SUMARI

A. ÍNDEX DE FIGURES 4

B. ÍNDEX DE TAULES 6

C. CAN(CONTROLLER AREA NETWORK) 7

C.1. Descripció del CAN 7
C.2. Funcionament del bus CAN 9
C.2.1.Descripció d'una xarxa CAN 9
C.2.2. Nivells de Bus 9
C.3. Missatges CAN 11
C.3.1. SOF (Start of frame) 11
C.3.2. Camp d'arbitratge (Arbitration field) 12
C.3.2. Camp de control (*Control Field*) 13
C.3.3. Camp de dades 13
C.3.4.Camp CRC 13
C.3.5. Camp de reconeixement (ACK) 14
C.3.6 EOF(End of frame) 14
C.4.Trama d'errors (Error frame) 15
C.5 Arbitratge CAN 16

D. PLACA D' ELECTRÒNICA DE L'AUTOMÒBIL 17

D.1. Esquemes 17
D.2 Imatges 20
D.3. Característiques dels microcontroladors PIC18FXX8 20
D.3.1. Configuration bits 21
D.3.2.Registres CAN 21

E. *KVASER* LEAF LIGTH HS 22

E.1. Característiques tècniques 22
E.2. Connexions CAN 23
E.3. Imatges 23
E.4. Llibreria CANLIB 23

F. MICROSOFT VISUAL BASIC 6.0 24

F.1. Microsoft Visual Basic 6.0 24
F.2. Creació d'una aplicació 25

G. CODIS FONT DELS *BOOTLOADERS* 26

G.1. Codi font del *Bootloader* modificat 26
G.2. Bootloader de Microchip 26

H. CODI PROGRAMA DEFINICIONS. 29

I. CODI PROGRAMADOR CAN. 30

J. CODI PROGRAMES DE TEST 31

J.1. Programa Leds 31
J.2. Programa missatges CAN 31
J.3. Programa interrupcions 31

K. CODIFICACIÓ DELS FITXERS *.HEX 32

L. MANUAL D'USUARI 33

L.1. Instal·lació 33
L.2. Utilització del software 33
L.2.1.Escollir els paràmetres del *Bootloader* 34
L.2.2. Compilar el *Bootloader* amb l'MPLAB 36
L.2.3. Gravar amb un programador de *Microchip* el *Bootloader* compilat anteriorment al microcontrolador. 37
L.2.4. Muntar la xarxa CAN, connectant la targeta *Kvaser* leaf light HS al bus. 38
L.2.5. Obrir el fitxer BOOTDATA.txt i entrar en mode boot amb el programador CAN. 38
L.2.6. Exportar un fitxer HEX i obrir-lo amb el programador. 40
L.2.7.Esborrar i Programar el microcontrolador. Llegir si és possible. 41
L.2.8.Sortir del mode boot 41

A. Índex de figures

Figura C.1. Capes del protocol definides per la norma ISO-11898 _____ 8

Figura C.2. Descripció de la Xarxa CAN _____ 9

Figura C.3. Estats possibles del Bus _____ 10

Figura C.4. Voltatges diferencials de cadascun dels estats _____ 10

Figura C.5. Camps dels missatges dades _____ 11

Figura C.6. Camps dels missatges de petició remota _____ 11

Figura C.7. Descripció del camp d'arbitratge i control dels missatges estándar _____ 12

Figura C.8. Descripció del camp d'arbitratge i control dels missatges estesos _____ 12

Figura C.9. Contingut del camp de control _____ 13

Figura C.10. Contingut del camp de dades _____ 13

Figura C.11. Contingut del camp de CRC. _____ 14

Figura C.12. Contingut del camp ACK. _____ 14

Figura C.13. Contingut del camp EOF _____ 14

Figura C.14. Trama d'error _____ 15

Figura C.15. Arbitratge CAN _____ 16

Figura D.1. Esquema de l'alimentació i Microcontrolador de la placa _____ 17

Figura D.2. Esquema de les comunicacions CAN i RS232 de la placa. _____ 18

Figura D.3. Esquema dels sensors i actuadors de la placa. _____ 19

Figura D.4. Imatge de la placa _____ 20

Figura E.1. Esquema de les connexions del *Kvaser* _____ 23

Figura E.2. Imatge del *Kvaser* Leaf Ligth HS. _____ 23

Figura G.1. Diagrama de flux del funcionament del *Bootloader* en mode *PUT* 27

Figura G.2. Diagrama de flux del funcionament del *Bootloader* en mode *PUTGET* 28

Figura L.1. Finestra del programa de definicions amb les dades desitjades. 34

Figura L.2. Avís amb informació dels paràmetres per defecte 35

Figura L.3. Quadres de diàleg per guardar els arxius 35

Figura L.4. Avís de la creació correcte dels arxius 36

Figura L.5. Afegir el fitxer CANBOOT.def. 36

Figura L.6. Compilació del *Bootloader*. 37

Figura L.7. Selecció del programador 37

Figura L.8. Programació del *Bootloader* al microcontrolador . 38

Figura L.9. Connector del *Kvaser*. 38

Figura L.10. Obrir fitxer BOOTDATA. 39

Figura L.11. Finestra d'informació per entrar en mode *boot.* 39

Figura L.12. Finestra que informa que s'ha entrat correctament al mode *boot* 39

Figura L.13. Exportar a fitxer hex des de l'MPLAB 40

Figura L.14. Obrir fitxer hex amb el programador 40

Figura L.15. Missatge informant que s'ha sortit del mode *boot* correctament 41

B. Índex de taules

Taula C.1. Principals protocols d'alt nivell _____ 8

Taula C.2. Us de les parts del camps d'arbitratge _____ 12

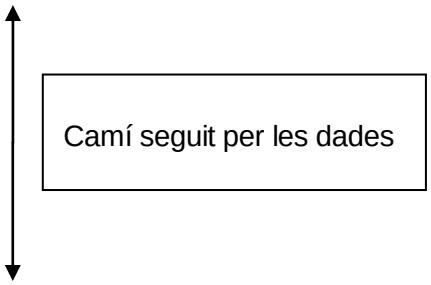
Taula E.1. Característiques tècniques de la sèrie *Kvaser* Leaf _____ 22

Taula E.2. Us de les parts del camps d'arbitratge _____ 22

C. CAN(Controller Area Network)

El CAN és un protocol de comunicació. La majoria de protocols estan definits usant el model OSI (Open System Interconnection), de la norma ISO 7498-1:1994. Aquest model divideix un protocol de comunicació en 7 nivells, cada nivell realitza unes funcions per a les capes contigües. D'aquesta manera les dades arriben correctament del medi físic (xarxa) a l'aplicació o a l'inversa.

1. Nivell físic
2. Nivell d'enllaç de dades
3. Nivell de xarxa
4. Nivell de transport
5. Nivell de sessió
6. Nivell de presentació
7. Nivell d'aplicació



C.1. Descripció del CAN

El CAN va ser creat per BOSCH l'any 1986 per connectar centraletes d'automoció. A l'any 1991 va esdevenir un estàndard, el ISO-11898. A continuació, es descriuen els nivells descrits per la norma ISO-11898-2 que és una revisió de la introduïda l'any 1991.

La norma creada per BOSCH estableix el següent:

- El nivell físic usa un parell de cables trenats amb transmissió diferencial.
- La comunicació és tipus sèrie.
- Per controlar l'accés al bus s'usa un arbitratge no destructiu.
- S'usen missatges per enviar les dades. Els missatges són curts, de com a molt 8bytes. Es controla la seva integritat amb un Checksum.
- Els missatges no tenen una adreça explícita. Cada missatge conté un valor numèric, l'identificador, que controla la seva prioritat al bus i informa sobre el seu contingut.
- Hi ha un control exhaustiu dels errors, i els missatges rebuts malament són reenviats.
- Hi ha mètodes efectius per aïllar les falles i extreure els nodes que fallen del bus.

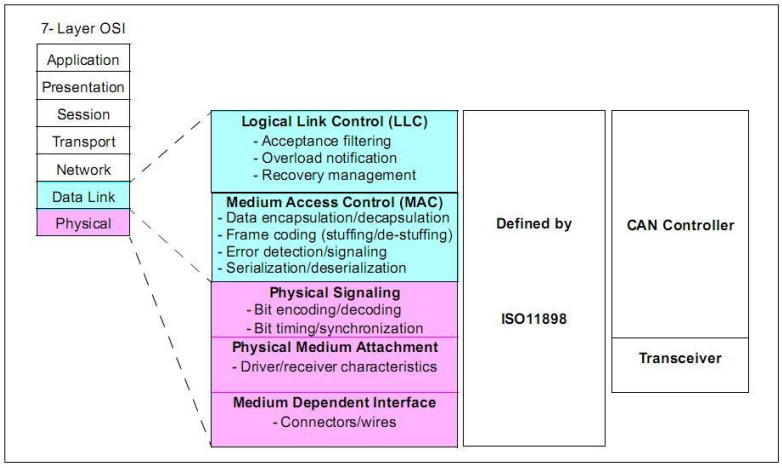


Figura C.1. Capes del protocol definides per la norma ISO-11898.

BOSCH només va definir el nivell físic i el d'enllaç de dades, com s'observa a la figura C.1. Això va donar peu a la creació de diferents protocols a partir de la base del CAN. Aquests protocols s'anomenen d'alt nivell (Higher layer protocols), i defineixen tots els nivells per estandarditzar els sistemes i facilitar la seva creació i modificació. Els principals protocols d'alt nivell es mostren a la taula C.1.

Higher Layer Protocols	**Característiques més destacables**
CANKingdom	-El node rei pren el control del bus, aquest permet o denega l'entrada al bus als altres nodes. -La presencia del rei simplifica el disseny de sistemes en temps real.
CANopen	-És un protocol no propietari, per usar-lo no cal pagar cap llicència. -És compatible amb altres protocols. -Es poden posar nous nodes al bus sense canviar els altres.
CCP/XCP	-És un protocol per la posta a punt d'un sistema CAN
Device Net	-És un protocol basat en CIP (Common Industrial Protocol). -Està especialment dissenyat per ser usat en plantes industrials. -Permet connexions a la xarxa Ethernet de la planta i a Internet.
J1939	- És una norma definida per la SAE (Society of Automobile Engineers). -S'usa en vehicles pesats, com camions i autobusos.
OSEK/VDX	-És un projecte comú de la indústria de l'automòbil. -Inclou un sistema operatiu en temps real per a cada ECU.

