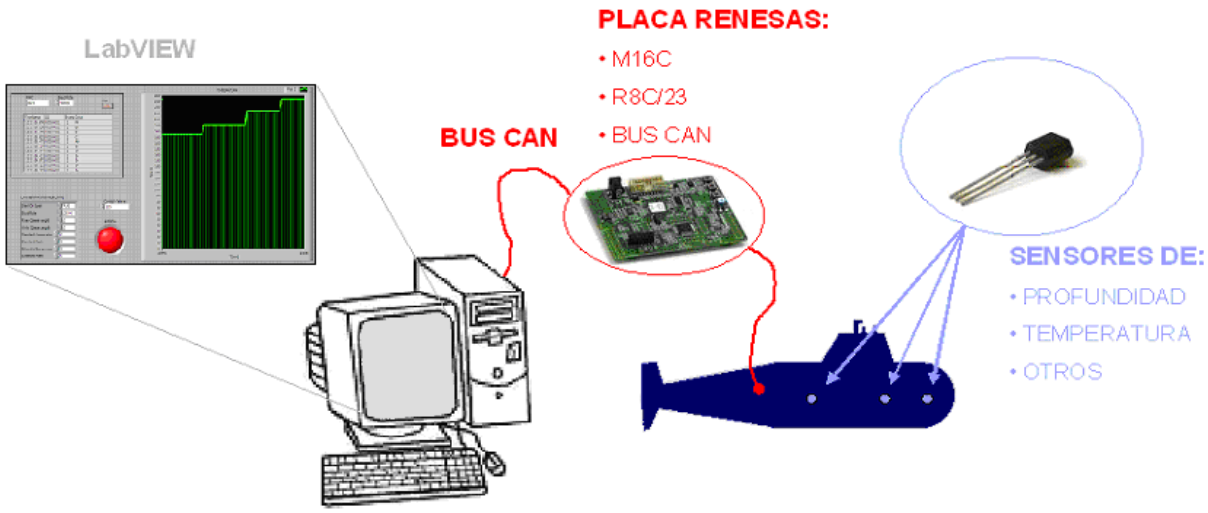
SUMARI

Capítol 1:

INTRODUCCIÓ I OBJECTIUS

1. Introducció i objectius

Els objectius d'aquest projecte consisteixen, ordenats per prioritats, en establir i acondicionar un sistema de comunicació mitjançant bus CAN, entre el microcontrolador d'una placa de proves i un PC, crear un instrument virtual utilitzant el programa LabView per visualitzar les dades que rebem del microcontrolador i per últim crear un hardware per poder implementar físicament el programa creat amb la placa de proves.



Il·lustració 1: Esquema del Projecte

Aquest projecte forma part d'uns dels diversos projectes d'investigació que el Centre Tecnològic de la Universitat Politècnica de Vilanova i al Geltrú, i el grup de treball dels professors Antoni Manuel Lázaro i Joaquim del Rio porten a terme.

El nostre projecte treballa amb un microcontrolador del grup R8C23 de Renesas. Paral·lelament estan treballant amb objectius similars al del nostre projecte dos grups de treball més, aquest grups però treballen amb microcontroladors Microchip i Freescale respectivament.

Un dels principals punts d'interès es l'estudi del bus CAN, el qual es un protocol de comunicacions normalitzat que suporta control distribuït en temps real amb un alt nivell de seguretat i multiplexació.

CAN està orientat a missatges, es a dir la informació que s'intercanvia es descompon en missatges, els quals se'ls assigna un identificador i s'encapsen en trames per transmetre'ls. Cada missatge te un identificador únic dintre de la ret, amb lo qual els nodes decideixen acceptar o no dit missatge.

Una de les nostres tasques rea la de realitzar estudiar com el micorchip Renesas gestionava la comunicació CAN i crear diverses aplicacions de comunicació.

També hem realitzat aplicacions que utilitzen la comunicació SERIE per transportar la informació, això ens ha servit per poder fer la comparativa amb els dos sistemes de comunicació abans esmentats.

Capítol 2:

EINES UTILITZADES

2. Eines utilitzades

Les eines utilitzades han sigut diverses ja que el projecte te part de programació de microcontroladors, comunicació de perifèrics, visualització a l'ordinador i treball de laboratori.

2.1. Software

Per desenvolupar el projecte hem agut d'aprendre un seguit d'eines informàtiques, indispensables per fer la programació i el control de la comunicació, la visualització mitjançant l'instrumentaria virtual i el disseny de plaques electròniques.

Cada programa ens ha sigut útil per realitzar una part del projecte i degut a ser tan diferents hem fet unes guies ràpides per facilitar l'ús i la creació de nous projectes.

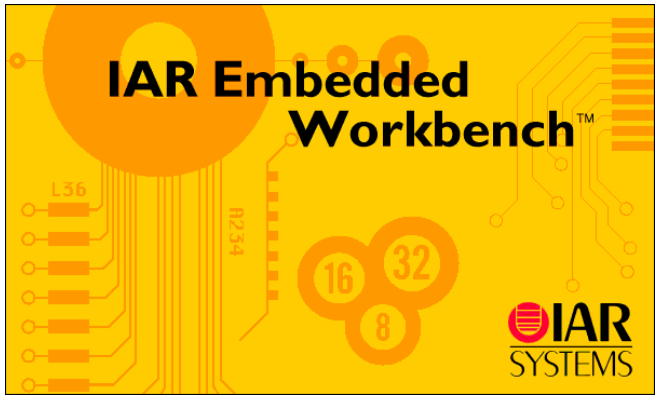
2.1.1. IAR Embedded Workbench

El IAR Embedded Workbench es una potent eina per programar microcontroladors tant en llenguatge C com C++ [7], [25], [1], [2].

Te detecció d'errades i depurador en funcionament (on line), per veure com treballa el programa creat, una vegada en funcionament.

Depenent del model i la casa del microcontrolador que facis servir, et crea un arxiu que et servirà per poder-lo compilar.

L'IAR necessita un compilador per passar el programa creat al microcontrolador.



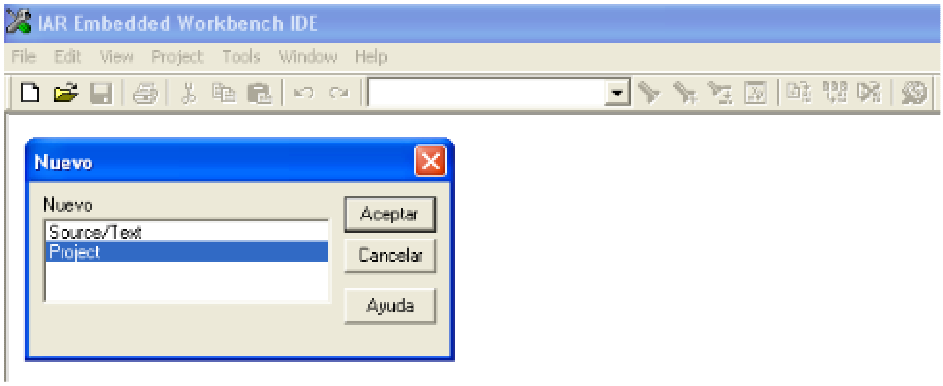
Il·lustració 2: Logotip IAR

A continuació hem fet una guia rapida per crear un projecte amb l'IAR, i que resulti ràpid i intuïtiu seguint les següents il·lustracions.

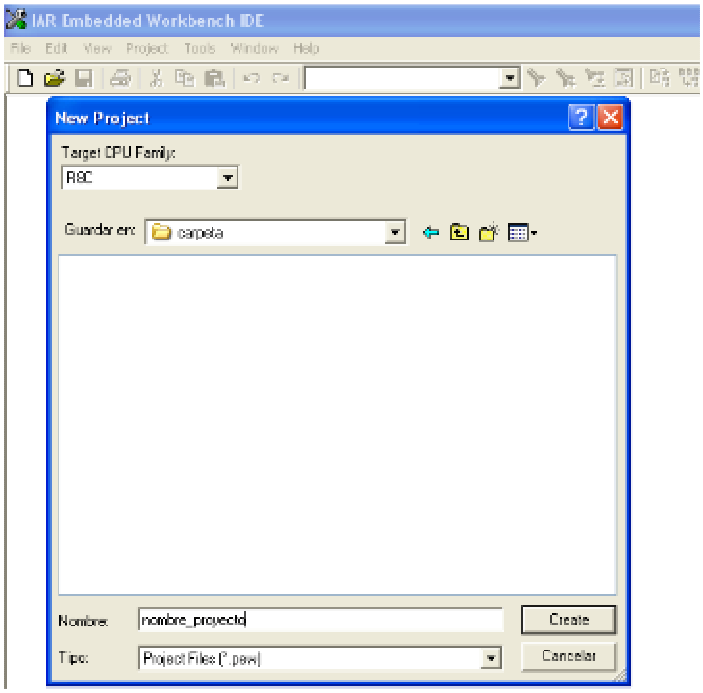
2.1.1.1. Guia ràpida per crear un projecte

La configuració seleccionada es la adequada per treballar amb l'R8C que es la família del microcontrolador de la casa Renesas amb la que hem realitzat el projecte.

- Pas1: Obrir l'IAR System → IAR Embedded Workbench → File → New → Project



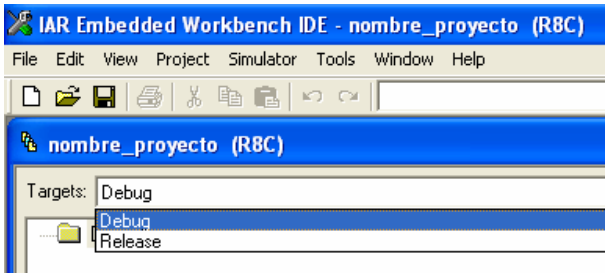
- Pas2: Seleccionar la família de la CPU amb que treballaràs (R8C) i nombrar el projecte.



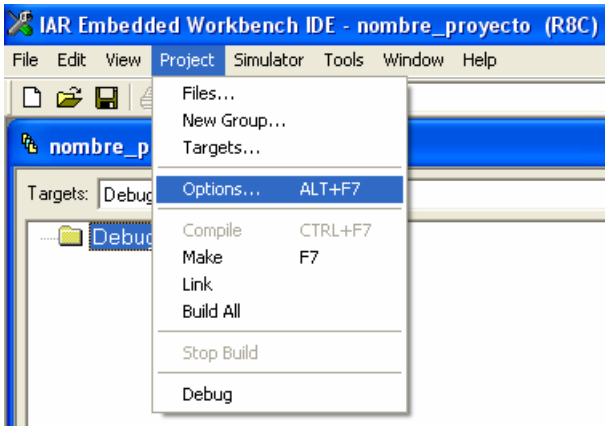
Configuració de les opcions del projecte:

Modificar les opcions del debug:

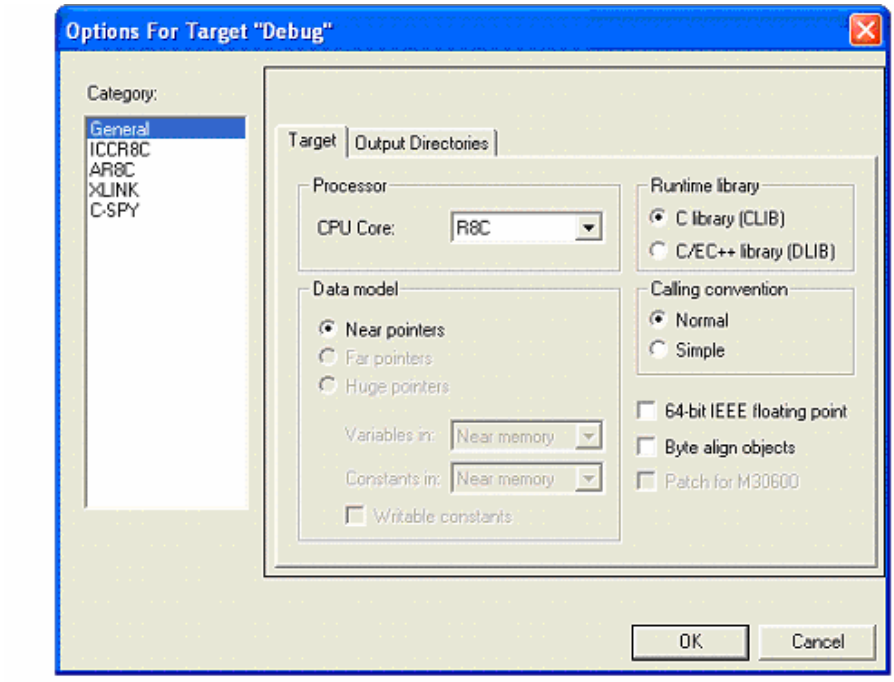
- Pas3: Targets → Debug



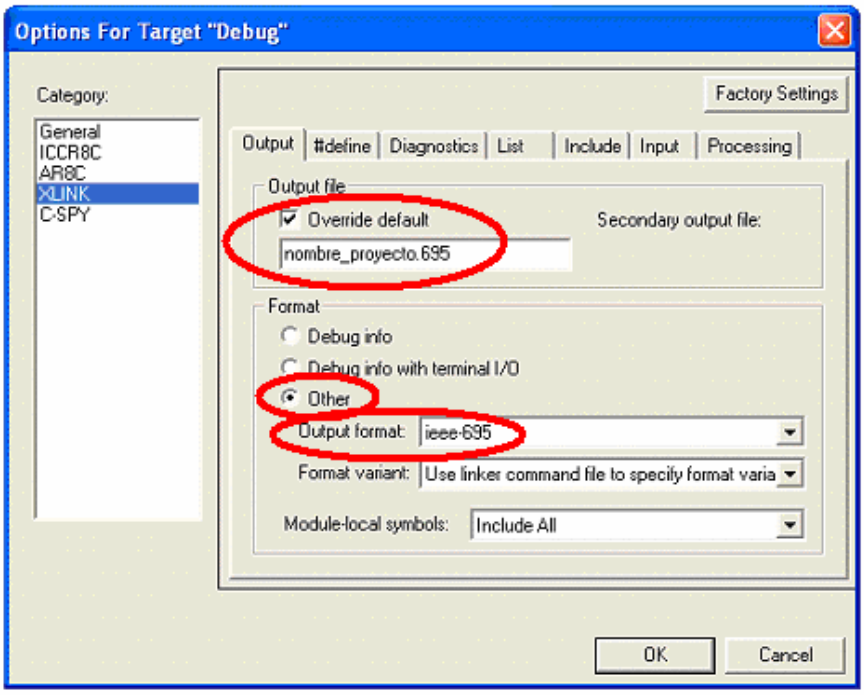
- Pas4:Project → Options



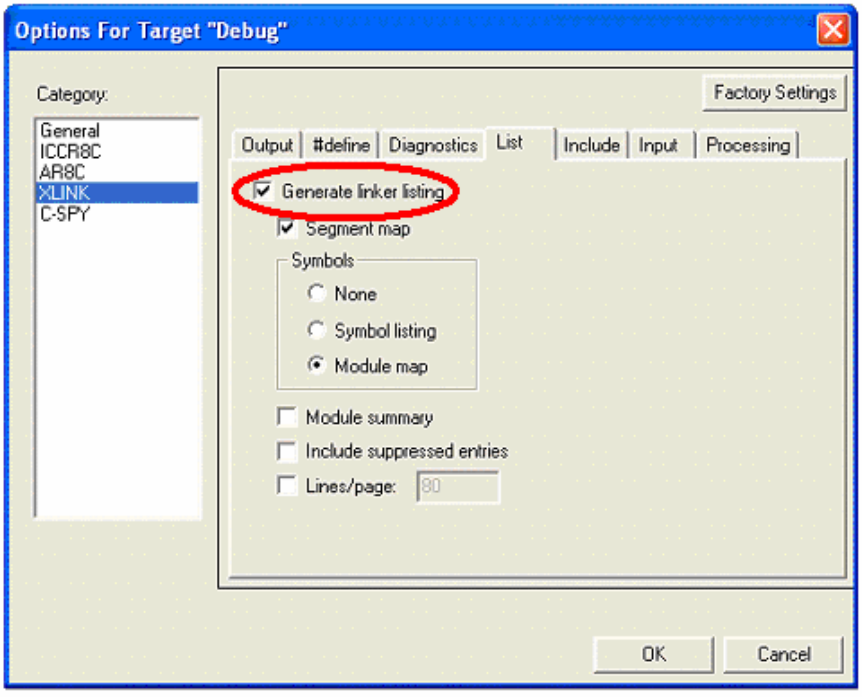
- Pas5: Category → General → Target → CPU Core → R8C



