utilitza pel seu manteniment.

CONCLUSIONS

L'objectiu principal del projecte era poder programar fàcilment les centraletes de la família *PIC18FXX8* de *Microchip* a través del bus CAN. Aquest objectiu s'ha aconseguit amb èxit, ja que els 2 programes en Visual Basic i el *Bootloader* han superat totes les proves de funcionament. Una vegada realitzades aquestes proves, han suggerit una sèrie de propostes per millorar el software i recomanacions per facilitar l'obtenció de bons resultats en l'ús del software.

Tot i que el software està dissenyat específicament per a uns dispositius, el projecte m'ha aportat una visió global en l'aplicació industrial de la programació a través de bus. Invertir en aquest tipus de software és rendible per les empreses de l'automòbil, permet un estalvi d'estocs ja que les centraletes connectades al bus del vehicle es programen en la mateixa cadena de muntatge.

Pel que fa a la realització del projecte, interpretar el funcionament del *Bootloader* a partir de la informació subministrada per *Microchip* ha estat la part que m'ha requerit més temps, perquè es dóna el codi en llenguatge assemblador i amb breus explicacions. Un cop entès el seu funcionament ja em permetia començar a crear els programes en Visual Basic. Aquesta part m'ha suposat un esforç perquè no tenia experiència en aquest llenguatge de programació. Finalment, el test del sistema ha estat la part més interessant, pel fet de cercar solucions als errors que anaven sorgint.

He adquirit nous coneixements sobre el funcionament dels busos de comunicació i de la programació de centraletes i PC. Tots aquests temes són aplicables en els diferents camps de la indústria on hi intervé l'ús de l'electrònica, no només per l'automoció.

- Els busos s'utilitzen per connectar multitud de sistemes, ja que redueixen el cost de cablejat en la comunicació.
- Els coneixements de programació de microcontroladors són necessaris en la majoria de projectes d'electrònica.
- La programació de PC és útil en tots els camps de l'enginyeria, permet l'estalvi de tasques repetitives o de càlcul per a l'enginyer.

He procurat que els programes fossin el més útils i pràctics possibles per a l'usuari. L'arrencada del *Bootloader* des del bus suposa un avantatge clar sobre el *Bootloader* de *Microchip*, que requeria escriure a la memòria EEPROM interna al microcontrolador. La configuració del programador mitjançant un fitxer evita molts errors, donat que no és

necessari introduir de nou els paràmetres del *Bootloader*. A més, s'ha creat un fitxer d'ajuda a mode de manual per resoldre els dubtes sobre el funcionament del programa.

Finalment, destacar l'experiència positiva que ha suposat aquest projecte. El procés de desenvolupament d'un projecte, des d'una idea inicial als resultats, el valoro com a un primer apropament a la tasca que hauré de realitzar en el món laboral, on l'adquisició ràpida i contínua de nous coneixements, la relació entre companys enginyers i la recerca de nous plantejaments i idees seran els eixos del treball com a enginyer.

AGRAÏMENTS

Vull agrair al meu director Juan Manuel Moreno Eguílaz la paciència i suport durant la realització del projecte. Les seves idees i recomanacions m'han encaminat a aconseguir els objectius del projecte més fàcilment.

Als meus amics de l'ETSEIB, per la seva companyia i el seu suport al llarg de la carrera. Sense ells les classes no haurien estat el mateix.

A la meva família, xicota i amics pel seu suport i solidaritat durant la realització del projecte. Han estat uns mesos difícils, en els que m'hauria agradat passar més temps amb ells. Sobretot amb la meva neboda nascuda el 21 d'abril del 2009, la qual espero es dediqui al món de l'enginyeria.

BIBLIOGRAFIA

Referències bibliogràfiques

[1] *MICROCHIP*. *AN247*. *Microchip* Technology INC. 2003

[2] *MICROCHIP*. *PiC18FXX8 Data sheet*. *Microchip* Technology , INC.2004.

[3] FREESCALE. *AN1798.* Freescale Semiconductor, Inc. 2004.

[4] SILLER,B. SPOTTS, J. FRAGUAS,S. *Edición Especial Visual Basic 6*.Prentice Hall Iberia.1999.

[5] *KVASER*. *CANLIB Help*. *Kvaser* AB. 2008

[6] *MICROCHIP*. *AN228. Microchip* Technology, INC. 2002

[7] PFEIFFER, O. *Embedded networking with CAN and CANopen*. RTC books.2003

[8] *KVASER*. *Kvaser Leaf ligth HS user guide*. *Kvaser* AB. 2006.

[9] INTREPID. *Microchip CAN Bit Timing Calculator*. Intrepid Control Sistems, INC. 2008.

[10] BOSCH. *CAN specification 2.0* . Robert Bosch GmbH. 1991.

[11] SOFTING AG. *Flashing ECUS on the assembly line*. Haar, 2009. [http://www.softing.com/home/en/automotive-electronics/services/flash-programming/flashing.php, 3 de febrer de 2009]

Bibliografia complementària

VECTOR CONSULTING SERVICES GMBH. *CandelaFlash 2.0.* Stuttgart ,2009. [http://www.vector.com/vi_candelaflash_functions_en,,223.html, 4 de febrer de 2009]

CAN IN AUTOMATION (CIA). Controller Area Network (CAN). Nuremberg, 2009. [http://www.can-cia.de/index.php?id=170, 10 de Novembre de 2009]