Taula C.1. Principals protocols d'alt nivell

C.2. Funcionament del bus CAN

C.2.1.Descripció d'una xarxa CAN

En una xarxa CAN els dispositius connectats a la xarxa s'anomenen nodes. Els nodes estan formats per un transceiver, un controlador CAN i un microcontrolador. El transceiver s'encarrega d'adequar els senyals entre el bus i el controlador CAN. El controlador CAN filtra i emmagatzema els missatges del bus i permet que el microcontrolador pugui enviar i rebre missatges. Finalment, el microcontrolador interpreta els missatges i els fa servir en el seu programa, també pot enviar nous missatges.

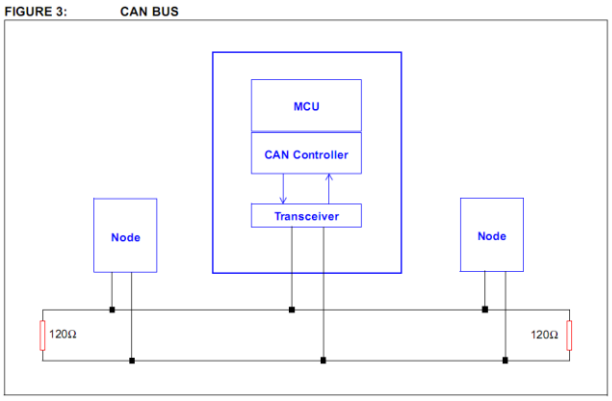


Figura C.2. Descripció de la Xarxa CAN

A la figura C.2 es pot observar que el bus consta de dos cables, el CANH i el CANL. Al final d'aquests dos cables hi ha una resistència de 120Ω a cada acabament del bus com dicta la norma 11898. Tot i que, existeixen altres acabaments possibles.

C.2.2. Nivells de Bus

El bus CAN té dos estats lògics: Recessiu i dominant. La ISO-11898 defineix un voltatge diferencial entre els dos cables del bus (CANH i CANL) per a cada estat. A la figura C.3 s'observen aquests dos estats.

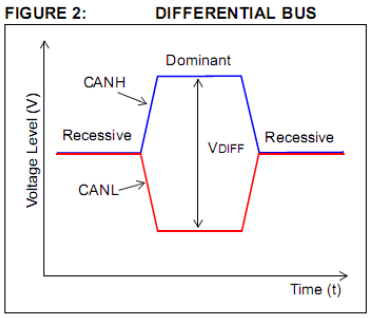


Figura C.3. Estats possibles del Bus

L'estat recessiu equival a un '1' lògic i el dominant a '0' lògic. Les diferències de tensió que defineixen un estat o altre es mostren a la figura C.4. Es pot observar a la figura que existeixen uns rangs de tensió d'entrada i sortida per interpretar cadascun dels estats.

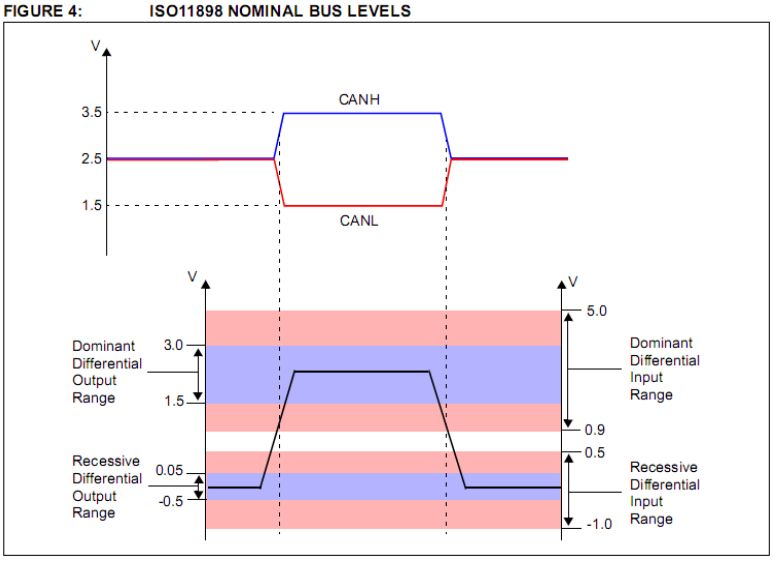


Figura C.4. Voltatges diferencials de cadascun dels estats.

Els estats dominants es sobreposen als recessius per permetre un arbitratge no destructiu en cas de que dos nodes accedeixin el bus simultàniament.

C.3. Missatges CAN

Inicialment, la norma 11898 només definia missatges de format estàndard. Però degut a el desenvolupament de sistemes més complexes, va sorgir la necessitat de crear un nou format, l'estès, que està descrit a la norma 11898-2. El nou format disposava d'un camp d'arbitratge (arbitration field) de 32 bits enlloc de 12 com en l'estàndard.

Hi ha dos tipus de missatges, els de dades i els de petició remota. Els camps que contenen els missatges de dades són els mostrats a la figura C.5, i els de petició remota a la figura C.6. Els missatges de dades serveixen per enviar informació i els de petició remota per demanar-la.

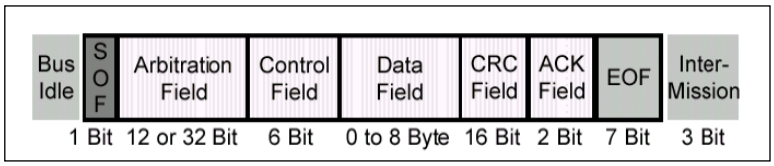


Figura C.5. Camps dels missatges dades

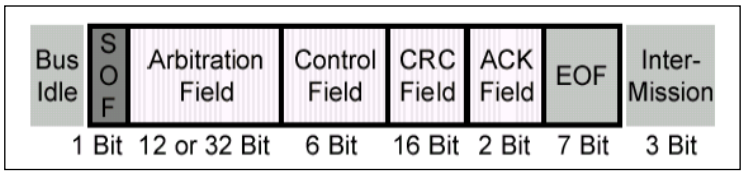


Figura C.6. Camps dels missatges de petició remota

En els apartats següents es descriuen cadascun dels camps que formen aquests dos tipus de missatges. Entre dos missatges consecutius cal que hi hagin 3 bits recessius.

C.3.1. SOF (Start of frame)

Consta d'un sol bit sempre dominant que indica l'inici del missatge. Serveix per a la sincronització amb altres nodes.

C.3.2. Camp d'arbitratge (Arbitration field).

Aquest camp és diferent segons el format del missatge. En els missatges estàndard consta dels bits mostrats a la figura C.7. I si són estesos dels bits mostrats en la figura C.8.

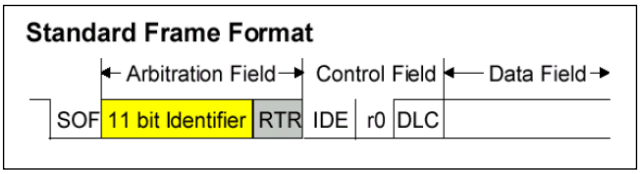


Figura C.7. Descripció del camp d'arbitratge i control dels missatges estándar

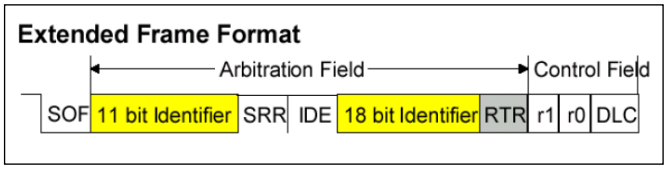


Figura C.8. Descripció del camp d'arbitratge i control dels missatges estesos

L'ús de cada part del camp s'explica a la taula C.2.

Part del Camp	**Significat**
Identifier (Identificador)	En missatges estàndard consta d' 11 bits i en estesos de 29. Serveix per saber quin és el contingut del missatge.
RTR (*Remote Transmission Request)*	Consta d'un bit. Distingeix el missatges de dades (dominant) de les peticions remotes(recessiu).
SRR(Substitute Remote Request)	Consta d'un bit. Substitueix l' RTR en missatges estesos. És sempre recessiu.
IDE(Identifier extension)	Consta d'un bit. És l' identificador del format de missatge, estàndard (dominant) o estès(recessiu).

Taula C.2. Us de les parts del camps d'arbitratge

C.3.2. Camp de control (*Control Field*)

El camp de control està format per dos bits reservats per a ús futur i quatre bits que indiquen el nombre de bytes de dades. El primer dels bits reservats s'usa en format estàndard per indicar el format de missatge (IDE). El segon bit reservat és sempre recessiu. Els quatre bits de codi de longitud (DLC) indiquen en binari el nombre de bytes de dades en el missatge (de 0 a 8). La figura C.9. mostra el contingut d'aquest camp.

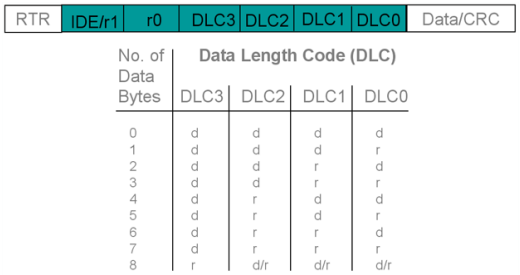


Figura C.9. Contingut del camp de control

C.3.3. Camp de dades

Està format per un nombre variable de bytes, de 0 a 8 bytes segons el DLC. Aquest camp només apareix en els missatges de dades. La figura C.10 mostra el contingut d'aquest camp.

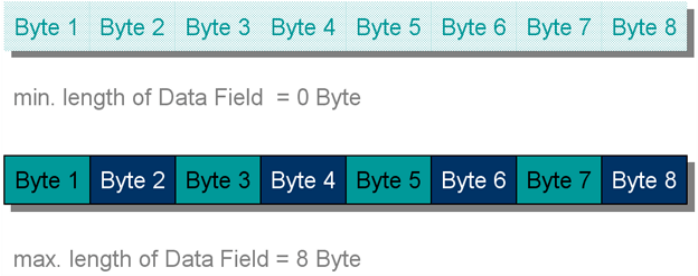


Figura C.10. Contingut del camp de dades

C.3.4.Camp CRC

Aquest camp conté 15 bits del càlcul del CRC(Cyclic redundancy check) i un bit delimitador que sempre és recessiu. El càlcul del CRC permet detectar errors aleatòriament distribuïts en SOF, *Arbitration*, *Control and Data Fields*. A més pot detectar errors de fins a 15 bits consecutius. La figura C.11 mostra el contingut d'aquest camp.

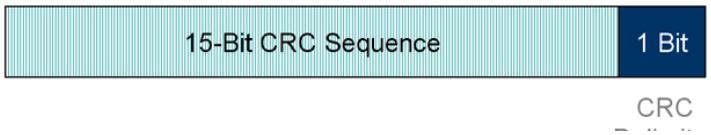


Figura C.11. Contingut del camp de CRC.