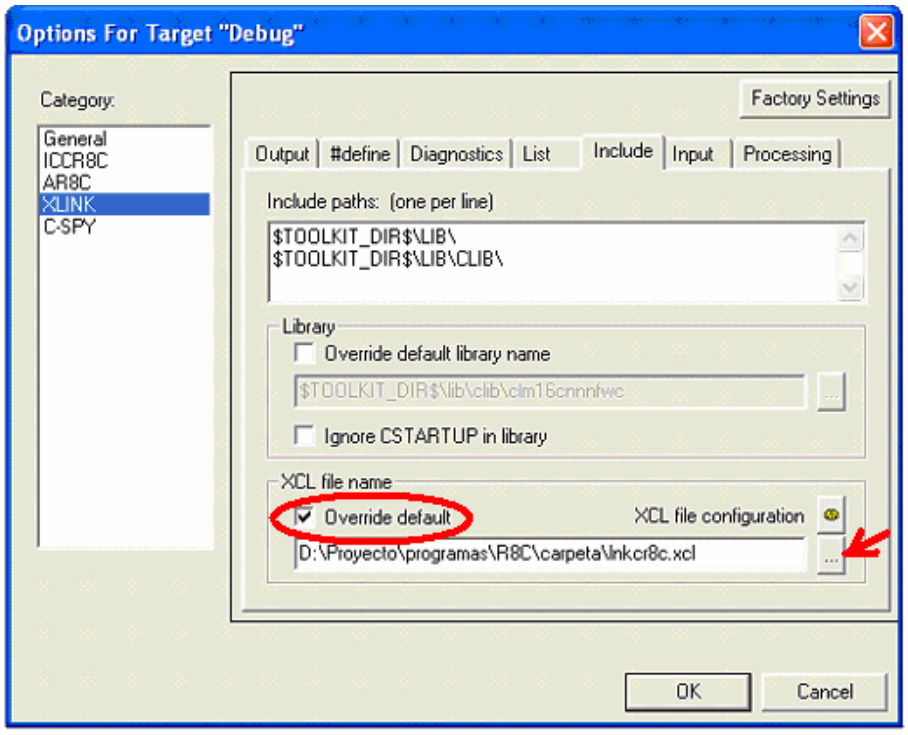
- Pas6: Category → XLINK → Output



- Pas7: Category → XLINK → List

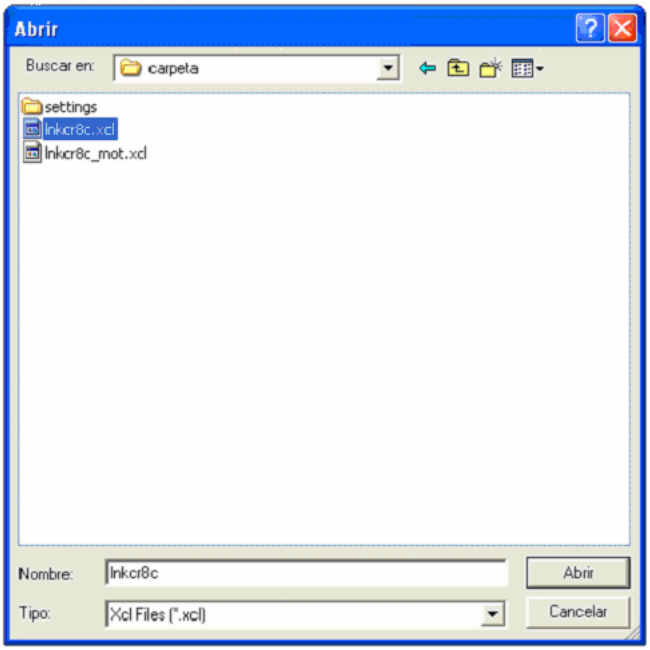


- Pas8: Category → XLINK → Include

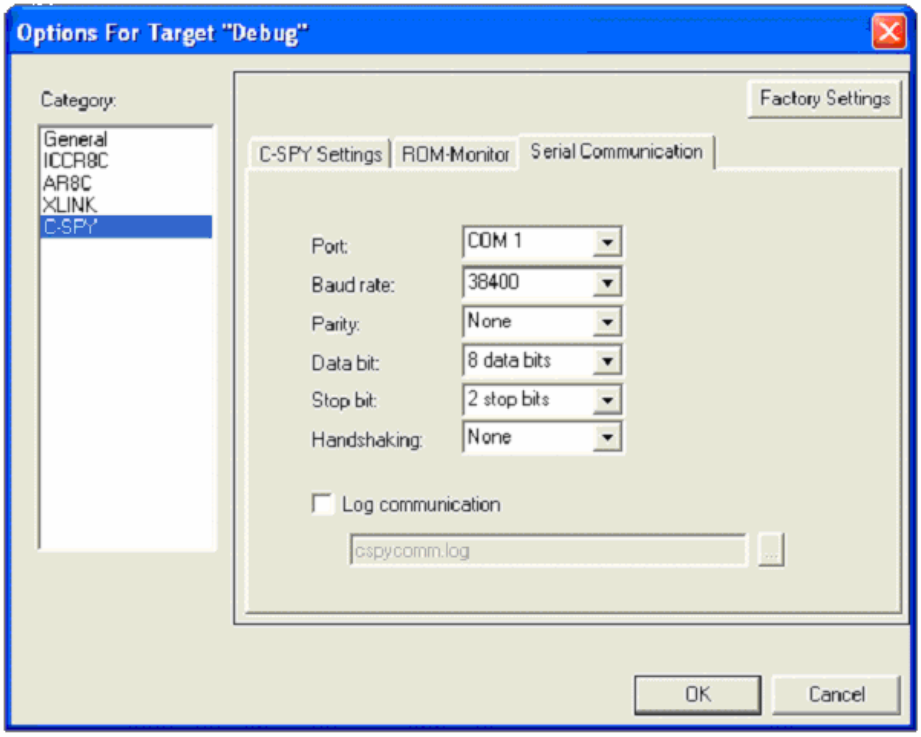


Seleccionar ruta d'accés

- Pas9: Agregar l'arxiu inkcr8c.xcl

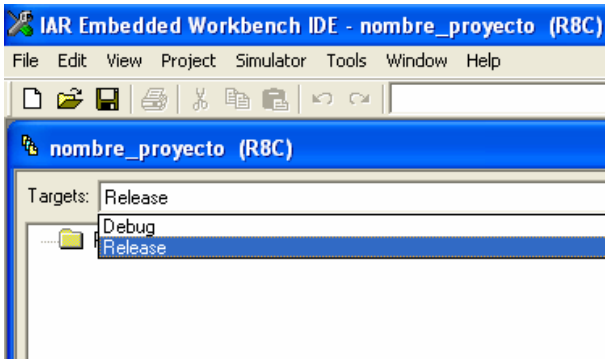


- Pas10: Category → C-SPY → Serial Comunication

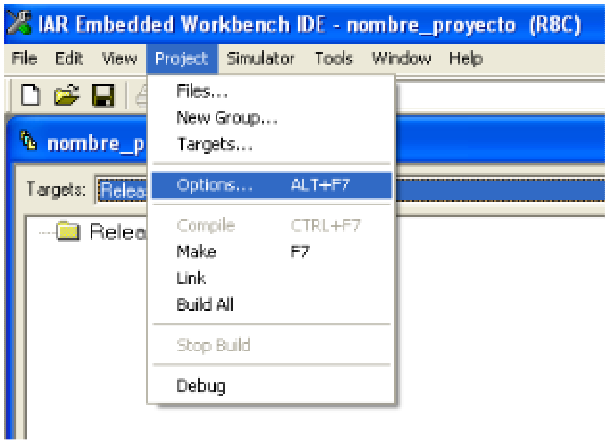


Modificar les opcions del release:

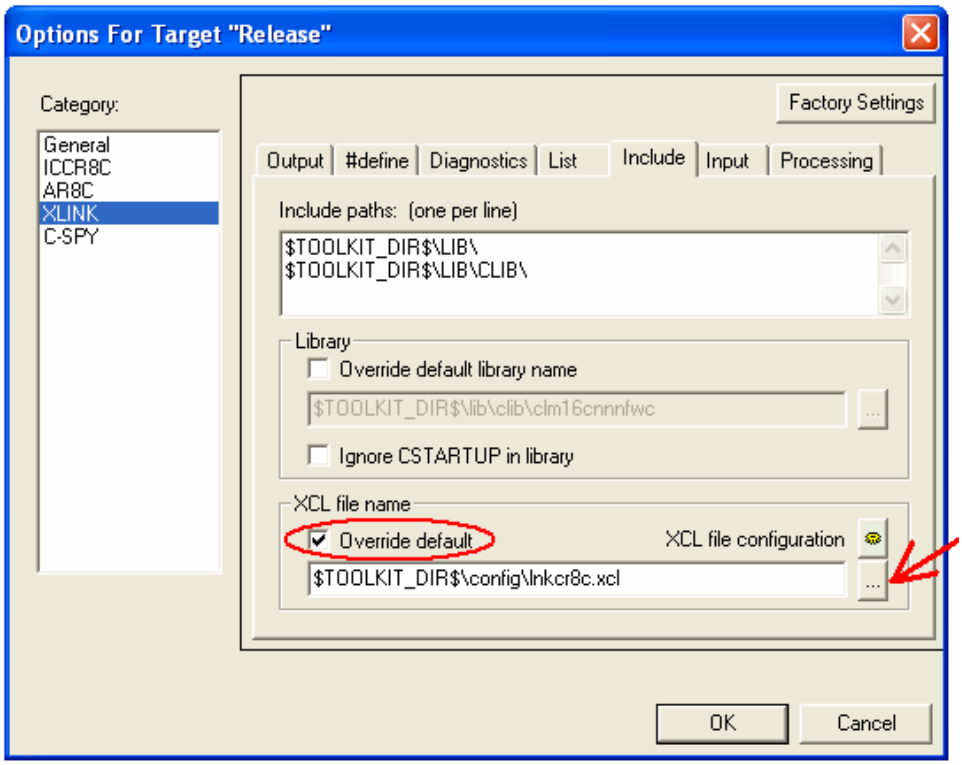
- Pas11: Targets → Release



- Pas12:Project → Options

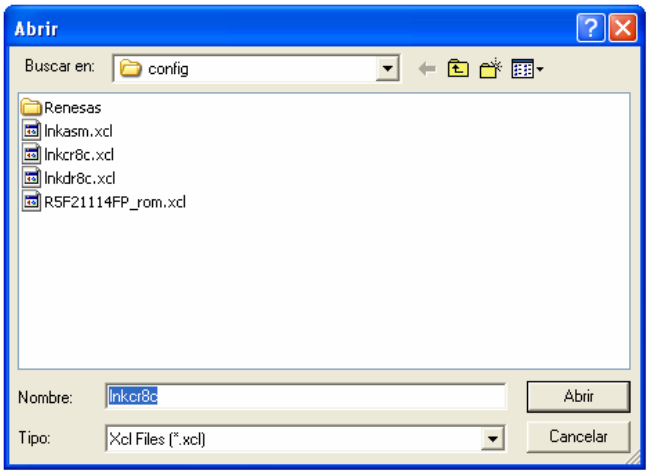


- Pas13: Category → XLINK → Include



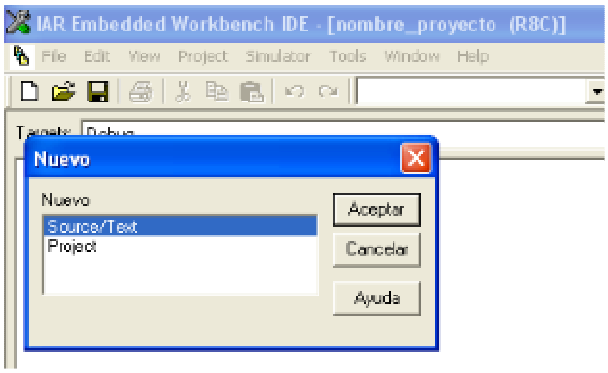
Seleccionar ruta d'accés

lnkcr8c_mot



Una vegada configurat, creem el fitxer punt c per escriure el codi font.

- Pas14: New → Source/Text



Una vegada creat el programa, compilarem, això ens mostra possibles errors, i un cop solucionats, la compilació crea un arxiu de sortida en el format IEE 965 adequat per carregar al depurador Renesas KD30 o Simulador PD30SIM.

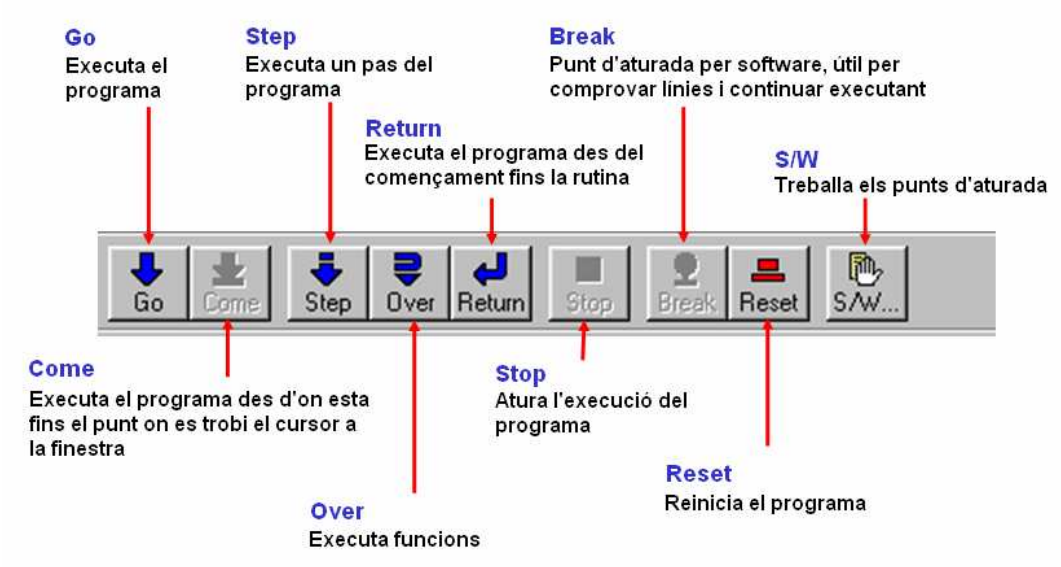
2.1.2. KD30

Aquest "debugger" de Renesas es fa servir per compilar programes per la família R8C, descarregar-los al microcontrolador i córrer-los. Tanmateix, és útil utilitzar un depurador [17].