C.3.5. Camp de reconeixement (ACK)

El camp ACK consta de dos bits, el ACK Slot i el ACK Delimiter. El node que transmet el missatge deixa aquests dos bits en estat recessiu, el receptor en rebre el missatge correctament posa en estat dominant el ACK Slot per comunica-ho al transmissor. Quan el transmissor detecta que s'ha canviat l'estat del ACK Slot, sap que com a mínim un node ha rebut correctament el missatge. A la figura C.12 es pot veure el contingut d'aquest camp.

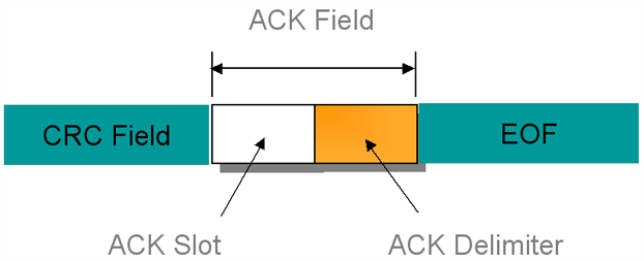


Figura C.12. Contingut del camp ACK.

C.3.6 EOF(End of frame)

Consta de 7 bits recessius que indiquen la fi del missatge, tan per missatges de dades com de petició remota. El contingut d'aquest camp es mostra a la figura C.13.

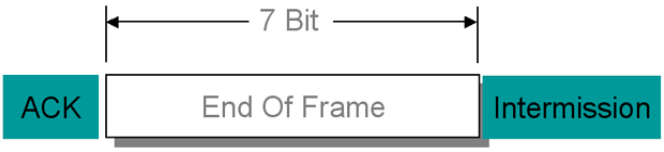


Figura C.13. Contingut del camp EOF

C.4.Trama d'errors (Error frame)

Les trames d'error són generades per qualsevol node que detecta un error. Consisteix en dos camps: Indicador d'error *(Error Flag)* i el Delimitador d'error. El Delimitador d'error consta de 8 bits recessius consecutius i permet als nodes reiniciar la comunicació netament després de l'error. L'indicador d'error és diferent segons l'estat d'error del node que detecta l'error. El node que detecta primer l'error està en estat d'error actiu i la resta en estat d'error passiu.

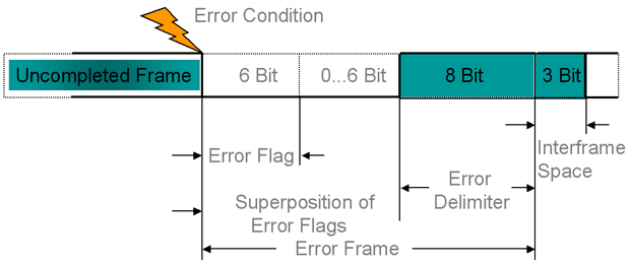


Figura C.14. Trama d'error

Si un node en estat d'error "Actiu" detecta un error al bus interromp la comunicació del missatge en procés, generant un "Indicador d'error actiu" que consisteix en una seqüència de 6 bits dominants successius. Aquesta seqüència provoca la generació de trames d'error en altres nodes. Per tant, l'indicador d'error pot estendre's entre 6 i 12 bits dominants successius. Finalment, s'espera el camp de delimitació d'error format pels 8 bits recessius. Llavors, la comunicació es reinicia i el node que havia estat interromput reintenta la transmissió del missatge.

Si un node en estat d'error "Passiu" detecta un error, el node transmet un "Indicador d'error passiu" seguit, de nou, pel camp delimitador d'error. L'indicador d'error de tipus passiu consisteix en 6 bits recessius seguits i, per tant, la trama d'error per a un node passiu és una seqüència de 14 bits recessius. D'aquí es dedueix que la transmissió d'una trama d'error de tipus passiu no afectarà cap node a la xarxa, excepte quan l'error és detectat pel propi node que està transmetent.

Un altre mètode per a la detecció d'errors és la revisió del camp CRC. Es calcula amb un polinomi generador d'igual manera en el receptor i a l'emissor. Això permet detectar errors aleatoris en 5 bits o una seqüència seguida de 15 bits corruptes. També es considera error si el camp ACK no es escrit com a dominant pel receptor.

C.5 Arbitratge CAN

El protocol CAN permet l'accés al bus a més d'un node alhora. El procés d'arbitratge està basat en el CSMA/CD *(Carrier Sense Multiple Access with Collision Detection).* Cada node ha d'esperar un període sense activitat abans d'enviar un missatge *(Carrier Sense).* Un cop transcorregut aquest temps tots els nodes tenen la mateixa oportunitat d'enviar un missatge *(Multiple Access).* En el cas que dos nodes comencin a transmetre al mateix temps, hi haurà una col·lisió.

Quan es succeeix una col·lisió és necessari l'arbitratge. Cada node emissor envia els bits de l'identificador del missatge i comprova l'estat del bus. El node amb més bits dominants a l'identificador accedirà primer al bus, perquè prevalen als recessius. Un cop transmès el missatge del node guanyador de l'arbitratge, el perdedor intenta de nou enviar el seu missatge. És un arbitratge no destructiu, ja que només comporta una espera pels missatges perdedors. A la figura C.15 es mostra un procés d'arbitratge, on surt guanyador el node 2.

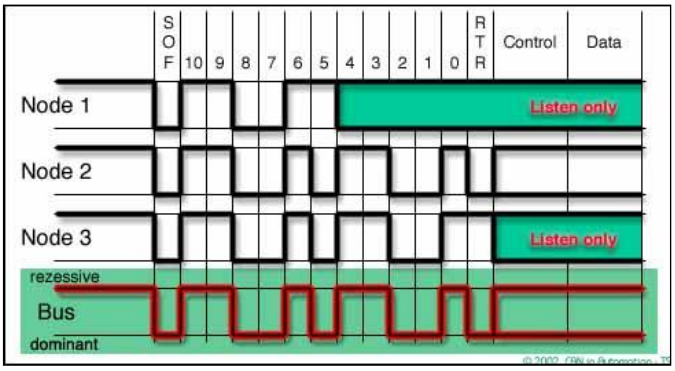


Figura C.15. Arbitratge CAN.