Il·lustració 3: Logotip KD30

Permet interrompre el programa en qualsevol localització desitjada, i te diverses formes per executar el programa, ja sigui mode continu, per funcions, o per instruccions fent un pas a pas.

Amb la següent il·lustració veiem els botons de la barra de tasques i les seves funcions.



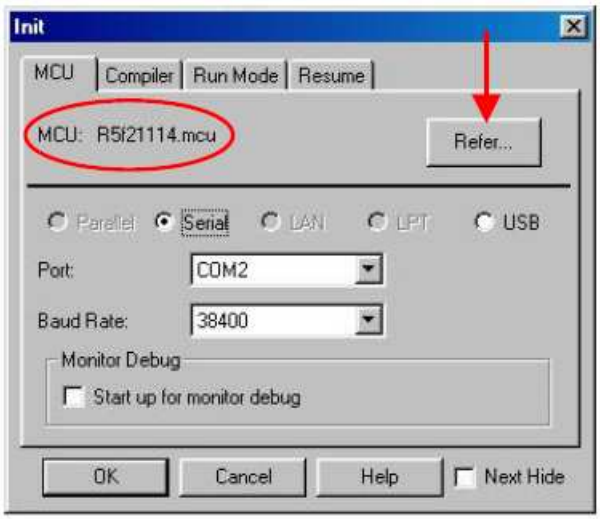
Il·lustració 4: Barra de treball del KD30

En el nostre cas fem servir el KD30 com a complement del IAR Embedded Workbench.

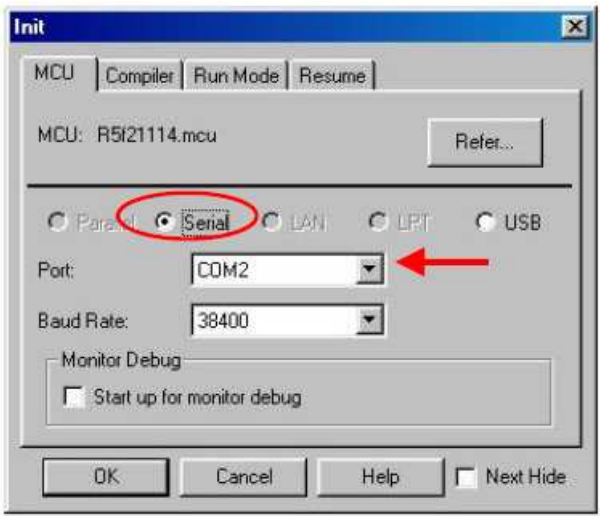
2.1.2.1. Guia ràpida

Per fer servir el KD30, li hem de dir quin port COM reconeix l'ordinador i seleccionar l'arxiu R5F21114.MCU que es troba polsant "Refer...".

- Pas1: Seleccionar el MCU que fem servir.

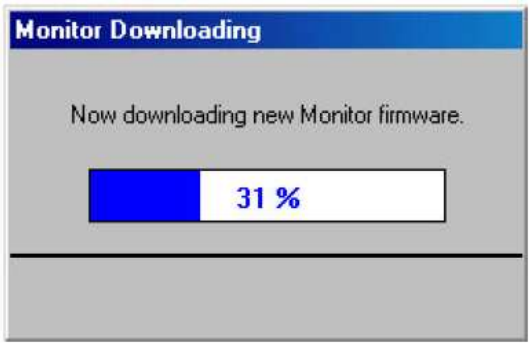


- Pas2: Encara que la connexió a l'ordinador es faci per un cable USB, aquest el reconeix com un port COM, per això seleccionem l'opció "serial". Per saber de quin COM es tracte, obrir Propietats del sistema → Harware → Administrador de dispositius → Ports, i aquí apareix en quin està.



- Pas3: Compiler → IAR ICCM16C.

- Pas4: Important: Abans de clicar "OK" polsar el reset de la placa perquè la reconegui l'ordinador.

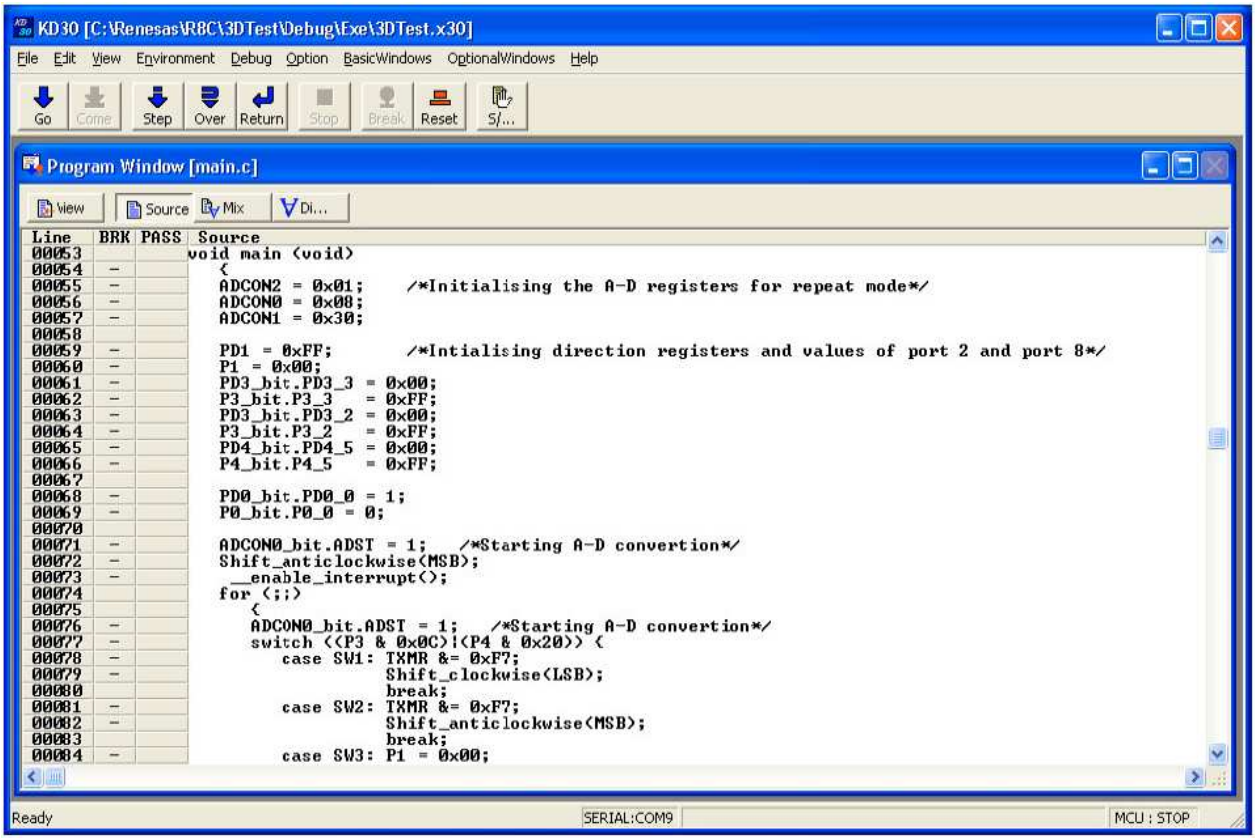


Després que faci clic damunt 'Correcte', el depurador primer descarrega una mica de firmware al microcontrolador. Aquell firmware dóna control ple al KD30 sobre el microcontrolador i el permet descarregar els programes en què està treballant. Pot descarregar un programa seleccionant que sigui d'extensió .x30.

- Pas5: File → Download → Load Module. Seleccionem el programa que hem creat.

- Pas6: Clicant Go carraguem el programa al micro i ja el tenim llest per treballar.

Punt d'aturada per software, util per comprovar linias i continuar executant

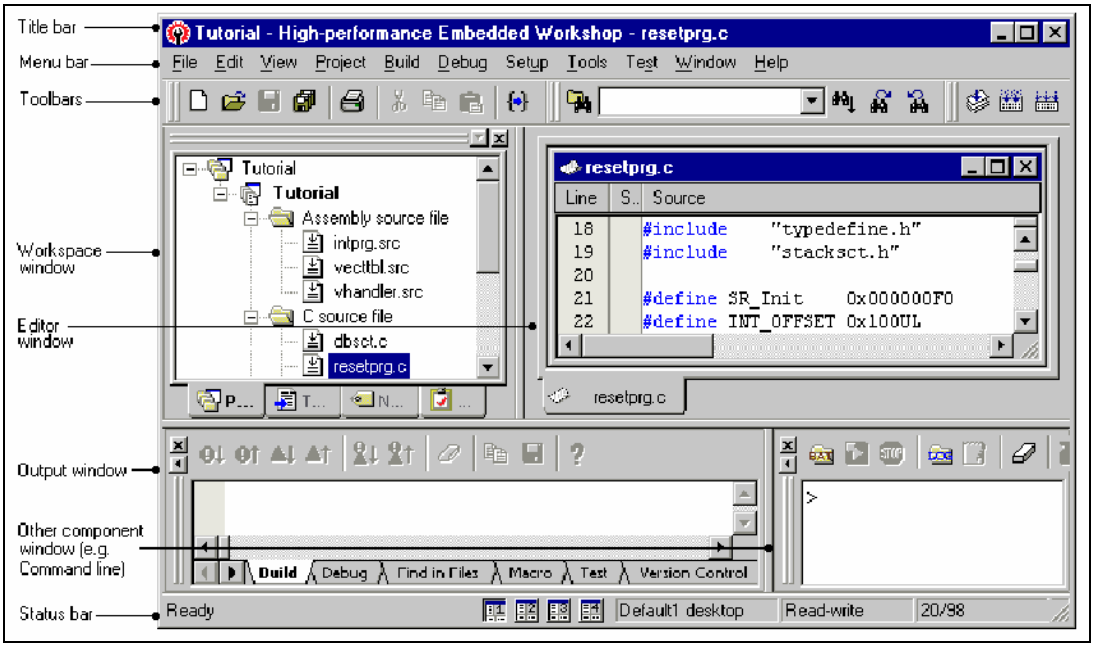


2.1.3. High-performance Embedded Workshop

El programa High-performance Embedded Workshop (HEW) proporciona un grup d'eines molt útils per el desenvolupament i depuració d'aplicacions per microcontroladors Renesas [10].

Il·lustració 5: Logotip High-performance

HEW, consta d'un seguit d'eines potents, encara que fàcils d'utilitzar. El pannell de treball es senzill, les barres d'eines (toolbars) son molt intuïtives, i també permet mouràs entre els diferents arxius que es van creant d'una manera ràpida mitjançant la finestra workspace.

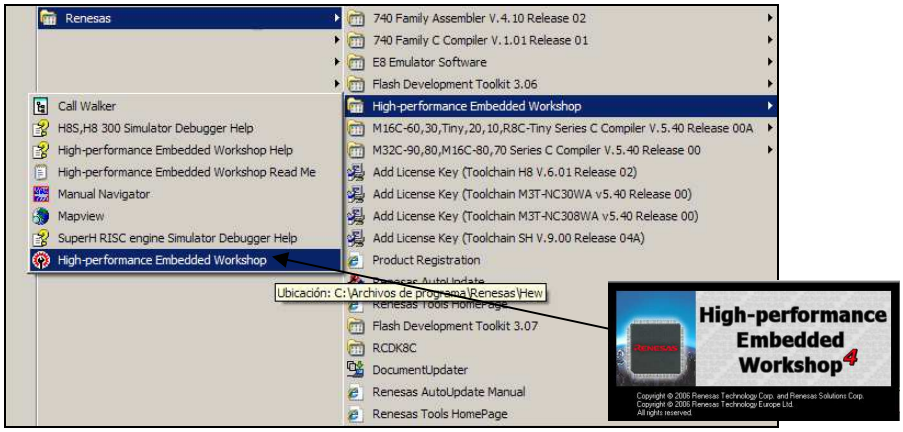


Il·lustració 6: Penell de treball del HEW

Incorpora compiladors de C/C++ [1], [2], i els elements del depurador per a diverses plataformes de depuració incloent-hi emuladors i programes d'avaluació. Permet la depuració i la visualització de variables online.

2.1.3.1. Guia ràpida

El software que utilitzarem per construir els programes del microchip R8C23, s'anomena High-performance Embedded Workshop4. La il·lustració següent ens indica com podem seleccionar-lo de la llista de programes que s'instal·len amb el pack que ens subministra el CD de Renesas.



Com es pot observar en la figura, n'hi ha un marcat en blau, aquest es el que seleccionem per programar el microcontrolador Renesas.

Pas 1; Instal·lació

1 - Posar el CD Renesas a dins del CD-Room, aquest hauria d'arrencar automàticament la instal·lació del programa, si no ho fa, fer doble click al "steup.exe", arxiu que es troba dins del CD.

2 - La instal·lació permet seleccionar la llengua.

3 - Has de clicar <Yes>, si estàs d'acord en les condicions de la llicencia.

4 - La pròxima finestra pregunta sobre la regió, selecciona la que et correspongui.

5 - Selecciona els components del software que t'interessen. Per RSKR8C23 selecciona "High performance Embedded workshop" i totes les subopcions (MC16/60, 30, Tiny, 21,10, R8C\Tiny Series, "Flash desvelopment tool kit" i "autoupdate utility").

Seguidament fes clic a continuar.

Pas 2; Connexió

6 - Ara connecti el E8 a J6/E8 en la board que utilitza el cable pla, també connecta el mòdul de LCD a J8 en el board, assegurant que la clavilla designada 1 correspon al marcat en la board.

7 - Connecti l'emulador E8 al port USB del PC, apareixerà *Nou hardware trobat*, segueixi els passos per instal·lar-lo.

8 - Verifiqui que l'opció "Recomanada" sigui seleccionada i faci clic a <Next>.

9 - Si utilitza Windows XP, salti fins el pas 11; altrament faci clic a <Next>.

10 - Faci clic a <Next> per instal·lar el connector.

11 – Seguidament polsi <Finish> per tancar el "wizard".

Pas 3: High-performance Embedded Workshop Workspace

"High-performance Embedded Workshop" integra diverses eines; compilador, muntador, depurador i editor en una interfície d'usuari gràfica comuna. Per aprendre més sobre com utilitzar el High-performance Embedded Workshop, obre el manual de "the High-performance Embedded Workshop" i instal·la'l al teu ordinador.

12 - Executa el High-performance Embedded Workshop des del menú d'inici.

13 - En el quadre de diàleg "welcome": Verifiqui que l'opció " Create New Workspace " és seleccionada i pulsi <OK>.

14 - En el quadre de diàleg polsi l'opció "New Project Workspace".

15 - Introdueix un nom pel workspace. El nom del projecta serà automàticament completat amb el nom del "workspace".

16 - En la finestra "RSKR8C23 Step 1": Selecciona "Tutorial" i polsa <Next>.