D. Placa d' electrònica de l'automòbil

D.1. Esquemes

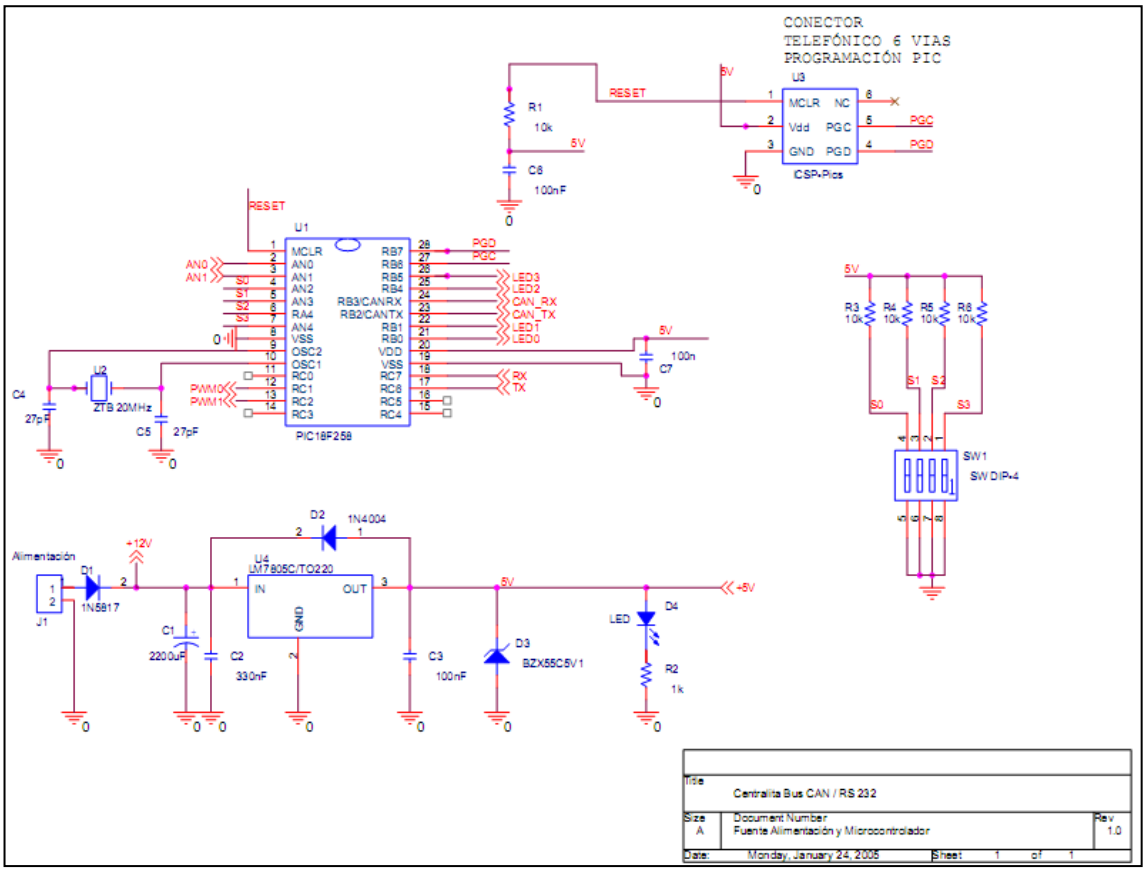


Figura D.1. Esquema de l'alimentació i Microcontrolador de la placa

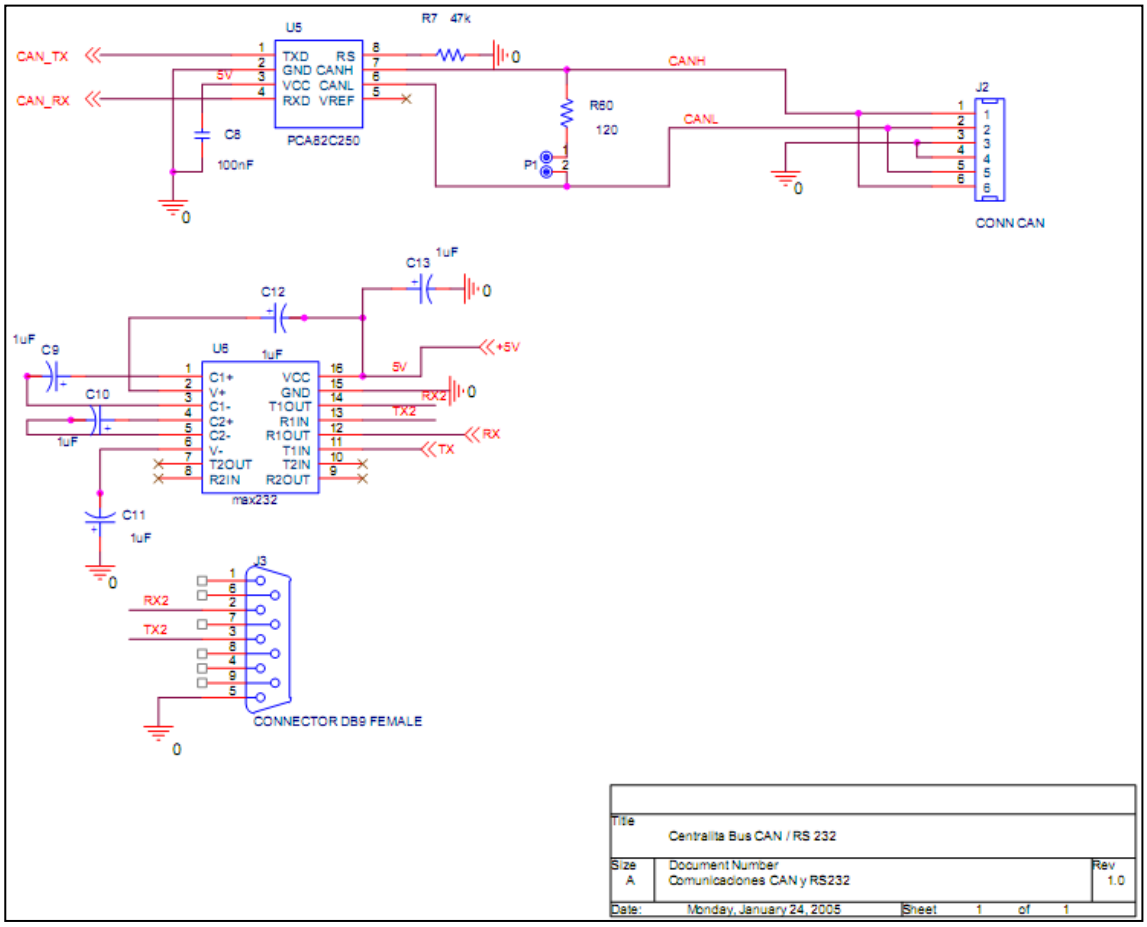


Figura D.2. Esquema de les comunicacions CAN i RS232 de la placa.

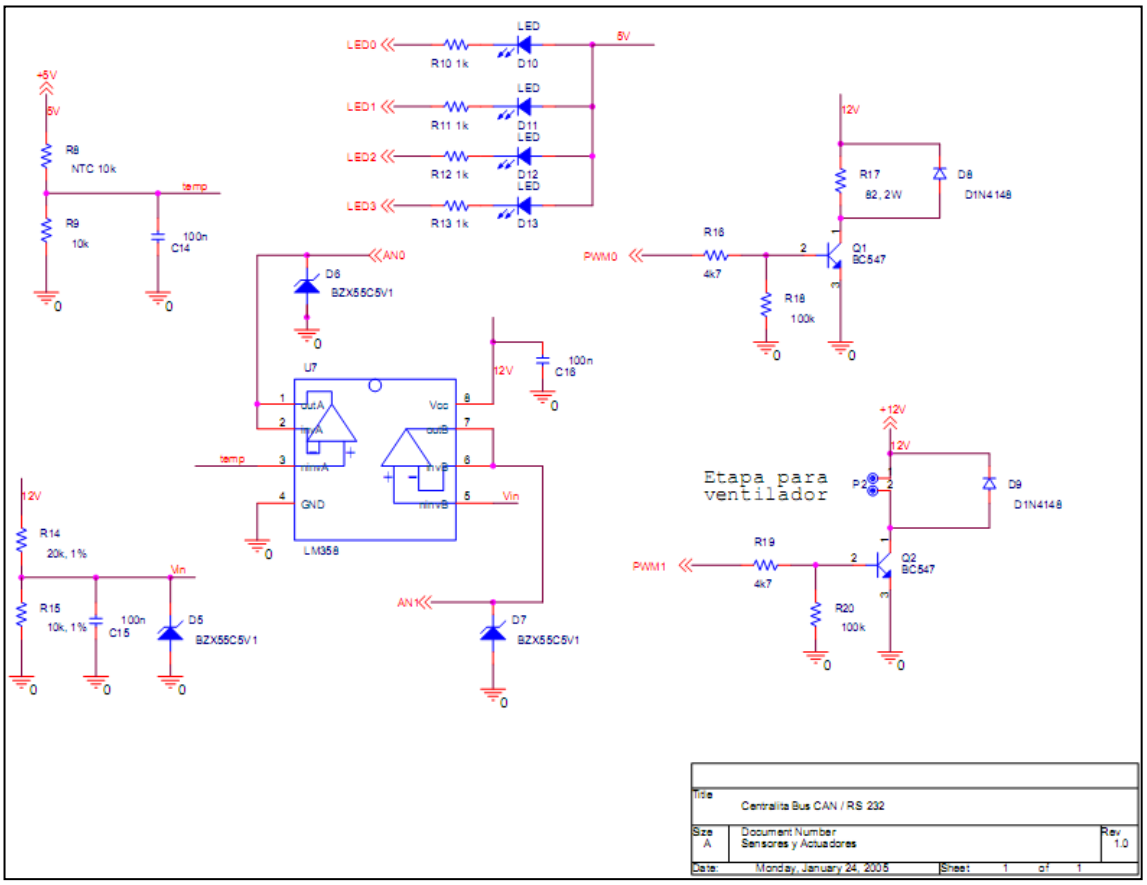


Figura D.3. Esquema dels sensors i actuadors de la placa.

D.2 Imatges

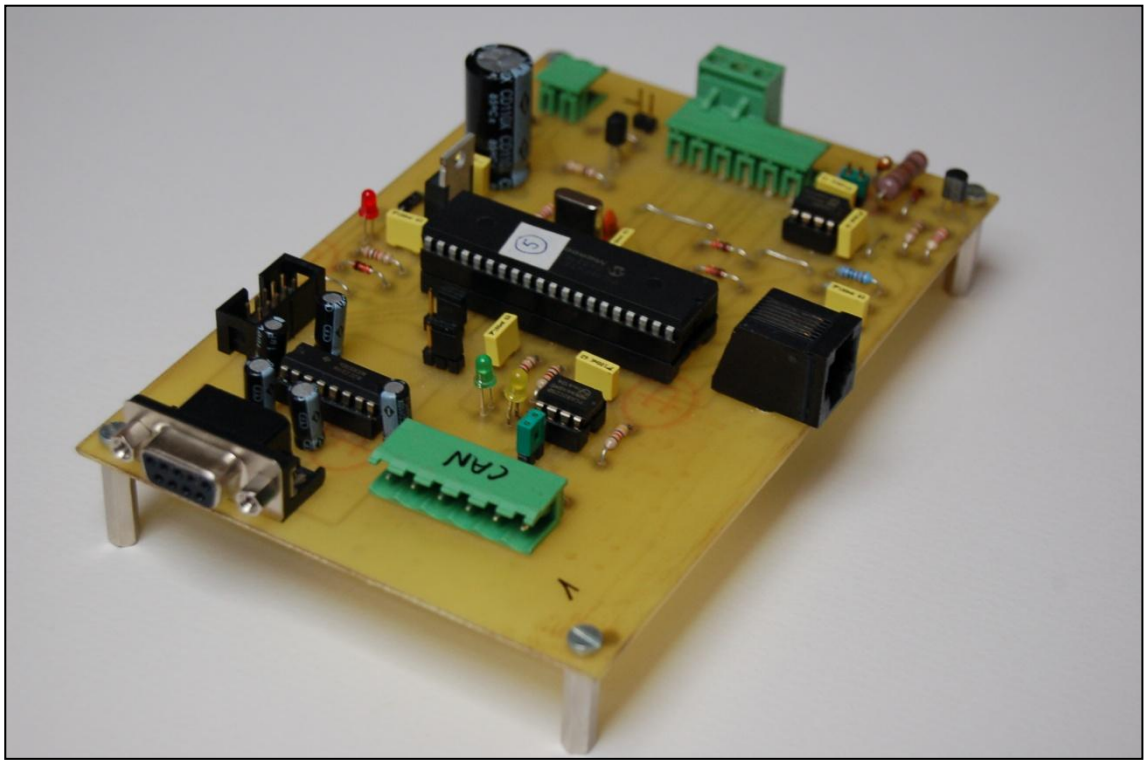


Figura D.4. Imatge de la placa

D.3. Característiques dels microcontroladors PIC18FXX8

- *L'arquitectura del processador segueix el model Harvard.* En aquesta arquitectura es disposa de dues memòries completament independents, una és la d'instruccions de 16 bits d'amplada i l'altra la de dades de 8 bits d'amplada. La CPU (Unitat Central de Processament) es connecta de manera independent a cadascuna de les memòries mitjançant dos busos diferents, i pot accedir de manera simultània a ambdues memòries.
- *S'usen "Pipelines" en l'execució de les instruccions*. Aquesta tècnica permet al processador realitzar alhora l'execució d'una instrucció i la cerca del codi de la següent (fetch), permetent així que es pugui executar cada instrucció en un cicle (4 cicles de rellotge), excepte les de salt que s'executen en 2 cicles(8 cicles de rellotge).
- *Totes les instruccions del PIC18FXX8 són de 16 bits.* Aquest fet afavoreix la optimització de la memòria d'instruccions i facilita la construcció d'assembladors i compiladors.

- *Processador RISC (Reduced Instructions Set Computer).* El repertori d'instruccions d'aquests microcontroladors és reduït, de 75 instruccions. Així les instruccions són simples i ràpides quan s'executen. Això permet optimitzar el hardware i software del processador.
- *Arquitectura basada en un "banc de registres".* Tots els objectes del sistema (ports d'entrada/sortida, temporitzadors, posicions de memòria, etc.) estan implementats físicament com a registres.
- *Eines de suport potents i econòmiques. Microchip*, l'empresa que fabrica els PIC18FXX8 posa a disposició dels usuaris nombroses eines per a desenvolupar hardware i software.

D.3.1. Configuration bits

La descripció de les funcions dels Configuration bits estan a la pàgina 267 del full de característiques del PIC18FXX8[2].

D.3.2.Registres CAN

Per consultar quins són els registres CAN de microcontrolador consultar el capítol 19 del full de característiques del PIC18FXX8[2].

E. *Kvaser* Leaf Ligth HS

E.1. Característiques tècniques

A la taula E.1 es mostren les característiques tècniques de la sèrie *Kvaser* Leaf.

Property	Description
CAN channels	1 (CAN 2.0A and 2.0B active)
Error Frame Detection	Yes
Hardware requirements	IBM PC AT or 100% compatible; USB host socket
USB interface	USB 2.0 or USB 1.1
Power consumption	5V and approx. 70mA powered from the USB side
Software requirements[1]	Windows 98, ME, 2000, XP, Server 2003
Configuration	Done by software (Plug & Play)
Dimensions (W*L*H)	25 mm * 100 mm * 20 mm
Housing	Black plastic
CAN Connector	9-pin male DSUB
USB cable length	110 cm (approx. 3.6 ft.)
USB connector	USB standard type “A” plug
CAN cable length	30 cm (1 ft.)
Load dump protection	Yes
Mix 11 / 29 bits messages	Yes

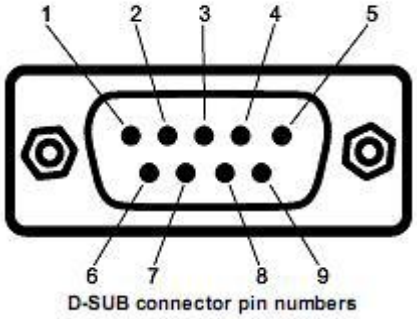
Taula E.1. Característiques tècniques de la sèrie *Kvaser* Leaf.

La Taula E.2. mostra les característiques tècniques concretes pel *Kvaser* Leaf Ligth HS.

Property	Description	Unit
CAN physical layer	High speed (ISO 11898-2)	
CAN transceiver	Texas SN65HVD251	
USB speed	12	Mbit/s
Bit rate	5-1000	Kbit/s
Temperature range	-0 - +70	°Celsius
Clock accuracy	100	µs
Max message rate	8000	Messages/s
Time stamp	32	Bits

Taula E.2. Característiques tècniques del *Kvaser* Leaf Ligth HS

E.2. Connexions CAN



D-SUB Pin number	High Speed
1	Not connected
2	CAN_L
3	GND
4	Not connected
5	Shield
6	Not connected
7	CAN_H
8	Not connected
9	Not connected

Figura E.1. Esquema de les connexions del *Kvaser*

E.3. Imatges

Figura E.2. Imatge del *Kvaser* Leaf Ligth HS.

E.4. Llibreria CANLIB

Per saber les funcions de la llibreria consultar el fitxer d'ajuda *"CANLIB SDK HELP"*, que es troba al directori Bibliografia del CD adjunt.

F. Microsoft Visual Basic 6.0

F.1. Microsoft Visual Basic 6.0

L'entorn de desenvolupament integrat *Microsoft Visual Basic 6.0* utilitza el llenguatge de programació *Visual Basic*. Gran nombre de tasques de programació es realitzen sense escriure codi, només realitzant operacions gràfiques. Les aplicacions creades en *Visual Basic* estan basades en objectes i són manejades per esdeveniments. Mentre que el codi que s'executa quan succeeix un esdeveniment s'escriu en llenguatge Basic.

Microsoft Visual Basic 6.0 està dissenyat per a la realització de programes per Windows, podent incorporar els elements gràfics d'aquest entorn. Cadascun dels elements gràfics és un tipus de control: els botons, les caixes de diàleg i de text, les caixes de selecció desplegables, els botons d'opció i de selecció, les barres de desplaçament horitzontals i verticals, els gràfics, els menús, i molts altres tipus d'elements. Cada control ha de tenir un nom per poder fer referència a ell dins el codi del programa.

S'anomena formulari (form) a una finestra de Windows. Un formulari és un contenidor de controls. Una aplicació pot tenir diversos formularis, però un únic formulari és suficient per a crear una aplicació. Els formularis també tenen un nom per poder-ne fer referència.

Els formularis i els controls tenen un conjunt de propietats que defineixen el seu aspecte gràfic (grandària, color, posició en la finestra, tipus i grandària de lletra, etc.) i la resposta que donen a les accions de l'usuari (si està actiu o no, per exemple). Generalment, les propietats d'un control o objecte es configuren amb valors lògics (True, False) o numèrics. Quasi totes les propietats dels objectes poden establir-se en temps de disseny o d'execució.

Els mètodes són funcions que són cridades des del programa, però a diferència de les propietats no són configurables per l'usuari, sinó que vénen ja preprogramades amb el llenguatge. Per exemple, per a un formulari hi ha el mètode Hide que fa que el formulari s'oculti. D'aquesta manera s'allibera al programador d'escriure tot aquest codi. Cada tipus d'objecte té mètodes diferents.

F.2. Creació d'una aplicació

El procés de creació d'una aplicació en *Visual Basic* té els següents passos:

- *Crear la interfície gràfica.* Es crea un formulari amb tots els objectes necessaris per a l'aplicació.
- *Definir les propietats dels objectes*. Poden establir-se valors inicials des de l'entorn de disseny o bé configurar-los en el codi del programa.
- *Escriure el codi dels esdeveniments.* Es defineix quina serà la resposta del programa segons els esdeveniments provocats per l'usuari en els objectes.
- *Provar i depurar l'aplicació.* Un cop escrit el codi dels esdeveniments es pot provar el seu funcionament amb la funció *Start.*
- *Crear un executable.* Un cop verificat el funcionament cal generar un programa amb extensió *.exe. Si el codi està lliure d'errors es pot crear amb la funció Make del menú File.
- *Crear una aplicació d'instal·lació.* Els programes de *Visual Basic* només poden funcionar si s'instal·len unes llibreries (*.dll) i complements (*.ocx) concrets, a no ser de que es disposi de *Microsoft Visual Basic* instal·lat.

G. Codis font dels *Bootloaders*

G.1. Codi font del *Bootloader* modificat

El codi font del *Bootloader* modificat es pot consultar obrint l'arxiu de projecte de l'MPLAB *Bootloader*_Modificat.mcp, situat al directori *Software projecte**codi Bootloader* del CD adjunt. Per obrir-lo es requereix la instal·lació del programa MPLAB. Per descarregar-lo de la web de *Microchip* consulteu l'annex L.

S'ha procurat que el codi sigui el més compacte possible, ja que es disposa d'un espai de memòria de *Boot* limitat. S'ha desestimat usar identificadors del missatge d'inici diferents als del *Bootloader* degut a aquesta limitació d'espai.

G.2. Bootloader de Microchip

El codi font del *Bootloader* de *Microchip* es pot trobar a l' APPENDIX B del document *AN247. pdf* , el qual està situat a la carpeta *Bibliografia* del CD adjunt. Per entendre més fàcilment el codi font del *Bootloader* de *Microchip*, observeu les figures G.1 i G.2 de les pàgines següents. Són diagrames de flux del funcionament del *Bootloader* en mode *PUT i PUTGET* respectivament.