17 - En la finestra "RSKR8C23 Step 2" : Clica <Finish>.

18. En la finestra d'informació general del projecta: Clica <OK>.

El projecte que has creat te dues configuracions:

- La configuració "Release" pot ser usada per la emissió de la versió final del codi.

- La configuració “ Debug” permet fer modificacions mentre estem provant i depurant el codi.

19 - Selecciona “Debug” en el menú desplegable de l’esquerra de la barra d’eines.

20 - Clica la icona “Build” per assemblar, compilar i linckar el projecte.

Pas 4: Programar i depurar

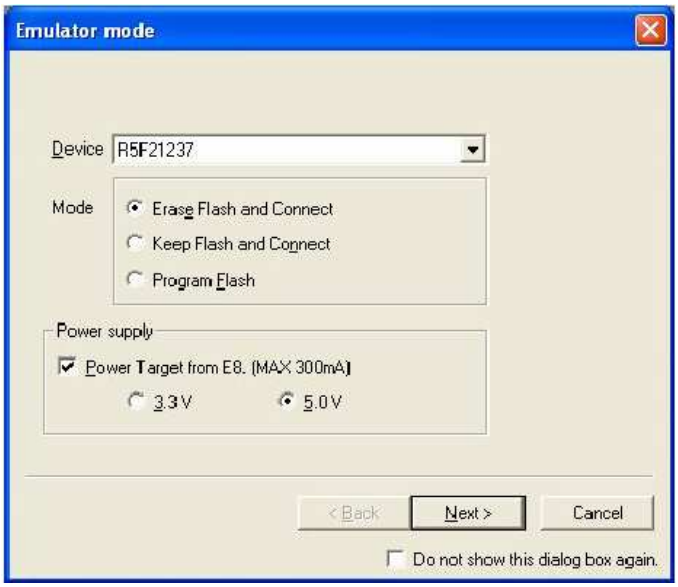
21 - Assegurat que l’opció del menú desplegable de la dreta, en la barra d’eines, sigui:

“SessionR8C_E8_SYSTEM”.

22 - Clica el boto <Connect> en la barra d’eines “debug”.

23 - Selecciona el tipus de microcontrolador correcta (ex. R5F21237 o RSKR8C23).

Fixa’t que el “Emulator mode wizard” mostrat aquí, sols apareixerà la PRIMERA vegada que es connecta la targeta al projecte. En les següents connexions apareix el diàleg de “Emulator setting”, escolliu les mateixes opcions per la connexió que en el “Emulator mode wizard”.



24 - Selecciona “Erase Flash and Connect”.

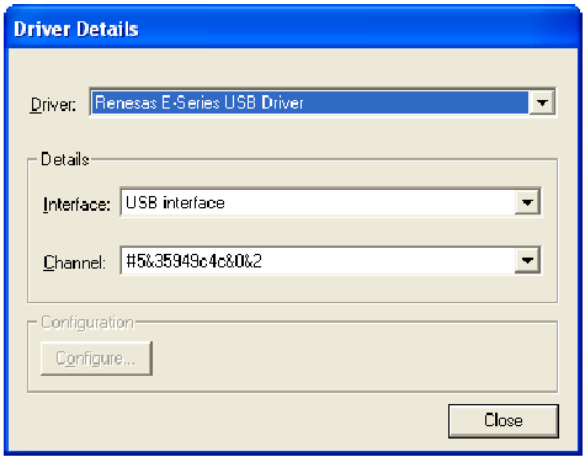
25 - Si “E8” alimenta a la CPU de la board, selecciona “Power Target from E8” i ajusta-la a 5.0 volts.

26 - Clica <Next>, <Next>, i finalment <Finish>.

27 - El primer cop que facis servir E8, apareixerà aquest missatge.

28 - Clica <OK>.

29 - Selecciona “Renesas E-Series USB Driver”. Selecciona “USB Interface”. El numero del canal serà diferent.

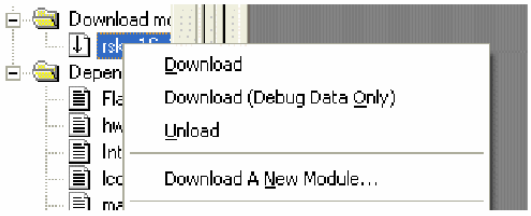


30 - Clica <Close>.

31 - Accepta qualsevol avís de diàleg que una versió antiga de “E8 firmware” es necessària per funcionar amb l'aplicació.

32 - Permet descarregar el “fireware” per completar, això pot portar uns quants segons.

33 - De la llista desplegada del menú “download module” , clica l’arxiu desplegat amb el boto de la dreta del ratolí i selecciona “Download”. El codi serà descarregat al microcontrolador. Això pot portar uns segons.



34 - Clica el boto <Reset Go> .

El codi s’executarà i podràs observar com s’encenen els LEDs a la board.

35. Clic el boto <Stop>.

El codi deixarà d'executar-se i al taulell de programa s'obrirà el codi font actual.

Després de seguir els passos anteriors ja tenim el sistema preparat per fer i executar els nostres propis programes.

2.1.4. LabView

LabView es una eina gràfica de test, control i disseny mitjançant la programació. El llenguatge que fa servir s'anomena llenguatge G. Aquest programa va ser creat per National Instruments (1976) per funcionar en maquines MAC, va sortir al mercat per primer cop en 1986. Ara esta disponible per les plataformes Windows, Unix, MAC i Linux, i va per la versió 8.20 i 8.21 amb el suport de windows vista [3], [18].

Il·lustració 7: Logotip LabView 8

La seva principal característica es la facilitat d'us, vàlid per programadors professionals, com per persones amb pocs coneixements en programació que poden fer programes relativament complexos, impossibles per ells de fer amb llenguatges tradicionals. També es molt ràpid de fer programes amb LabView i qualsevol programador, per experimentat que sigui, pot beneficiar-se d'ell. Per els amants del complexa, amb LabView poden crear-se programes de milers de VIs(equivalent a milions de pagines de codis de text) per aplicacions complexes, programes d'automatitzacions de milers de punts d'entrades i sortides, etc.

Presenta facilitats per la manipulació de :

Port Sèrie, Port paral·lel, GRIB, PXI, VXI, TCP/IP, UDP, DataSocket, Irda, Bluetooth, USB, OPC...

En aquest projecta utilitzarem Labview per manipular el port sèrie i llegir dades del bus CAN mitjançant la PCMCIA – CAN/2 de National Instruments.

Els programes fets amb LabView se'ls anomena VI (virtual instrument), el que dona una idea de l'ús, sobre tot, el control d'instruments. Entre els seus objectius estan el reduir el temps d'aplicacions de tot tipus(no tan sols en àmbits de test, control i disseny). Això no significa que l'empresa faci únicament software, sinó que busca combinar aquest software amb tot tipus de hardware, tant propi com targetes d'adquisició de dades, PAC, Visió, i altre hardware com de terceres empreses.

Cada VI consta de tres components que seguidament enumerarem:

1- Un pannell frontal: Es la interfície d'usuari.

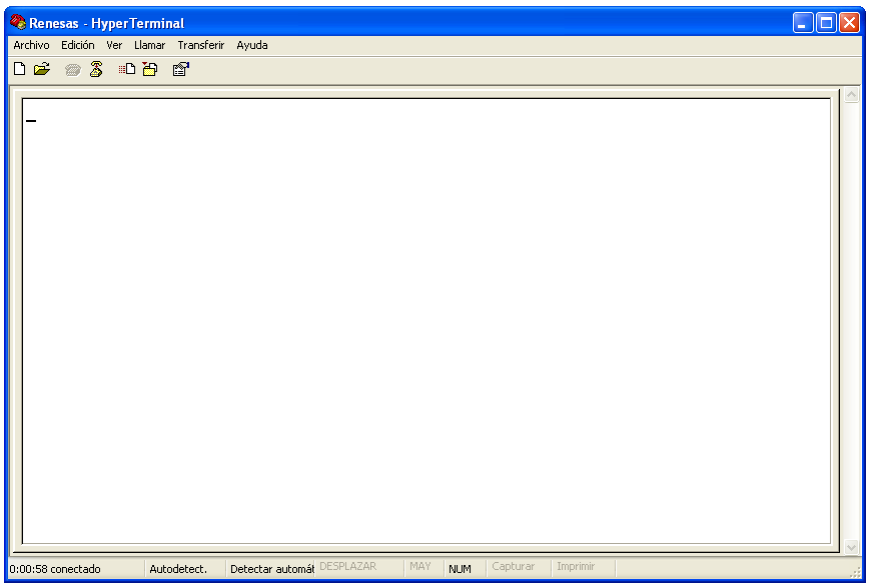
2- Un diagrama de blocs: Conte el codi font gràfic que defineix la funcionalitat VI.

3- Connector o icona: identifica cada VI, de manera que podem utilitzar-lo dintre d'un VI. Un VI dins d'un altre VI, rep el nom de subVI. Es com una subrutina en un llenguatge de programació basat en text.

Utilitzarem aquest programa en les nostres aplicacions per crear un pannell, on l'usuari podrà visualitzar les dades del port sèrie o les del bus CAN.

2.1.5. HyperTerminal

HyperTerminal es un programa que pot utilitzar-se per connectar amb altres equips, llocs Telnet, sistemes de butlletins electrònics (BBS, *Butlletí Board Systems*), serveis en línia i equips host, mitjançant un mòdem, un cable de mòdem nul o Ethernet.



Il·lustració 8: HyperTerminal

HyperTerminal es un medi útil per configurar i provar el mòdem o examinar la connexió amb altres llocs, encara que també es pot utilitzar HyperTerminal amb un servei de butlletí electrònic per tenir accés a informació d'equips remots.

HyperTerminal grava els missatges enviats o rebuts per serveis o equips situats al altre extrem de la connexió. Per aquesta raó, pot actuar com una valuosa eina per solucionar problemes de configuració i us del mòdem. Per confirmar que el mòdem es ben connectat o veure la seva configuració, pot enviar comandos mitjançant el HyperTerminal i veure els resultats. HyperTerminal ofereix la funcionalitat de desplaçament, que li permet revisar el text rebut que sobrepassi l'espai de la pantalla.

Pot utilitzar HyperTerminal per ajudar a depurar el codi font des d'un terminal remot. També pot utilitzar HyperTerminal per comunicar-se amb els equips antics basats en caràcters. HyperTerminal serveix també per transferir arxius grans d'un equip a un equip portàtil mitjançant el port sèrie, en lloc de realitzar la configuració del portàtil en una ret.

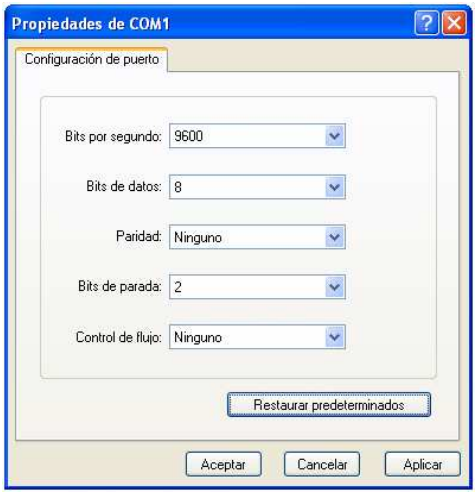
HyperTerminal hi es dissenyat per ser una eina fàcil d'utilitzar i no ve a substituir a altres eines principals disponibles en el mercat.

La funcionalitat que li donem, es la de provar la comunicació entre el microcontrolador i l'ordinador, i veure que realment el codi font creat, te correctament programades les velocitats i les altres opcions de configuració, abans de treballar amb el LabView.

Per configurar el HyperTerminal tens que fixar el port COM per el que està connectada la placa a l'ordinador, i establir les propietats de connexió, que deuran coincidir amb les fixades a l'aparell que connectis.

Els paràmetres a configurar son:

- Bits per segon: Velocitat a la que es farà la connexió.
- Bits de dades: número de bits que es transmetran.
- Paritat: Pot escollir entre paritat parella, senar o no fixar cap.
- Bits de parada: Aquests bits li serveixen al HyperTerminal per identificar el principi i el final d'un enviament.
- Control de flux: Les possibilitats son Xon/Xoff, Hardware o cap control.



2.1.6. Protel

Altium, empresa creadora del Protel, ha estat pionera en la tecnologia de disseny de circuits basada en PC's. Protel va ser la primera eina de disseny preparada pel sistema operatiu Windows. La combinació d'un software d'alta qualitat disponible en un paquet fàcil d'aprendre, utilitzar i comprar, ha fet del Protel sinònim d'innovadora tecnologia de disseny de PCB, i representa una elecció intel·ligent per els dissenyadors [4], [24].

Al llarg dels anys, els productes Protel s'han utilitzat per dissenyar circuits impresos en tot el mon, i els han fet servir companyies i enginyers per desenvolupar els seus productes electrònics.

Il·lustració 9: Logotip DXP 2004

El Protel, es una eina de disseny assistit per ordinador, entre les seves aplicacions, podríem destacar, l'editor d'esquemàtics, que te una llibreria amb una amplia gamma de components i la possibilitat de retocar, importar i crear de nous, per que s'adaptin a les necessitats del teu circuit. Una altre aplicació, es el Protel PCB, està orientada per treballar de forma senzilla amb el disseny final de la placa.

Dintre de la PCB cal destacar les regles d'enrutat, on pots definir des del color de les pistes per la cara top i bootom, amplada màxima i mínima, distancia entre pistes, si permets els angles rectes, etcètera.

També tens l'opció de fer l'enrutat manual o automàtic, amb diverses possibilitats de prioritats de distribució.

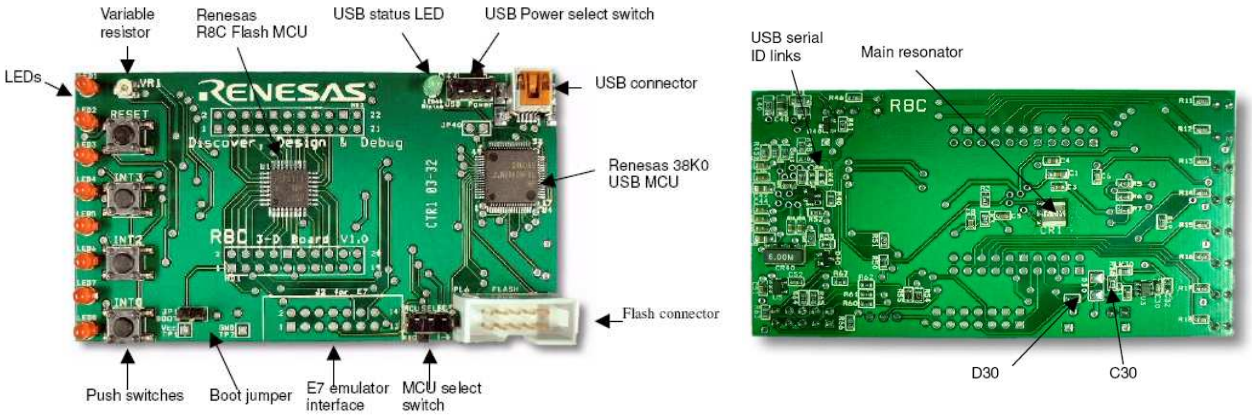
Aquest programa incorpora wizards, els wizards ens faciliten el treball, per exemple tenim un per crear la PCB, i ens son mol útils sobre tot, si et trobes en el nivell d'aprenentatge del programa.