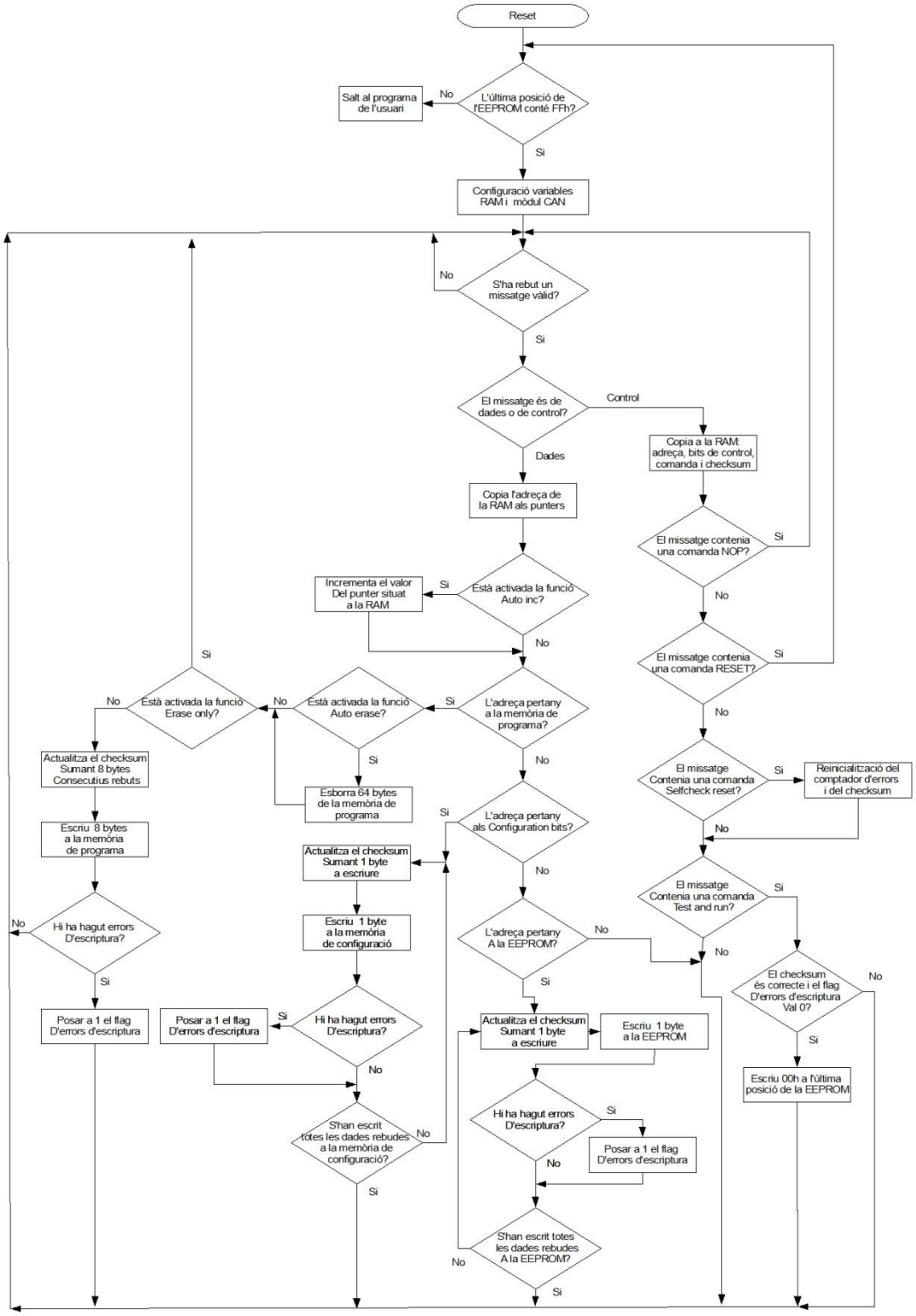


Figura G.1. Diagrama de flux del funcionament del *Bootloader* en mode *PUT*

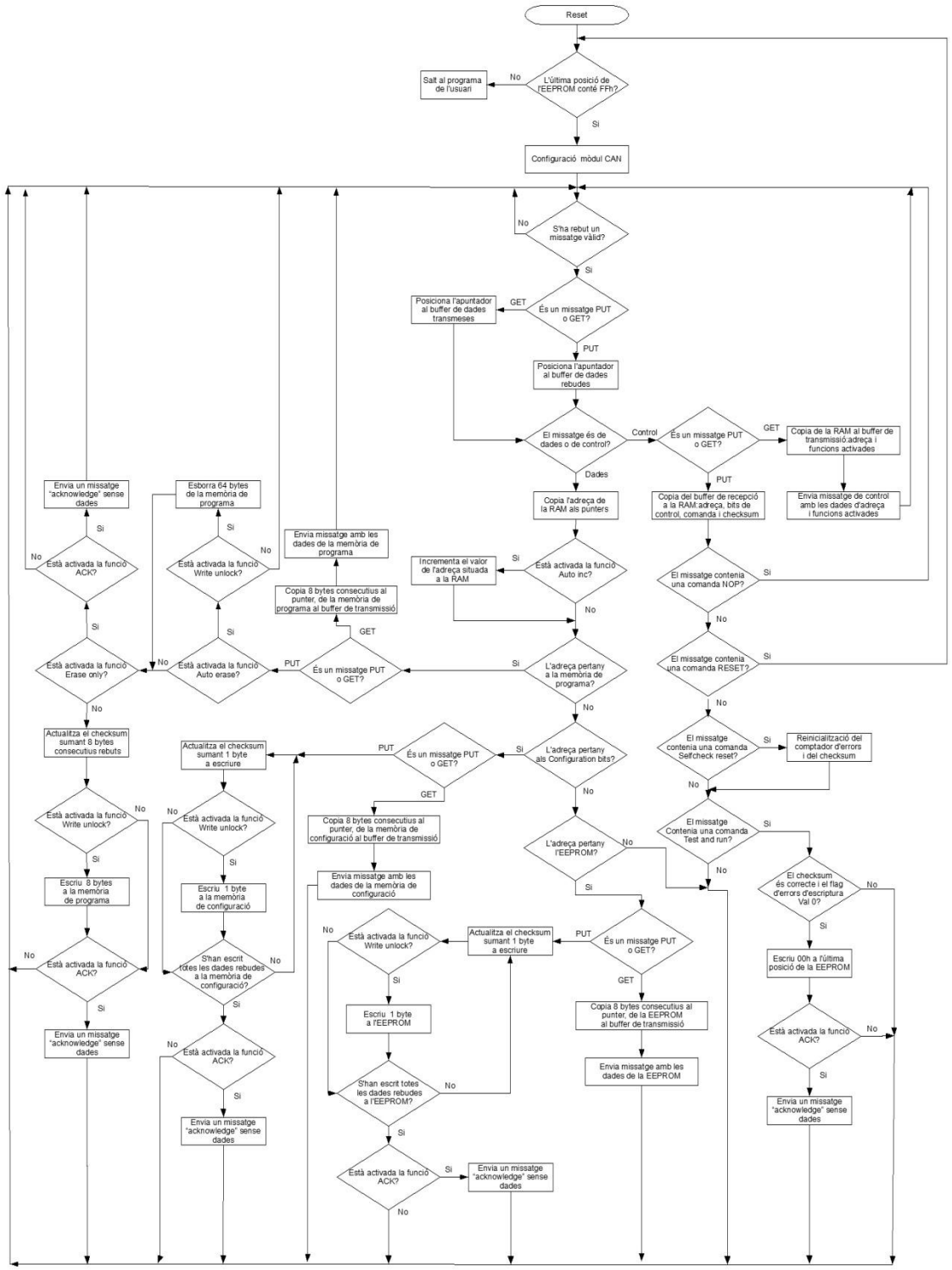


Figura G.2. Diagrama de flux del funcionament del *Bootloader* en mode *PUTGET*

H. Codi programa definicions.

El codi font del programa de definicions es pot consultar obrint el fitxer de projecte de *Visual Basic, Proyecto 1.vbp*, situat al directori *\Software projecte\codi programa definicions* del CD adjunt. Per obrir-lo es requereix la instal·lació del programa *Microsoft Visual Basic 6.0*.

I. Codi programador CAN.

El codi font del programa de definicions es pot consultar obrint el fitxer de projecte de *Visual Basic, project1.vbp*, situat al directori *\Software projecte\codi programador CAN* del CD adjunt. Per obrir-lo es requereix la instal·lació del programa *Microsoft Visual Basic 6.0*.

S'ha procurat seguir un disseny estructurat alhora de programar, dividint les operacions a realitzar en altres de més petites per donar modularitat al codi i facilitar les tasques de depuració. Un cop obert el projecte es poden veure els diferents mòduls que li pertanyen, en cadascun d'ells s'agrupen les funcions que fan tasques similars. Els noms de les funcions i els comentaris permeten fer canvis al programa amb facilitat.

J. Codi programes de test

J.1. Programa Leds

El codi font del programa de test que encén els leds es pot consultar obrint l'arxiu de projecte de l'MPLAB, *test.mcp*, situat al directori *\Programes test\Programa leds* del CD adjunt. Per obrir-lo es requereix la instal·lació del programa MPLAB. Per descarregar-lo de la web de *Microchip* consulteu l'annex L.

J.2. Programa missatges CAN

El codi font del programa de test que envia i rep missatges CAN es pot consultar obrint l'arxiu de projecte de l'MPLAB, *S5.mcp*, situat al directori *\Programes test\Programa CAN* del CD adjunt. Per obrir-lo es requereix la instal·lació del programa MPLAB. Per descarregar-lo de la web de *Microchip* consulteu l'annex L.

J.3. Programa interrupcions

El codi font del programa de test de les interrupcions es pot consultar obrint l'arxiu de projecte de l'MPLAB *test.mcp*, situat al directori *\Programes test\Programa interrupcions* del CD adjunt. Per obrir-lo es requereix la instal·lació del programa MPLAB. Per descarregar-lo de la web de *Microchip* consulteu l'annex L.

K. Codificació dels fitxers *.hex

El format de fitxer hex emmagatzema informació binària útil per programar microcontroladors i EEPROM's. És un fitxer de text en què cada línia conté valors hexadecimals que representen l'adreça d'inici o relativa i la seqüència de dades a escriure a partir d'ella.

Cada línia del fitxer hex consta de sis parts:

1. Codi d'inici: Consta d'un caràcter ASCII, els dos punts":".
2. El comptador de bytes: són dues xifres hexadecimals que representen el nombre de bytes de dades que conté la línia. Són de 16 bytes, 10h en hexadecimal.
3. Adreça: Mostra l'adreça d'inici o relativa de les dades. Consta de 2 bytes, 16 bits. Es pot ampliar la seva longitud usant un tipus de seqüència concret.
4. Tipus de seqüència (Record type): N'hi ha sis de diferents i serveixen per interpretar les dades. Consten d'un byte.
    1. 00h: Conté dades i l'adreça és de 16 bits.
    2. 01h: Fi de fitxer, no conté dades. És la última fila del fitxer.
    3. 02h: Es fa servir per definir adreces de 24 bits. El byte següent representa el byte més significatiu de l'adreça de 24 bits.
    4. 03h:No usat en els processadors PIC18.
    5. 04h: Es fa servir per definir adreces de 32 bits. Els dos bytes següents representen els dos bytes més significatius de l'adreça de 32 bits.
    6. 05h:No usat en els PIC18.
5. Dades: Seqüència de bytes de dades.
6. Checksum: 1 byte. És el byte menys significatiu del complement a dos de la suma de tots els bytes de la línia menys el del Checksum.

Exemple: Fitxer hexadecimal per programar l' EEPROM del microcontrolador de la posició 00h a la 1Fh de memòria amb tots els bytes iguals a 0.

Fitxer:

:0200000400F00A

:10000000000000000000000000000000000000F0

:10001000000000000000000000000000000000E0

:00000001FF

Significat línies:

1. Carrega l'adreça d'inici 00F00000h
2. Comença a gravar les dades a 00F00000h i acaba a 00F0000Fh.
3. Comença a gravar les dades a 00F00010h i acaba a 00F0001Fh.
4. Fi de fitxer

Checksum de la primera línia: 02h+04h+F0h=F6h => 100h-F6h=0Ah

L. Manual d'usuari

L.1. Instal·lació

Per facilitar la instal·lació del programa s'ha creat un programa d'instal·lació, està al directori *\Software projecte\instal·lador* del CD adjunt. Aquest programa instal·la els arxius i llibreries necessaris perquè funcionin els programes de *Visual Basic*. Aquest instal·lador ha estat creat amb un programa gratuït anomenat *Inno Setup*. Per més informació es pot consultar la seva web http://www.innosetup.com/isinfo.php.

A part dels programes creats en aquest projecte, també és necessari instal·lar el següents:

1. El kit de desenvolupament de software de *Kvaser*, el *CANlib*. Es pot descarregar gratuïtament de: http://www.*Kvaser*.com/cgi-bin/sendfile.cgi?dir=files&file=canlib.txt
2. El driver del Kvaser. Es pot descarregar de: http://www.kvaser.com/cgi-bin/sendfile.cgi?dir=files&file=kvaser_drivers_w2k_xp.txt
3. El programa MPLAB de *Microchip*. Es pot descarregar gratuïtament de: http://ww1.*Microchip*.com/downloads/en/DeviceDoc/MPLAB_8.30.zip
4. Per monotoritzar el bus es recomana l'eina *CANKing* de *Kvaser*. És gratuïta i es pot descarregar de: http://www.*Kvaser*.com/cgi-bin/sendfile.cgi?dir=beta&file=*Kvaser*_canking_setup.txt

També és recomanable usar el programa *Microchip* CAN Bit Timing Calculator, per configurar correctament els paràmetres del CAN pel PIC18. Es pot descarregar gratuïtament de: http://intrepidcs.com/modules/CmodsDownload/upload/Software/MBTime.zip

L.2. Utilització del software

A continuació es descriu el procediment que permetrà a l'usuari programar una centraleta amb el microcontrolador PIC18FXX8 a través del bus CAN. Els passos a realitzar són els següents:

1. Escollir els paràmetres del *Bootloader* i generar els fitxers CANBOOT.def i BOOTDATA.txt amb el programa de definicions.
2. Compilar el *Bootloader* amb l'MPLAB fent servir el fitxer CANBOOT.def.
3. Gravar amb un programador de *Microchip* el *Bootloader* compilat anteriorment al microcontrolador.
4. Muntar la xarxa CAN, connectant la targeta *Kvaser leaf light HS* al bus.
5. Obrir el fitxer BOOTDATA.txt i entrar en mode boot amb el programador CAN.

6. Obrir amb el programador un fitxer HEX amb el programa d'usuari desitjat.
7. Esborrar i Programar el microcontrolador. Llegir si és possible.
8. Sortir del mode *boot.*

Totes aquestes operacions s'hauran de repetir per cada centraleta de la xarxa.