Una altre eina que ens ha sigut molt útil dintre del Protel ha sigut la generació d'arxius Gerber i NC Drill, aquests arxius, duen tota la informació de pistes, footprints i forats, en el nostre cas, per exportar el disseny a una fressa automàtica per crear la placa, en comptes d'isolar-la.

2.2. Hardware

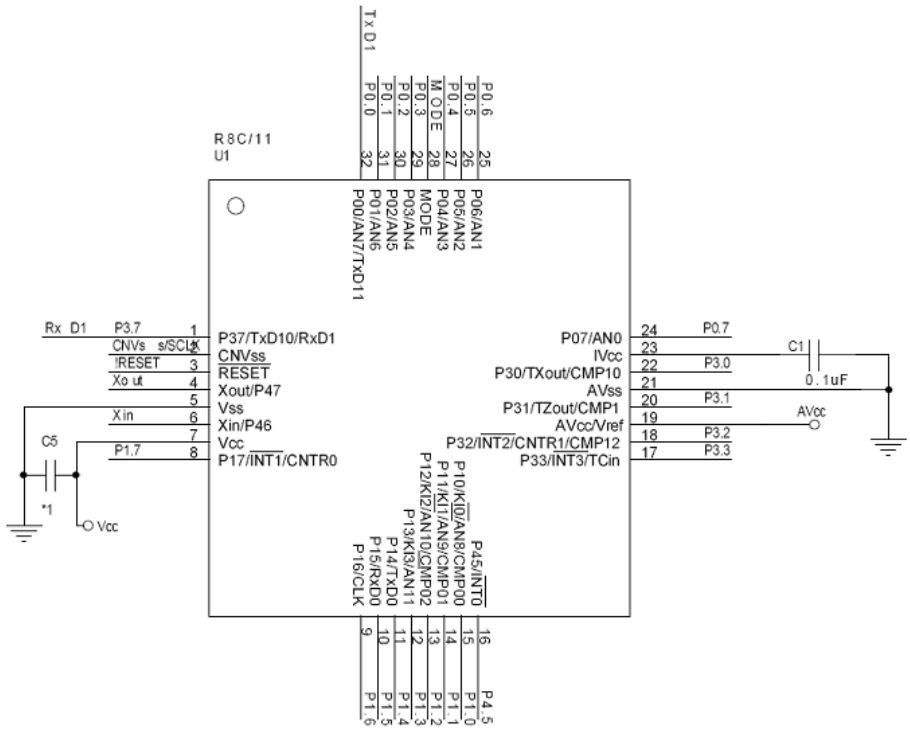
El hardware empleat per el projecta consta de dues plaques de proves de la casa Renesas, una tarja PCMCIA de CAN per poder connectar el PC amb la placa del microcontrolador per el bus CAN, una centraleta CAN que envia i rep missatges i el desenvolupament del nostre hardware a mida per poder connectar el microcontrolador tan pel bus SERIE com pel CAN.

2.2.1. 1ª placa R8C 3-D Starter Kit



Il·lustració 10: R8C 3-D Starter Kit

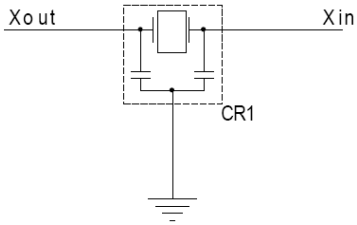
L'objectiu final del treball amb aquest "startet kit" es programar el microcontrolador perquè tingui una comunicació amb l'ordinador via bus SERIE i utilitzar aquesta per enviar dades dels canals analògic/digital [6].



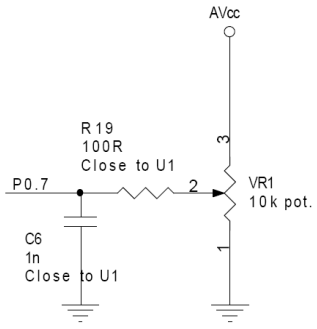
Il·lustració 11: Pins del R5F21114FP

El microcontrolador que utilitza aquesta placa, es el R5F21114FP amb una ROM de 16K bytes, una RAM de 1K byte i memòria Flash.

Es un microchip de 32 potes de les quals les mes interessants per el nostre projecte son: Xin i Xout per connectar l'oscil·lador extern de 20M i fer els càlculs de temps.



I la P07/AN0 que te un potenciòmetre per fer conversions A/D.

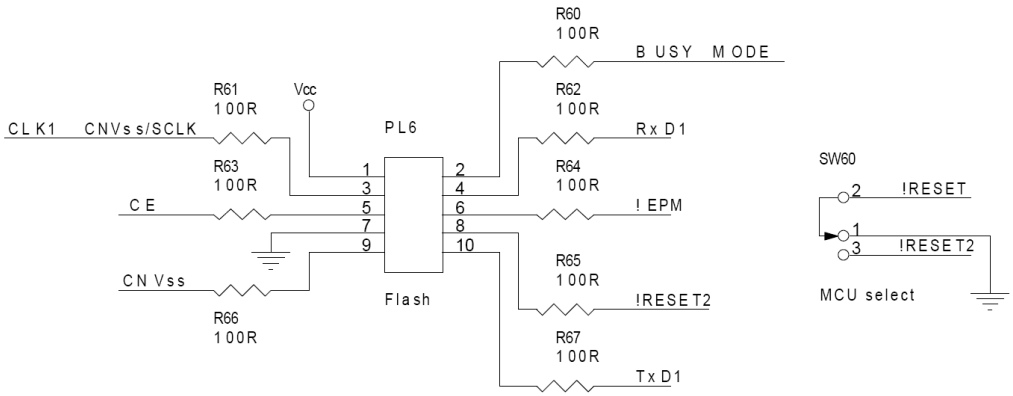


Aquest Starter Kit de prova te dos opcions a l'ora d'alimentar i transferir els programes:

Es pot fer servir el connector Flash o el connector USB.

El connector Flash, depèn de l'accessori de programació. En cas de tenir aquest accessori, es procediria a connectar-lo a la clavilla PL6 o Flash.

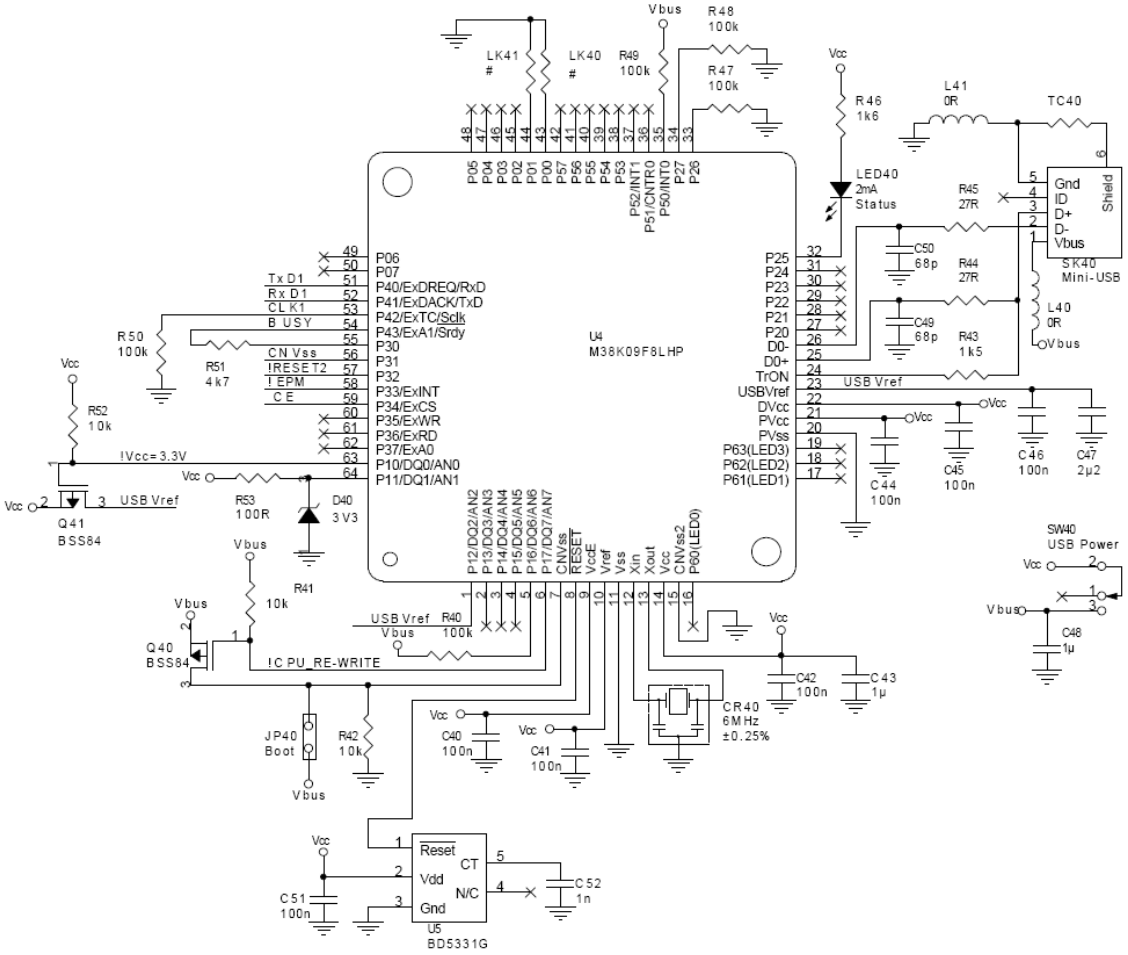
Si decideixes fer servir aquest mètode, cal desplaçar el selector anomenat "MCU select switch".



Il·lustració 12: Connector Flash i selector

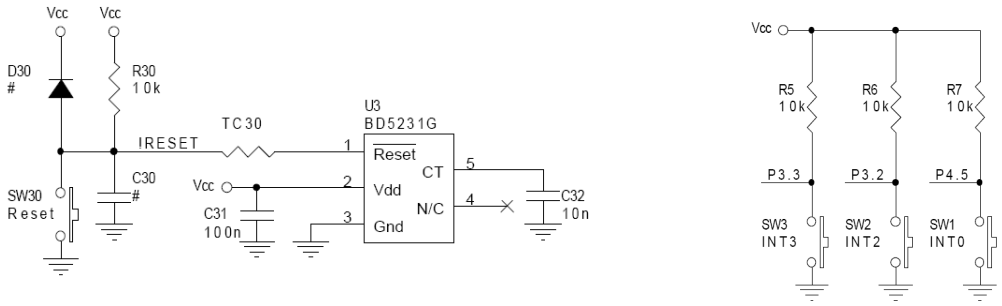
L'altre opció, es el connector USB, que es reconeixerà a l'ordinador com un port COM.

Per activar-lo te el selector USB POWER i el LED40 per indicar el funcionament.

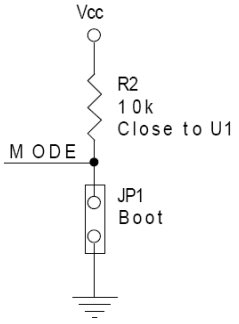


Il·lustració 13: Sistema de connexió USB

Te quatre polsadors de lògica negativa, que envien un 0 lògic quan estan polsats, un d'ells es el reset de la placa, útil per restablir la placa al mode d'inici, per que la reconegui l'ordinador i per poder guardar els programes a la memòria Flash.



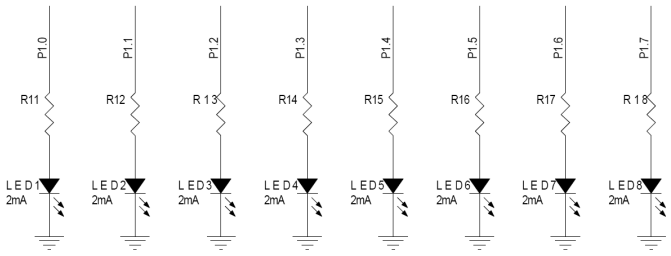
Per poder gravar el programa a la memoria del microcontrolador, per tal de no haver d'estar connectat amb el programa KD30, tenim el "Boot jumper". La forma d'utilitzar-lo, es transmetin el programa des del KD30 i una vegada programat, polsar el stop per que no es quedi penjat sense trobar la placa.



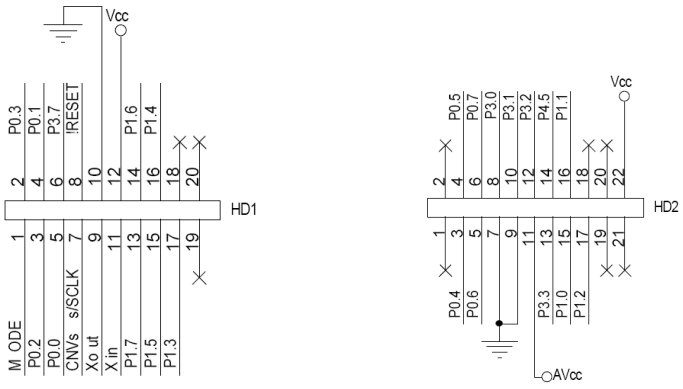
Una vegada ja te el programa dins i el KD30 esta aturat, traiem el jumper i polsem el reset per tal que es quedi enregistrat a la memòria Flash.

La memòria Flash esta limitada a 100 programacions. Aquesta limitació fa que hagis d'estar segur a l'hora d'escriure a la memòria Flash.

També té vuit led's, molt útils per poder fer proves i utilitzar-los per seguir el programa visualment i trobar possibles errades.



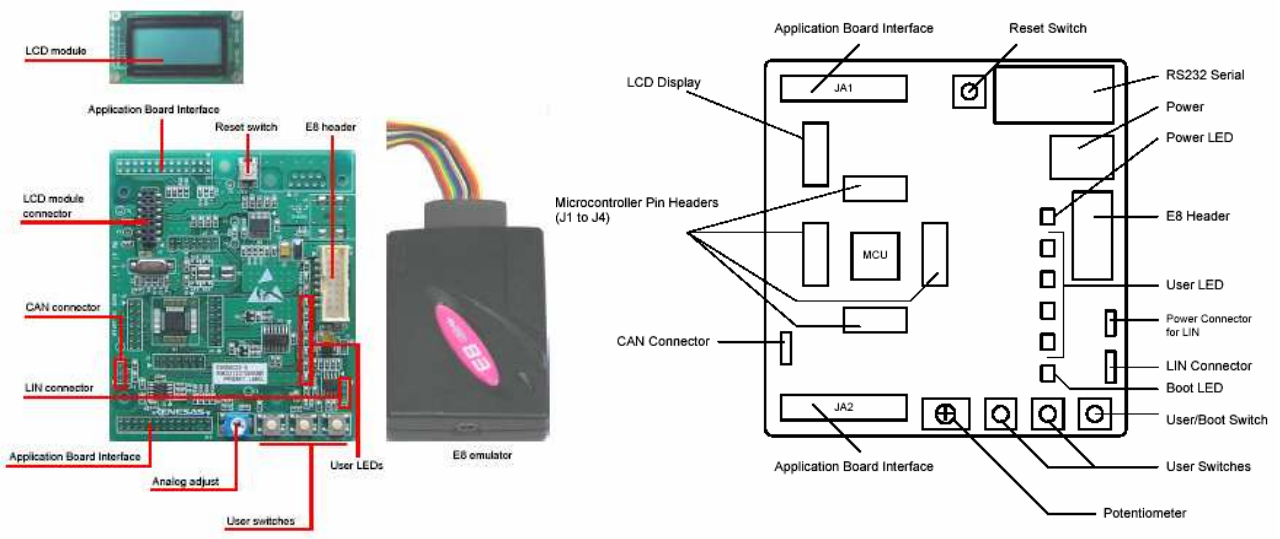
Els Pins lliures, tels deixa disponibles en els connectors HD1 i HD2.