L.2.1.Escollir els paràmetres del *Bootloader*

Per escollir les dades del *Bootloader* entrarem a l'aplicació Automatic tool for CAN *Bootloader*. Entrarem totes les dades desitjades i premerem el botó "*Generate files*". Com s'indica a la figura L.1.

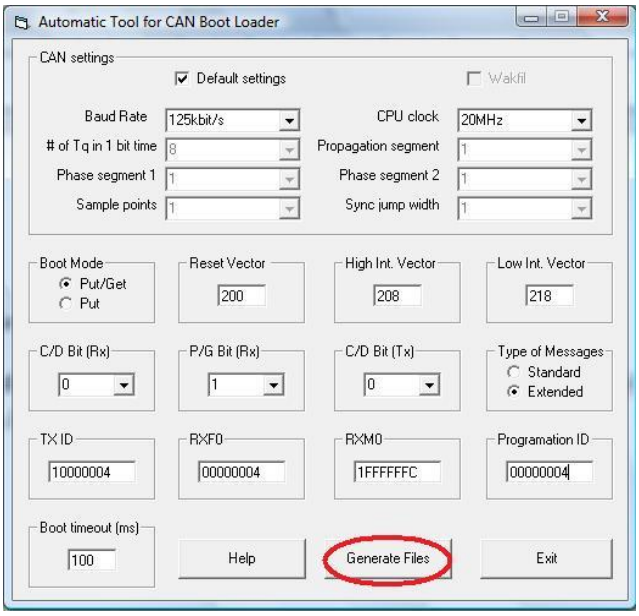


Figura L.1. Finestra del programa de definicions amb les dades desitjades.

Quan haguem premut aquest botó, si hem seleccionat els paràmetres per defecte ens apareixerà el un avís que es mostra a la figura L.2.

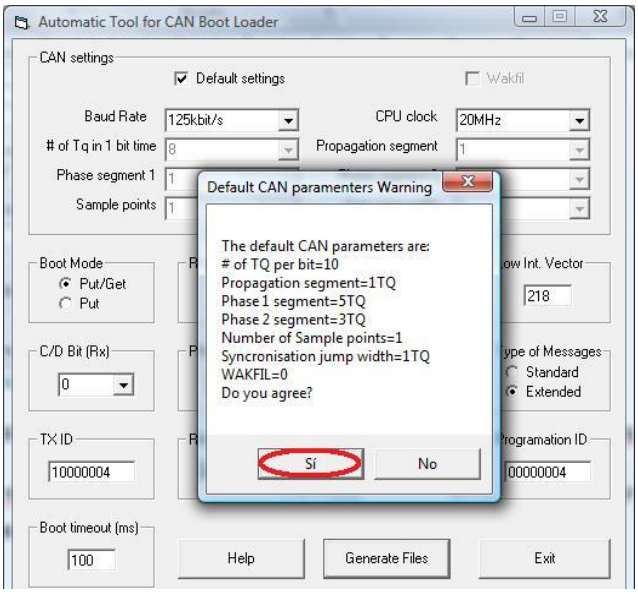


Figura L.2. Avís amb informació dels paràmetres per defecte

Per continuar clicarem Si. I ens apareixerà un quadre de diàleg per escollir la carpeta on guardar els arxius. Primer es guardarà el fitxer CANBOOT.def i després el BOOTDATA.txt. Com es mostra a la figura L.3.

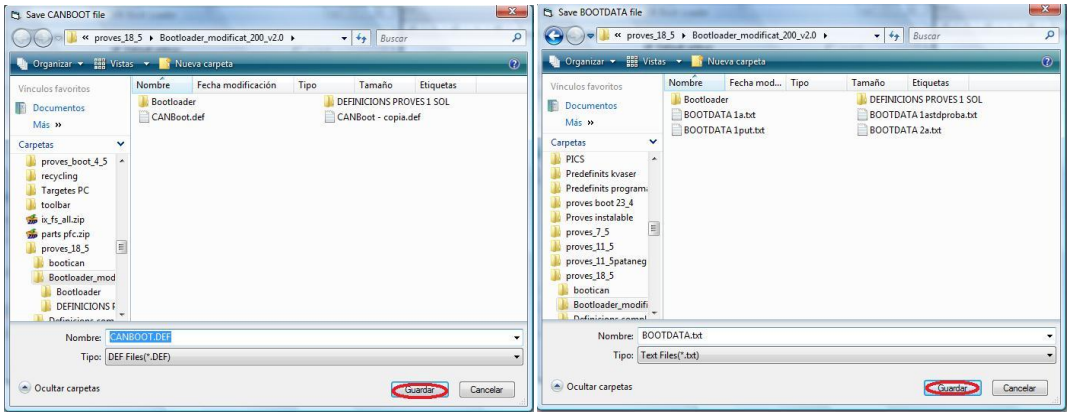


Figura L.3. Quadres de diàleg per guardar els arxius

Finalment, apareixerà un missatge, figura L.4, avisant-nos de que els fitxers s'han creat correctament. Premerem "Aceptar" per tornar a la pantalla principal.

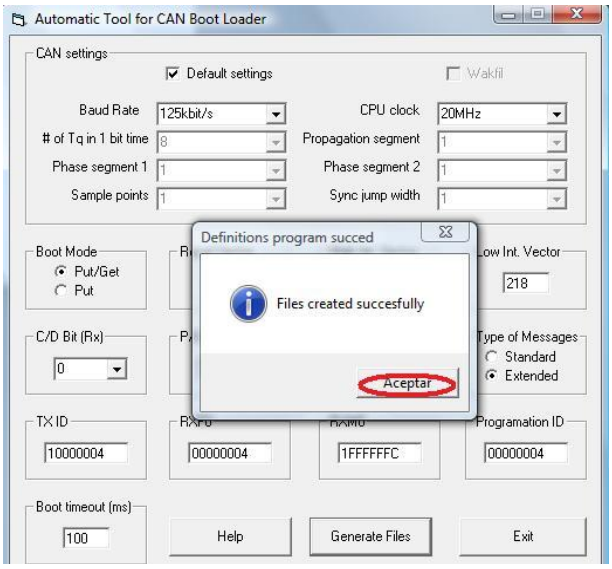


Figura L.4. Avís de la creació correcte dels arxius.

Per sortir del programa premerem “Exit”.

L.2.2. Compilar el *Bootloader* amb l’MPLAB

Obrirem amb l’MPLAB el fitxer de projecte *Bootloader*_modificat.mcp que està a la carpeta *Bootloder* de la carpeta d’instal·lació del programa. Afegirem als fitxers del projecte el fitxer CANBOOT.def dins la carpeta “Other files”, com es mostra en la figura L.5.

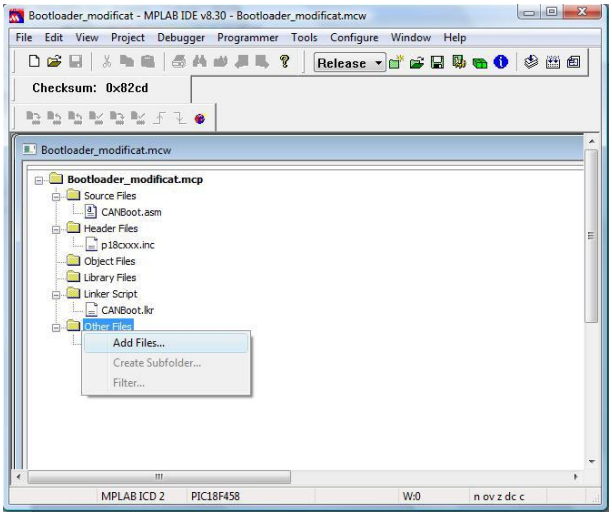


Figura L.5. Afegir el fitxer CANBOOT.def.

Seguidament ja podem compilar el *Bootloader*, amb la funció *Build all* del menú *Project*, com en la figura L.6.

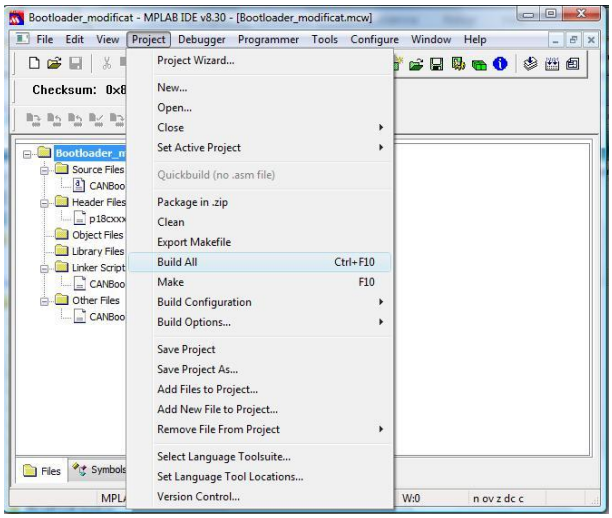


Figura L.6. Compilació del *Bootloader*

L.2.3. Gravar amb un programador de *Microchip* el *Bootloader* compilat anteriorment al microcontrolador.

Ara el *Bootloader* ja està compilat. Per programar-lo ho farem amb algun programador de *Microchip*. Per seleccionar el programador l'escollirem amb la funció *Select programmer* del menú *Programmer*. En aquest cas seleccionarem el model ICD2, com s'observa en la figura L.7.

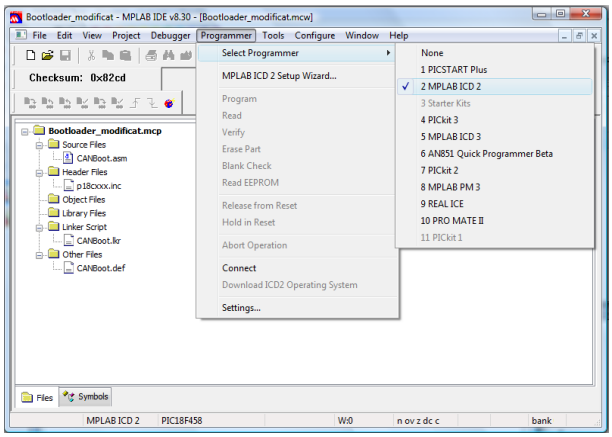


Figura L.7. Selecció del programador

Seguidament cal gravar el *Bootloader* al microcontrolador, ho farem amb la funció *Program* del menú *Programmer. Es pot veure l'operació a la figura L.8.*

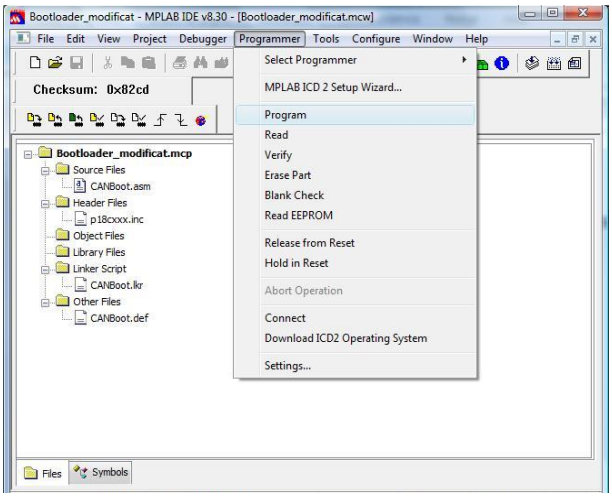
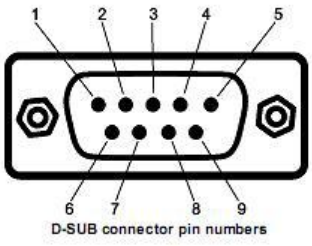


Figura L.8. Programació del *Bootloader* al microcontrolador .