El port sèrie que fem servir al nostre projecte esta habilitat al HD1 pins 15 i 16. El pi 16 connectat a P1.4 es. TxD0, per transmetre dades, i al pin 15 connecta amb P1.5 del microcontrolador i el fem servir com RxD0 per rebre dades.

2.2.2. 2ª placa Starter Kit for R8C/23

La placa Renesas Starter Kit for R8C/23, està formada per una placa de proves, amb els components necessaris per provar les principals característiques del microcontrolador. També consta de l'E8, un emulador que ens permetrà depurar els programes que desenvolupem, i així facilitar-nos la tasca de depuració d'errors [8].



Il·lustració 14: R8C/23 Starter Kit

Switches: La board conté quatre botons. La funció de cada un d'ells s'explica en la taula següent:

switch	Funció	Microcontrolador
RES	Quan es polsat, el microcontrolador es reseteja.	RESET Pin 7
SW1	Connectat a una línia d'interrupció de IRQ per usar-se com a control	IRQ0 Triga Pin 25
SW2	Connectat a una línia d'interrupció de IRQ per usar-se com a control	INT 1, Pin 20 (port 1 pin 7)
SW3	Connectat a una línia d'interrupció Key per fer-se servir com a control	KI3 Pin 24 (port 1 pin 3)

Taula 1: Funció del polsadors de l'Starter Kit R8C/23

LEDs: La board conté sis leds. El led verd s'encén quan la placa esta connectada a una font d'energia. Els quatre led's restants s'il·luminaran quan els ports als quals estan connectades donin un zero.

Referència del led	Color	Port del microcontrolador	Pin del microcontrolador
LED 0	Verd	Port 2.4	15
LED 1	Taronja	Port 2.5	14
LED 2	Vermell	Port 2.6	13
LED 3	Vermell	Port 2.7	12

Taula 2: Leds

Potenciòmetre: Aquest potenciòmetre esta connectat al canal AN8 del converso analògic/digital, P1.0 del microcontrolador. Potser usat per variar el valor del voltatge que entre pel pin juntament amb Vref i massa.

Port sèrie: El port sèrie 1 del microcontrolador programable es connectat al transceiver RS232.

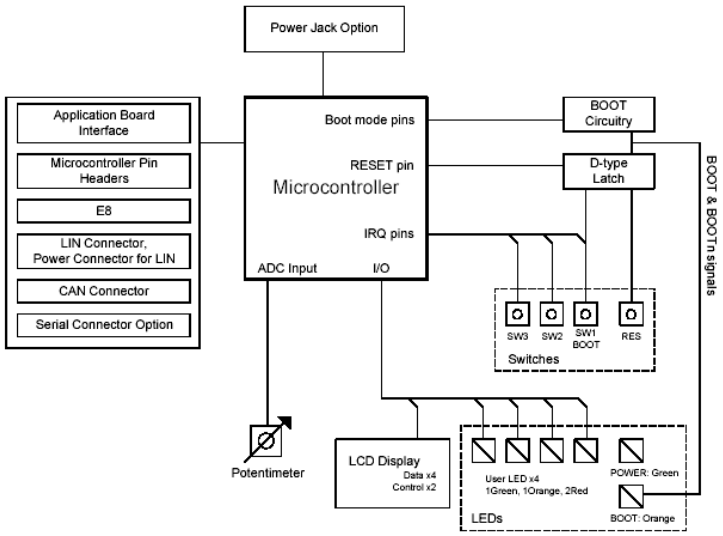
Si hem d'utilitzar el port sèrie, opcionalment podem connectar-los mitjançant unes resistències de zero ohms, ja que no venen de sèrie.

Les connexions que s'haurien de soldar es mostren en la següent taula.

Descripció	Funció	Localització
TxD1	Programació del port sèrie	R45
RxD1	Programació del port sèrie	R46

Taula 3: Port sèrie

Mòdul LCD: El mòdul LCD de dos línies s'ha de connectar al connector J8.

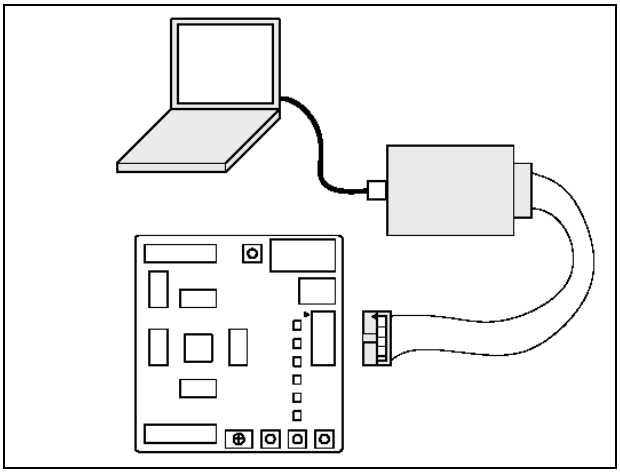


Il·lustració 15: Diagrama de blocs del R8C/23

En aquest diagrama de blocs es representa la connexió entre els components de la board i el microcontrolador.

Es pot observar com el potenciòmetre està directament connectat a una entrada del convertidor analògic/digital.

Els led's i el display estan connectats a ports d'entrada/sortida., i els polsadors, a pins IRQ.



Il·lustració 16: Connexió Renesas Starter Kit R8C/23 a PC

Aquesta figura ens indica la forma correcta de connectar els dos elements principals del kit (l'emulador E8 i la board) amb l'ordinador que conté el software de Renesas.

2.2.3. Centraleta CAN automòbil

Il·lustració 17: Penell SEAT amb centraleta CAN

Dintre del Hardwar utilitzat, hem fet servir a mode de comprovació de connexio i interacció del bus CAN, aquesta centraleta del projecte de fi de carrera titulat *monotorització de bus can amb compact_Rio* , que recrea el frontal d'un choche seat ibiza.

La centraleta CAN, envia misatges de l'estat de les llums, intermitents, bateria, diposit de bencina i totes les altres dades de que disposa el panell d'un cotxe.

També pot rebre misatges que si coincideixen amb els establerts a la centraleta, fa que puguis interactuar amb ella.

2.2.4. Tarja PCMCIA CAN

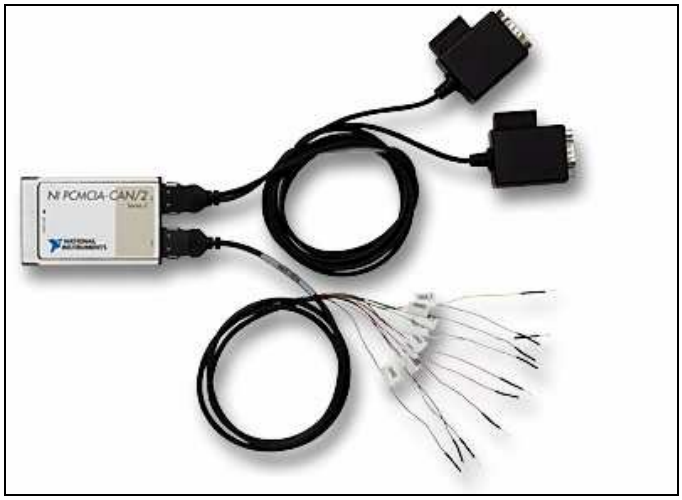
La tarja PCMCIA CAN es la que ens permet que l'ordinador reconegui les dades enviades pel bus CAN de la placa R8C/23 [11],[18]. Les característiques de la tarja són:

Per la tarja PCMCIA-CAN, la capa física es implementada dins del cable.

Existeixen tres tipus de capes físiques disponibles per la tarja PCMCIA-CAN, són:

- High-Speed.
- Low-Speed/Fault-Tolerant.
- Single Wire.

Els cables High-Speed i Low-Speed/Fault-Tolerant, s'alimenten internament mitjançant un convertidor de DC-DC de la board. El cable Single Wire ha de ser alimentat externament, fent servir el bus de CAN.



Il·lustració 18: PCMCIA-CAN/2 Series 2

1. PCMCIA-CAN High-speed cables:

La capa física d'alta velocitat PCMCIA-CAN es alimentada internament (des d'una tarja a través d'un convertidor DC-DC), esta òpticament aïllada fins a 500Vdc (resisteix un màxim de 2 segons) del canal del bus. Aquest aïllament protegeix el NICAN hardware i el PC on esta instal·lat, de ser danyat per pics de sobre tensió.

Transceiver:

PCMCIA-CAN High-speed hardware fa servir el transceiver TJA1041 High-speed CAN de Philips. El chip TJA1041 es completament compatible amb l'estàndard ISO 11898 i esta preparat per un baud rate de fins a 1 Mbps. Aquest aparell també esta preparat per gestionar el consum d'energia a través del sleep mode.

Bus power Requirements:

Degut a que la capa física de la PCMCIA-CAN High-speed s'alimenta completament de forma interna, no te cap necessitat d'una font d'energia externa.

La senyal V- serveix de referència de massa pels senyals aïllats.

2. PCMCIA-CAN Low-speed/Fault-Tolerant Cables

La capa física de la PCMCIA-CAN/LS (cable) es alimentada internament (des d'una tarja a través d'un convertidor DC-DC), esta òpticament aïllada fins a 500Vdc (resisteix un màxim de 2 segons) del canal del bus. Aquest aïllament protegeix el NI CAN hardware i el PC on esta instal·lat, de ser danyat per pics de sobre tensió.

Transceiver:

PCMCIA-CAN low-speed/Fault-Tolerant hardware fa servir el chip TJA1054A de Philips, aquest chip es completament compatible amb l'estàndard ISO 11898 i esta preparat per un baud rate de fins a 125 kbps. El transceiver pot detectar errors i automàticament arreglar-los. Hi ha diversos tipus d'errors:

- Interrupció del cable CAN_H.
- Interrupció del cable CAN_L.
- Curt-circuit de CAN_H a la bateria.
- Curt-circuit de CAN_L a la bateria.
- Curt-circuit de CAN_H a la VCC.
- Curt-circuit de CAN_L a la VCC.
- Curt-circuit de CAN_H a massa.
- Curt-circuit de CAN_L a massa.
- Curt-circuit múltiple entre CAN_H i CAN_L.

Bus power Requirements:

La capa física de la PCMCIA-CAN/LS es alimentada internament hi ja no necessita font d'alimentació del bus. La senyal V- serveix com a referència a massa per l'aïllament de senyals.

3. PCMCIA-CAN Single Wire Cables

La capa física d'alta velocitat PCMCIA-CAN es alimentada internament (des de una tarja a través d'un convertidor DC-DC), esta òpticament aïllada fins a 500Vdc (resisteix un màxim de 2 segons) del canal del bus. Aquest aïllament protegeix el NICAN hardware i el PC on esta instal·lat, de ser danyat per pics de sobre tensió.

Transceiver:

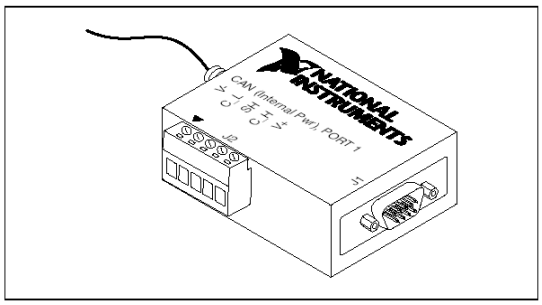
PCMCIA-CAN Single Wire hardware fa servir el transceiver AU5790 de Philips. El chip AU5790 esta preparat per un baud rate de fins a 33.3 Kbps en una transmissió normal i 83.3Kbps en una transmissió d'alta velocitat. El baut rate que es pot aconseguir, depèn en primer lloc de la ret de treball. Cada cable del bus local ha de tenir una resistència de carrega entre els pins CAN_H i RTH del transceiver per proporcionar protecció contra les pèrdues a massa. Aquest aparell també esta preparat per gestionar el consum d'energia a través del sleep mode.

Bus power Requirements:

La capa física del Single Wire requereix una energia externa del bus per proveir els nivells necessaris de senyal per utilitzar tots els modes d'operació. Es necessari proporcionar energia a la senyal V+ mitjançant una font de voltatge en continua de entre 8 i 18V, s'aconsella 12V. El corrent típic es de 40mA i el màxim de 90mA.

PCMCIA-CAN Conector Pinout:

La PCMCIA-CAN te dos tipus de sortida, una de 9 pins mascle D-SUB i Cobicon-style amb 5 pins femella, ajustades amb cargol. Totes les senyals dels 5 pins son connectades als 5 pins corresponents del connector D-SUB de 9 pins.



Il·lustració 19: PCMCIA-CAN Cables

D-SUB Pin	Combinació de pins	Senyal	Descripció
1	-	Cap conexió	-
2	2	CAN_L	Linea del bus CAN_H
3	1	V-	Referencia a massa de CAN
4	-	Cap conexió	-
5	3	*(Shield)*	Shield CAN opcional
6	-	(V-)	Referencia Opcional a massa de CAN
7	4	CAN_H	Linea del bus CAN_H
8	-	Cap conexió	-
9	5	Cap conexió	-

Taula 4: Pins del connector PCMCIA-CAN

CAN_H i CAN_L son les línies de senyal que transporten les dades, aquests cables han d'estar trenats per evitar els camps magnètics.

V- serveix com a referència a massa del CAN_H i CAN_L.

Cabling Requirements for Hight-speed CAN:

Els cables han de tenir les característiques físiques requerides en l'ISO 11898, mostrada en la taula següent:

Característiques	Valor
Impedáncia	Mínim 108Ω, nominal 120Ω, màxim 132 Ω
Resistència Length-related	Nominal 70mΩ/m
Retrart específic de línia	Nominal 5ns/m

Taula 5: Especificacions dels canals L i H

Cable Lenght:

La llargària permesa del cable es afectada per les característiques del cable i la transmissió realitzada.

ISO 11898 especifica que 40m de llargada de cable per un baut rate d'1Mb/s.

La següent taula mostra la relació entre la llargada del cable i el bit Rate:

Bit Rate	Cable gruixut	Cable prim
500 Kb/s	100 m	100 m
250 Kb/s	200 m	100 m
100 Kb/s	500 m	100 m

Taula 6: Mides permeses

Number of devices:

El màxim numero d'aparells depèn de les característiques elèctriques i dels aparell connectats a la ret de treball. Si tots els aparells compleixen la ISO 11898, almenys 30 aparells poden ser connectats al bus.

Si les característiques elèctriques no degraden la qualitat de senyal per sota del nivell de senyal especificat en ISO 11898, poden ser connectats més aparells.

Si tots els aparells de la ret de treball compleixen les especificacions del Device Net podem connectar fins a 64 aparells a la ret.