L.2.4. Muntar la xarxa CAN, connectant la targeta *Kvaser* leaf light HS al bus.

Un cop gravat el *Bootloader* al microcontrolador ja es pot connectar a la xarxa CAN. Connectarem la targeta de PC *Kvaser* a la xarxa CAN i a l'ordinador. Els pins del connector de la xarxa CAN és mostren a la figura L.9.



D-SUB Pin number	High Speed
1	Not connected
2	CAN_L
3	GND
4	Not connected
5	Shield
6	Not connected
7	CAN_H
8	Not connected
9	Not connected

Figura L.9. Connector del *Kvaser*.

L.2.5. Obrir el fitxer BOOTDATA.txt i entrar en mode boot amb el programador CAN.

Primer obrim el programa *PIC18FXX8 CAN Programmer*. Un cop obert el programa seleccionarem els paràmetres de *Kvaser* desitjats. En aquest cas farem servir les opcions per defecte i el canal 0 que correspon a la targeta que volem fer servir. Seguidament premerem el botó *Boot* de la barra d'eines per obrir el un fitxer BOOTDATA.txt. Un cop obert el fitxer es visualitzaran les dades del fitxer BOOTDATA a la finestra principal del programa. Com es pot observar en la figura L.10.

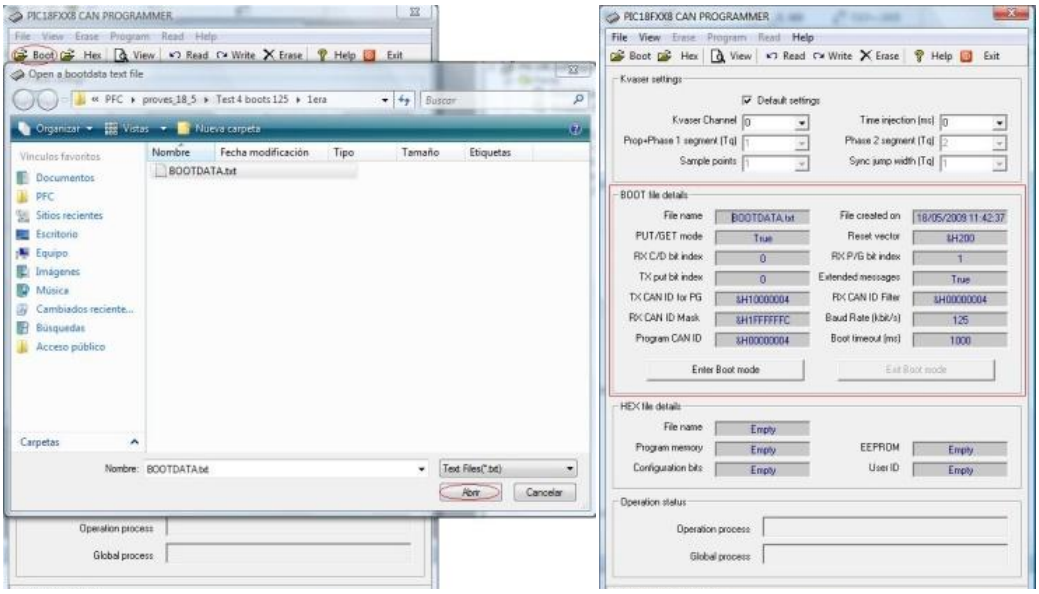


Figura L.10. Obrir fitxer BOOTDATA.

Un cop carregat el fitxer *bootdata,* podem entrar en mode *boot.* Per fer-ho polsarem el botó "*Enter boot mode*". Seguidament apareixerà un missatge dient que es faci un *Reset* a la centraleta i es premi OK. A partir d'aquest moment disposem de 30 segons per fer un *Reset* al microcontrolador. El missatge d'avís es mostra a la figura L.11.

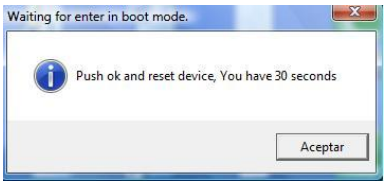


Figura L.11. Finestra d'informació per entrar en mode *boot.*

Al prémer "*Aceptar"* ens apareixerà un compte enrere a la barra d'estat. Si s'entra en mode *boot* apareixerà el missatge d'informació de la figura L.12.

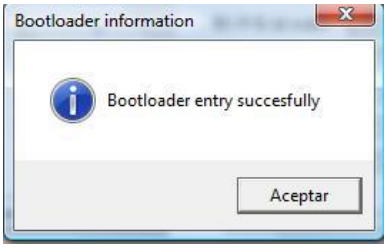


Figura L.12. Finestra que informa que s'ha entrat correctament al mode *boot.*

L.2.6. Exportar un fitxer HEX i obrir-lo amb el programador.

Un cop haguem compilat el nostre programa d'usuari cal exportar-lo a un fitxer HEX amb l'MPLAB. Per fer-ho, un cop compilat el nostre programa, seleccionarem l'opció *Export* del menú *File*. Seguidament, seleccionarem les regions de memòria a exportar i polsarem *"Aceptar".* Ho guardarem en format de fitxer INHX32, que és el format per defecte. La figura L.13 mostra la seqüència d'operacions a seguir.

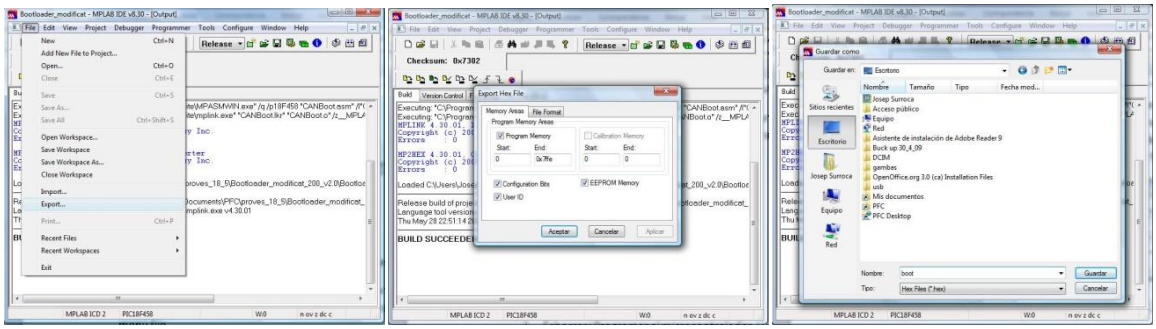


Figura L.13. Exportar a fitxer hex des de l'MPLAB

Per obrir amb el programador el fitxer hex premerem el botó *HEX* de la barra d'eines del programador i seleccionarem l'arxiu que acabem de crear. Ens apareixeran les dades en el requadre de detalls del fitxer HEX, com es mostra en la figura L.14.

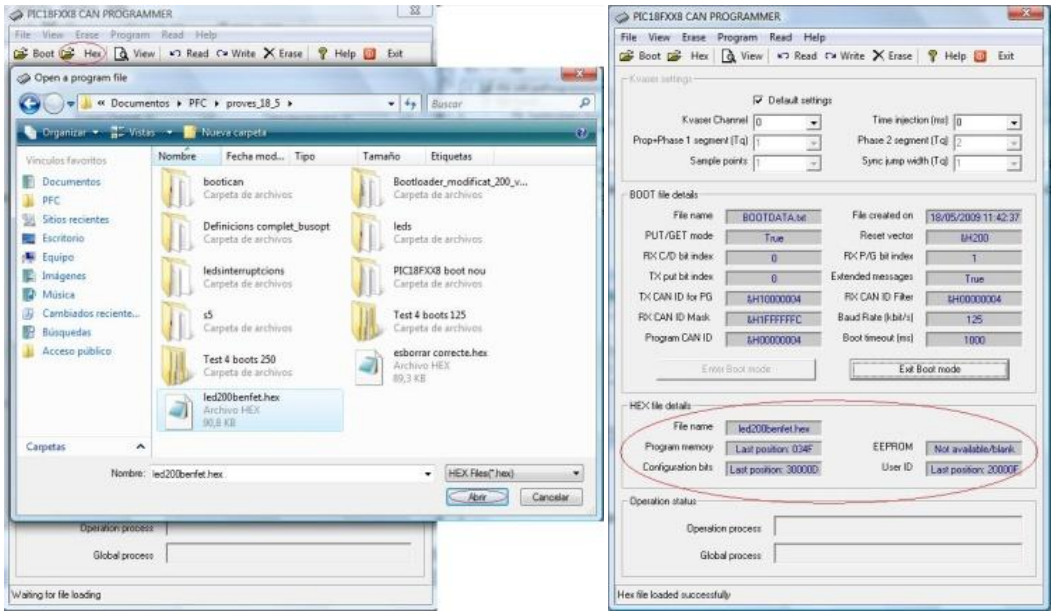


Figura L.14. Obrir fitxer hex amb el programador

L.2.7.Esborrar i Programar el microcontrolador. Llegir si és possible.

Un cop s'hagi entrat en mode *boot* i s'hagi carregat un fitxer en format *HEX* es pot programar, esborrar i si estem en mode *PUTGET* llegir la memòria. Per accedir a les diferents funcions només cal accedir als menús *Erase, Program o Read.*

Els botons de la barra d'eines *Erase, program i Read* realitzen aquestes operacions a totes les memòries, en el cas de programar ho fan només amb aquelles de que es disposa informació per fer-ho.

L.2.8.Sortir del mode boot

Per sortir del mode *boot* polsarem el botó *"Exit Boot mode",* i ens apareixerà un missatge dient que s'ha sortit correctament del mode *boot.*

Figura L.15. Missatge informant que s'ha sortit del mode *boot* correctament

Per sortir del programa polsarem el botó *Exit* de la barra d'eines o bé anirem al menú file i polsarem *Exit.*