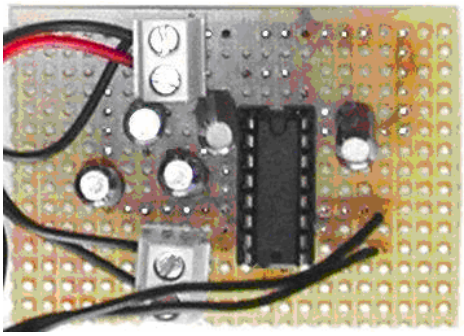
2.2.5. Desenvolupament Hardware a mida

Per tal de complir les necessitats del projecte hem tingut que crear el nostre propi hardware, aquest s'explica amb detall a l'apartat Desenvolupament del Hardware.

El primer que hem necessitat, ha estat una placa convertidora de nivell que reguli els diferents voltatges entre les I/O del port sèrie de la placa Renesas Starter Kit R8C/11 i el port COM de l'ordinador.

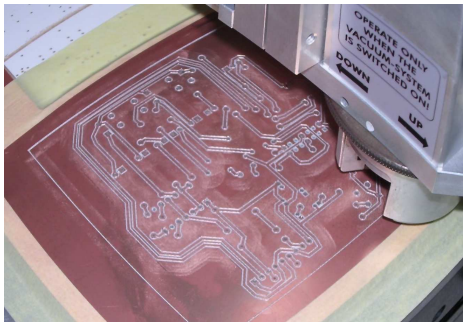
Il·lustració 20: Organigrama placa convertidora de nivell

Per fer les proves de comunicació sèrie amb la primera placa 3D Starter Kit, tenim que implementar el circuit del transceiver RS232 [12]. Aquest circuit el que farà serà adaptar els nivells de voltatge de la placa de prova a les de l'ordinador.



Il·lustració 21: Transceiver RS232

El segon Hardware que creem, es per englobar tot el projecte en una única placa, aquesta es basa en el microcontrolador Renesas R5F212237JFP (D),de la família R8C/23.



Il·lustració 22: Placa pròpia

Com a principals característiques que ens interessa implementar els circuits del convertidor analògic/digital, el del CAN i el del RS232. Encara que també afegim polsadors i led's de proves.

Per fabricar la PCB hem fet servir la fresadora de l'aula de projectistes, LPKF ProtoMat C30, per utilitzar-la hem llegit el manual d'usuari [16].

Capítol 3:

PROTOCOLS DE COMUNICACIÓ

3. Protocols de comunicació

Per comunicar el microcontrolador amb l'ordinador, cal seguir uns protocols de comunicació ja establerts.

Els dos Protocols que fem servir, son molt diferents, tot i així veiem que depenent la utilització que li volem donar tots dos poden ser igualment útils. Les qüestions de velocitat, espai disponible, qualitat de comunicació o del pressupost que disposem, son las que faran que escollim un o l'altre.

3.1. CAN

3.1.1. Introducció al bus CAN

CAN es un protocol de comunicacions desenvolupat per la firma alemanya Robert Bosch GmbH, basat en una topologia bus per la transmissió de missatges en ambients distribuïts, a més ofereix una solució a la gestió de la comunicació entre múltiples unitats centrals de procés [19], [20], [21], [22], [23].

El protocol de comunicacions CAN proporciona els següents beneficis:

- Es un protocol de comunicacions normalitzat , amb el que es simplifica i economitza la tasca de comunicar subsistemes de diferents fabricants sobre una ret comú o bus.

- El processador amfitrió (host) delega la carga de comunicacions a un perifèric intel·ligent, per lo tant el processador amfitrió disposa de major temps per executar les seves pròpies tasques.

- Al ser una ret multiplexada, redueix considerablement el cablejat i elimina les connexions punt a punt.

Per simplificar encara mes l'electrònica del cotxe es pot utilitzar una subret més simple, que es connecta a la ret CAN, anomenada LIN.

3.1.2. Principals característiques del bus CAN

Can es basa en el model client/servidor, amb el concepte de comunicacions de dades, que descriu una relació entre un productor i un o més consumidors. CAN es un protocol orientat a missatges, els quals se'ls assigna un identificador únic dintre de la ret, amb el qual decideixen acceptar o no aquest missatge. Dintre de les seus principals característiques es troben:

- Prioritat de missatges.

- Garantia de temps de latència.

- Flexibilitat en la comunicació.

- Recepcions múltiples (multicast) amb sincronització de temps.

- Sistema robust en relació a la consistència de dades.

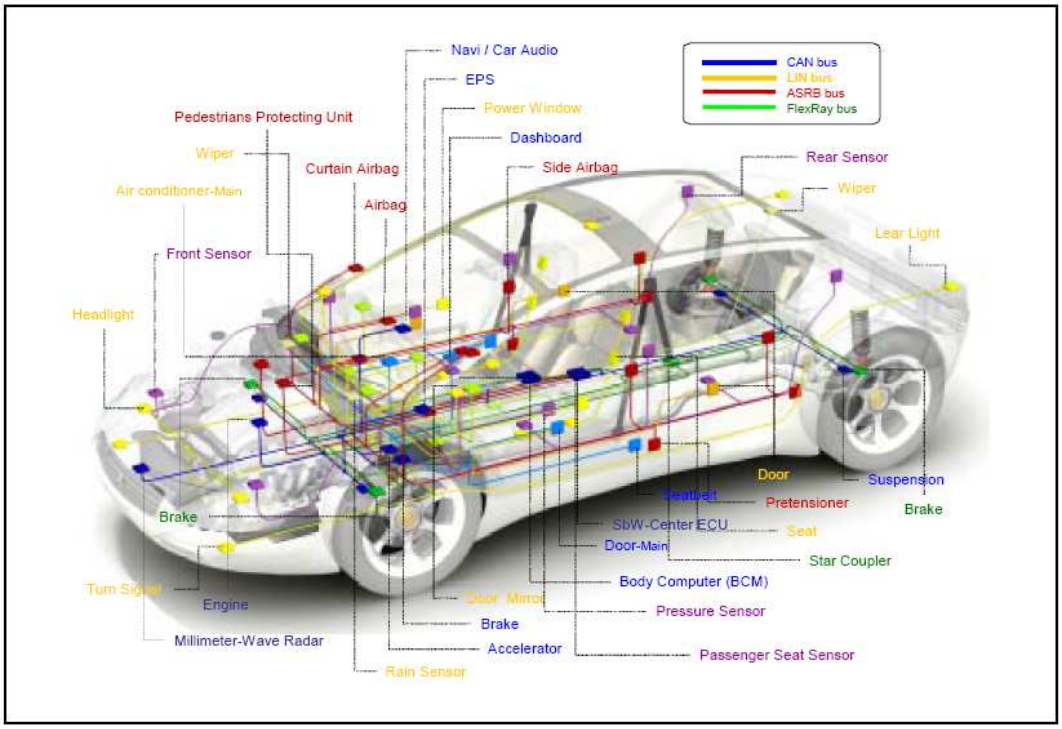
- Sistema múltiples mestres.

- Detecció i senyalització d'errors.

- Retransmissió automàtica de trames errònies.

▪ Distinció entre errors temporals i errades permanents dels nodes de ret, i desconnexió autònoma de nodes defectuosos.

CAN va ser desenvolupat , inicialment per aplicacions en els automòbils, i per tant la plataforma del protocol es resultat de les necessitats existents en l'àrea de l'automació. La Organització internacional per la estandardització (ISO, international Organization for Standarization) defineix dos tipus de rets CAN: una ret d'alta velocitat (fins a 1 Mbps), baix l'estendard ISO (ECU); i una ret de baixa velocitat tolerant a errors (menor o igual 125Kbps), sota l'estendard ISO 11519-2/ISO 11898-3, dedicada a la comunicació de dispositius electrònics interns d'un automòbil com son el control de portes, sostre corredissos, llums i seients.



Il·lustració 23: Aplicació de CAN al sector automobilístic

Com es pot veure a la imatge el CAN es fa servir al sector automobilístic combinat amb altres sistemes de comunicació, entre ells el LIN, que es prou semblant pro més econòmic i no tan veloç.

3.1.3. Protocol de comunicacions CAN

CAN es un protocol de comunicacions sèrie que suporta control distribuït en temps real amb un alt nivell de seguretat i multiplexació.

L'establiment d'una ret CAN per connectar entre si els dispositius electrònics interns d'un cotxe te la finalitat de substituir o eliminar cablejat. Las ECU's(enginyeria control unitat), sensors, sistemes contra lliscaments, etc. Es connecten mitjançant una ret CAN a velocitats de transferència de fins a 1Mbps.

D'acord al model de referència OSI (Open Systems Interconnection), l'arquitectura dels protocols CAN inclou tres capes, física, d'enllaç de dades i aplicació, a més d'una capa especial per al transferència en alta i baixa velocitat.

3.1.3.1. Capa Física

Defineix els aspectes del model físic per la transmissió de dades entre nodes d'una ret CAN, els mes importants son nivells de senyal, representació, sincronització i temps en que els bits es transfereixen al bus.

L'especificació del protocol CAN no defineix una capa física, tot i això, els estàndards ISO 11898 estableixen les característiques que deuen complir les aplicacions per la transferència en alta i baixa velocitat.

3.1.3.2. Capa d'enllaç de dades

Defineix les tasques independents del mètode d'accés al medi, a més ja que una ret CAN dona suport pel processament en temps real a tots els sistemes que l'integren, el intercanvi de missatges que demana l'esmenta't processament, requereix un sistema de transmissió a freqüències altes i retards mínims. En rets amb múltiples mestres, la tècnica d'accés al mitja es molt important ja que tot node actiu te drets per controlar la ret i acaparar els recursos. Per tant la capa d'enllaç de dades defineix el mètode d'accés al medi, així com els tipus de trames per l'enviament de missatges.

Quan un node necessita enviar informació a través d'una ret CAN, pot passar que diversos nodes intentin transmetre simultàniament. CAN ho resol assignant prioritats mitjançant el identificador de cada missatge, on la esmentada assignació es realitza durant el disseny en forma de números binaris i no es pot modificar dinàmicament. El identificador amb el menor número binari es el que te major prioritat.

El mètode d'accés al medi utilitzat es l'accés múltiple per detecció de portador, amb detecció de col·lisions i arbitratge per prioritat de missatges (CSMA/CD+AMP). D'acord amb aquest mètode, els nodes en la ret que necessiten transmetre informació han d'esperar a que el bus estigui lliure(detecció de portador); quant es compleix aquesta condició, aquests nodes modifiquen un bit d'inici (accés múltiple). Cada node llegeix el bus bit a bit durant la transmissió de la trama i comparen el valor transmès amb el valor rebut, si es detecta una diferencia en els valors de bits, es porta a cap un mecanisme d'arbitratge.

CAN estableix dos formes de trames de dades (data frame) que difereixen en la longitud del camp identificador, les trames estàndard (standard frame) amb un identificador d'11 bits definides en l'especificació CAN 2.0A, i les trames extenses (exttended frame) amb un identificador de 29 bits definides en l'especificació CAN 2.0B

Per la transmissió i control de missatges CAN, es defineixen quatre tipus de trames: de dades, remota (remote frame), d'error (error frame) i sobrecarrega (overload frames). Les trames remotes també s'estableixen en els dos formats, estàndard i extenses, i tan les trames de dades com les remotes es separen de trames precedents mitjançant espais entre trames (interframe space).

En quan a la detecció y control d'errors, un controlador CAN te la capacitat de detectar i manipular els errors que sorgeixen en una ret. Tot error detectat en un node es notifica a la resta de nodes.

3.1.3.3. Capa de supervisor

La substitució del cablejat convencional per un sistema de bus sèrie presenta el següent inconvenient, un node defectuós pot bloquejar el funcionament del sistema al complet. Cada node actiu transmet una bandera d'error quan detecta un error i pot ocasionar que un node defectuós acapari el medi físic. Per eliminar aquest risc el protocol CAN defineix un mecanisme autònom per detectar i desconnectar un node defectuós del bus, l'esmenta't mecanisme es coneix com aïllament d'errades.

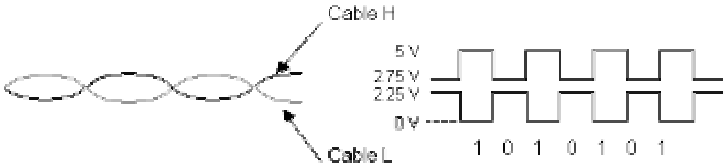
3.1.3.4. Capa de aplicació

Existeixen diferents estendards que defineixen la capa d'aplicació, alguns son molt específics i estan relacionats amb els seus camps d'aplicació. Entre les capes d'aplicació més utilitzades s'ha de mencionar: CAL, CANopen, DeviceNet, SDS(Smart Distributed System), OSEK, CANKingdom.

3.1.3.5. Estructura d'un bus CAN

3.1.3.5.1. CABLES

La informació circula per dos cables trenats que uneixen totes les unitats de control que formen el sistema. Aquesta informació es transmet per diferencia de tensió entre dos cables, de forma que un valor alt de tensió representa un 1 i un valor baix de tensió representa un 0. la combinació adequada d'uns i zeros conformen el missatge a transmetre.



Il·lustració 24: Cable de comunicació CAN

En un cable els valors de tensió oscil·len entre 0V i 2.25V, pel que es denomina cable L(Low) i l'altre, el cable H(High) ho fa entre 2.75 i 5V. En cas de que s'interrompi la línia H o que es derivi a massa, el sistema treballarà amb la senyal de low amb respecte a massa, en cas que s'interrompi la línia L, passarà el contrari. Aquesta situació permet que el sistema segueixi treballant amb un dels cables tallats o comunicats a massa, inclòs amb els dos comunicats seria possible el funcionament, quedant fora de servei solsament quan els dos cables es tallen.

Es important tenir en compte que el trenat entre les dues línies serveix per anular els camps magnètics, per lo que no s'han de modificar en cap cas ni el pes ni la longitud dels esmentats cables.

3.1.3.5.2. ELEMENT DE TANCAMENT

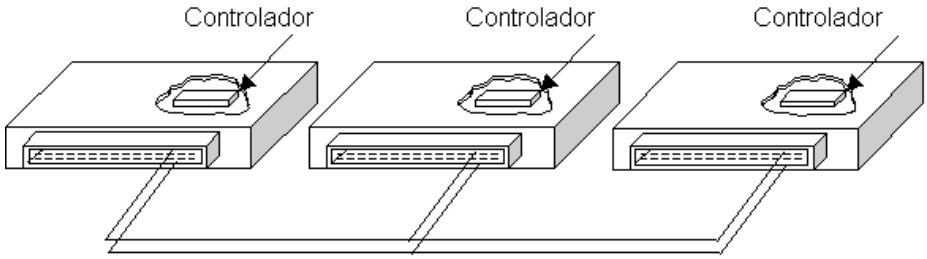
Son resistències connectades als extrems dels cables H i L. Els seus valors s'obtenen de forma empírica i permeten adequar el funcionament del sistema a diferents longituds de cables i número de unitats de control abonades, ja que impedeixen fenòmens de reflexió que poden pertorbar el missatge.

Aquestes resistències estan allotjades en el interior d'algunes unitats de control del sistema per qüestions d'economia i seguretat del funcionament.

3.1.3.5.3. CONTROLADOR

Es l'element encarregat de la comunicació entre el microprocessador de la unitat de control i el transmissor i o receptor. Treballa condicionant la informació que entra i surt entre els dos components.

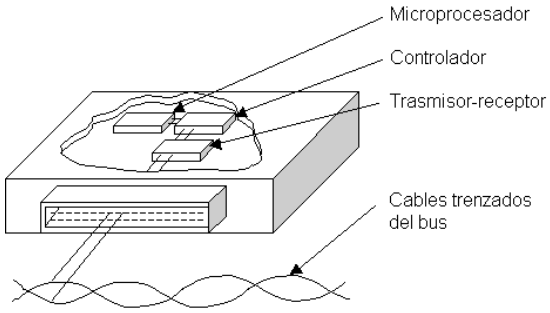
El controlador esta situat a la unitat de control, pel que existeix tants com unitats estiguin connectades al sistema. Aquest element treballa amb nivells de tensió molt baixos i es el que determina la velocitat de transmissió dels missatges, que serà mes o menys elevada segons el compromís del sistema. Així, es pot fer que els més prioritaris es transmetin a 500Kbaudis, i els que no tenen una importància significant a 62.5Kbaudis. Aquest element també intervé en la necessària sincronització entre les diferents unitats de comandament per la correcta emissió i recepció dels missatges.



3.1.3.5.4. TRANSMISSOR/RECEPTOR

El transmissor/receptor es l'element que te la missió de rebre i transmetre dades, a mes de condicionar i preparar la informació per que pugui ser utilitzada pels controladors. Aquesta preparació consisteix en situar els nivells de tensió de forma adequada, amplificant la senyal quan la informació es posa en línia i reduint-la quan es recollida.

El transmissor/receptor es bàsicament un circuit integrat que esta situat en cada una de les unitats de control del sistema, treballa amb intensitats pròximes a 0.5A i en cap cas intervé modificant el contingut del missatge. Funcionalment esta situat entre els cables que formen la línia bus CAN i el controlador.



3.1.3.5.5. FUNCIONAMENT DEL BUS CAN

Les unitats de comandament que es connecten al sistema bus CAN son les que necessiten compartir informació, pertanyi o no al mateix sistema. En l'automoció generalment estan connectades a una línia les unitats de control del motor, del ABS i del canvi automàtic, i a l'altre línia (de menor velocitat) les unitats de control relacionades amb el sistema confort.

El sistema bus CAN esta orientat al missatge i no al destinatari. La informació en la línia es transmesa en forma de missatges estructurats, on una part del mateix es un identificador que indica la classe de dada que conte. Totes les unitats de control reben el missatge, el filtren i només el fan servir els que necessiten la dada esmentada. Quan el bus no te cap unitat connectada pot començar a transmetre un nou missatge.

En cas que una o varies unitats vulguin introduir un missatge al mateix temps, ho farà la que tingui major prioritat. Aquesta prioritat esta indicada en el identificador.

El procés de transmissió de dades es desenvolupa seguint un cicle de varies fases:

Subministrament de dades: Una unitat de comandament rep informació dels sensors que te associats (velocitat, temperatura, humitat, proximitat, etc.)

El microprocessador passa la informació al controlador on es gestiona i condiciona, i a la vegada ser passat al transmissor/receptor on es transforma en senyals elèctriques.

Transmissió de dades: El controlador d'aquesta unitat transfereix les dades i el seu identificador junt amb la petició d'inici de transmissió, assumint la responsabilitat que el missatge sigui correctament transmès a totes les unitats de comandament associades. Per transmetre el missatge s'ha tingut que trobar el bus lliure, i en cas de col·lisió amb una altre unitat de comandament intentant transmetre simultàniament, tenir una prioritat mes alta. A partir del moment en que això passa, la resta d'unitats de comandament es converteix en receptor.

Recepció del missatge: Quant la totes les unitats del comandament reben el missatge, verifiquen el identificador correspongui amb els que poden admetre. Si no es part d'aquests, l'ignoren.

El sistema bus CAN disposa de mecanismes per detectar errors en la transmissió de missatges, de forma que tots els receptors realitzen una comprovació de missatges analitzen una part del mateix, anomenat camp CRC. Altres mecanismes de control s'apliquen en les unitats emissores que monitoritzen el nivell de bus, la presencia de camps de format fix en el missatge (verificació de trama), l'anàlisi i estadístiques per part de les unitats de comandament dels seus propis errors.

Aquestes mesures fan que les probabilitats d'error en la emissió i recepció de missatges siguin baixes, i fa que sigui un sistema extraordinàriament segur.

El plantejament del bus CAN, permet disminuir notablement el cablejat, posat que si una unitat de comandament disposa d'una informació, com per exemple, la temperatura d'un sensor, aquesta pot ser utilitzada per el resta d'unitats de comandament sense que sigui necessari que cada una d'elles rebi la informació de l'esmentat sensor.

Un altre avantatge, es que les funcions puguin ser repartides entre diferents unitats de comandament, i que incrementar les funciones d'una unitat, no impliqui un cost addicional excessiu.

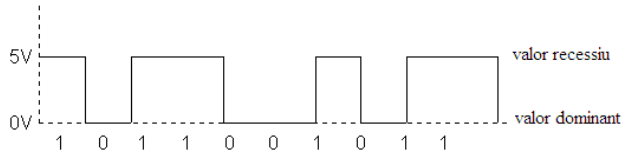
3.1.3.5.6. CARACTERÍSTIQUES D'UN MISSATGE CAN

El missatge es una successió de "0" i "1", que com s'explicava al principi, estan representats per diferents nivells de tensió en els cables del Bus CAN i que s'anomena "bit".

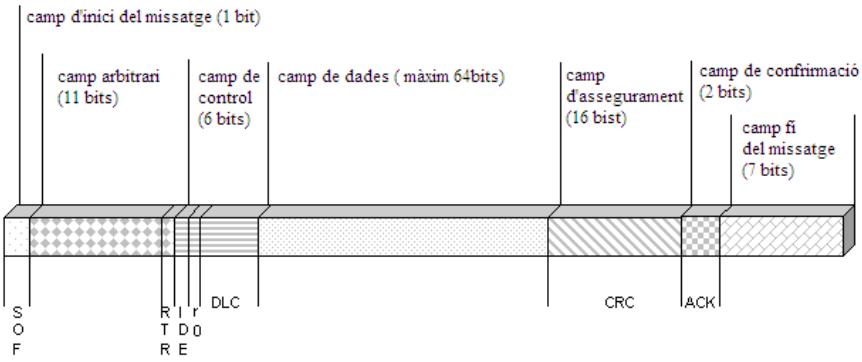
El missatge te una sèrie de camps de diferents longituds (números de bits) que permeten realitzar el procés de comunicació entre les unitats de comandament, segons el protocol definit per Bosch pel Bus CAN, que faciliten des de identificar la unitat de comandament, indicar el principi i el final del missatge, mostrar les dades i permetre diferents controls entre d'altres.

Els missatges son introduïts en la línia com una cadència que oscil·la entre els 7 i els 20 mili segons depenent de la velocitat de l'àrea i la unitat de comandament que els introdueix.

Exemple de com s'escriu un missatge:



Estructura d'un missatge estendard:



Camp d'inici del missatge:

El missatge s'inicia com un bit dominant, del qual, el flac descendent es utilitzat per les unitats de comandament per sincronitzar-se entre si.

Camp arbitrari:

Els 11 bits d'aquest camp s'utilitzen com identificador, que permet reconèixer a les unitats de comandament la prioritat del missatge. Quant més baix es el valor de l'identificador mes alta es la prioritat, i per lo tant determina l'ordre en que seran introduïts els missatges en la línia.

El bit RTR indica si el missatge conte dades (RTR=0) o si es tracta de una trama remota sense dades (RTR=1).Una trama de dades sempre te una prioritat mes alta que una trama remota.

La trama remota s'utilitza per sol·licitar dades a altres unitats de comandament o be perquè es necessita o per realitzar un comprovació.

Camp de control:

Aquest camp informa sobre les característiques del camp de dades. El bit IDE indica quan es un "0" que es tracta d'una trama estàndard i quan es un "1" que es tracta d'una trama extensa. Els quatre bits que composen el camp DLC indiquen el número de bytes continguts en el camp de dades.

La diferencia entre una trama estàndard i una trama extensa es que la primera te 11bits i la segona 29 bits. Les dues trames poden coexistir eventualment, i la raó de la seva presencia es la existència de dos versions de CAN.

Camp de dades:

En aquest camp apareix la informació del missatge amb les dades que la unitat de comandament corresponent introdueix en la línia de Bus CAN. Pot contindré entre 0 i 8 bytes (de 0 a 64bits).

Camp d'assegurament (CRC):

Aquest camp te una longitud de 16 bits i pot ser utilitzat per la detecció d'errors pels 15 primers, mentre que l'últim sempre es un bit recessiu (1) que delimita el camp CRC.

Camp de confirmació (ACK):

El camp ACK esta compost per dos bits que son sempre transmesos com recessius (1).Totes les unitats de comandament que reben el mateix CRC modifiquen el primer bit del camp ACK per un dominant (0), de

manera que la unitat de comandament que esta encara transmetent reconeix com a mínim que alguna unitat de comandament ha rebut un missatge escrit correctament. De no ser així, la unitat de comandament transmissora interpreta que el seu missatge presenta error.

Camp final de missatge (EOF):

Aquest camp indica el final del missatge amb una cadena de 7 bits recessius. Pot passar que en determinats missatges es produeixin llargues cadenes de zeros i uns, i que això provoqui una pèrdua de sincronització entre unitats de comandament.

El protocol CAN resol aquesta situació, introduint un bit de diferent polaritat cada cinc bits iguals: cada cinc "0" s'insereix un "1" i viceversa. La unitat de comandament que utilitza el missatge, descarta un bit posterior a cinc bits iguals. Aquests reben el nom de bit stuffing.

Exemple d'un missatge real:

SOF	IDENTIFICADOR	RTR	DE	DLC	DATO	DATO	CRC	ACK	FN
0	1100010000	0	000	0010	00010110	00000000	0	01	11111

Taula 7: Missatge CAN

3.1.3.5.7. DIAGNÒSTIC DEL BUS CAN

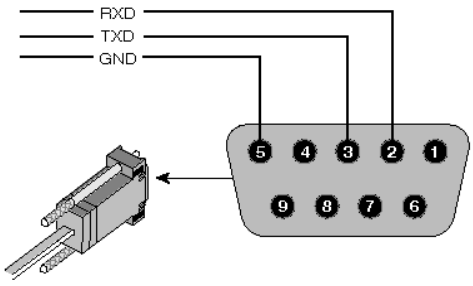
Una possibilitat es fer servir el programa informàtic CANALyzer(Vector Informatik GmbH) amb l'ordenador i la connexió adequada. Aquest programa permet visualitzar el tràfic de dades del bus CAN, indica el contingut dels missatge i realitza l'estadística del missatge, rendiment i errors.

Provablement, l'eina mes adequada i assequible sigui l'oscil·loscopi digital amb dos canals, memòria i un ample de banda de 20MHz, amb el que es poden visualitzar perfectament els missatges utilitzant una base de temps de 100 micro segons i una base de tensió de 5 volts. En aquest cas, s'ha de tenir en compte que els bits stuff (els que s'afegeixen després de cinc bits iguals) han de ser eliminats.

3.2. RS232

RS-232 es una interfaz que designa una norma per l'intercanvi sèrie de dades binàries, la utilització mes comú es entre un DTE (Equip terminal de dades) i un DCE (Equip de terminació del circuit de dades).

El RS-232 consisteix en un connector del tipus DB-9 (de 9 pins) on els pins mínims necessaris per fer la comunicació, son GND, RXD i TXD.

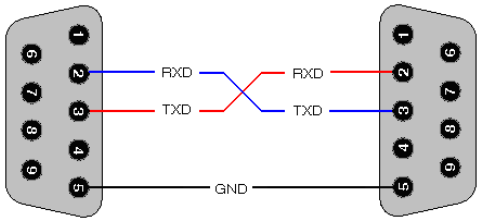


Il·lustració 25: Connector DB:9

Pin	Senyal	In/Out	Descripció
1	DCD	In	Detecció Dades
2	RxD	In	Recepció Dades
3	TxD	Out	Transmissió Dades
4	DTR	Out	Terminal Dades Llest
5	GND	-	Massa
6	DSR	In	Dades Llestes
7	RTS	Out	Petició Per Enviar
8	CTS	In	Neteja De Enviar
9	RI	In	Senyal Indicador

Taula 8: Pins del DB-9

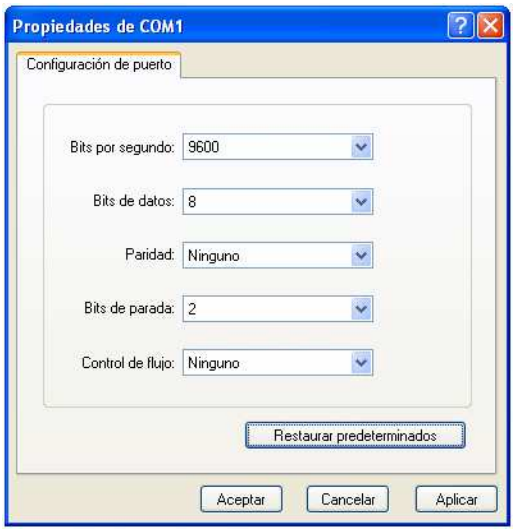
Una interfase *full dúplex* pot abstenir-se solsament amb 3 cables. Per construir el cable, haurem de creuar els pins dos i tres, enviar i rebre, de manera que tots dos puguin enviar i rebre.



Il·lustració 26: Cable de connexió

La interfaz RS-232 està dissenyada per distancies curtes, d'uns 15 metres o menys, i per unes velocitats de comunicació baixes, de no més de 20Kb. S'ha de tenir en compte a l'hora de programar, que la velocitat la fixa el perifèric que llegeix ja que si enviem informació sense tenir-ho en compte, les dades rebudes poden no ser correctes.

Aquesta velocitat anomenada bud rate a de ser configurada tant a l'ordinador com a la UART, juntament amb la paritat, bits de dades, bits de parada i control de flux.



Il·lustració 27: Configuració Hyperterminal

En aquest exemple, es pot veure la configuració amb el programa HyperTerminal del port COM1 de l'ordinador.

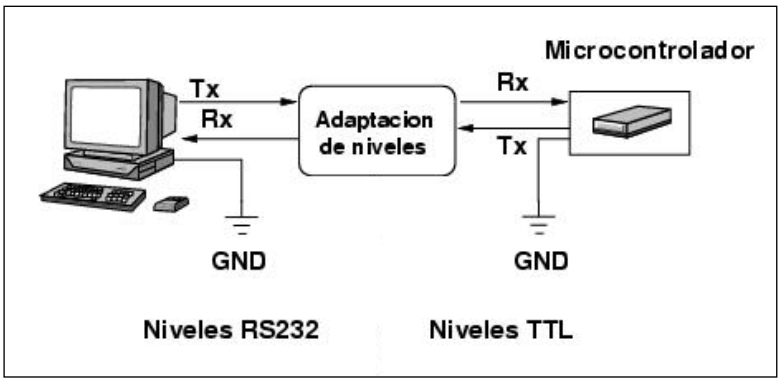
En el programa exemple realitzat amb l'IAR es veurà amb més detall la configuració escollida.

A l'hora de treballar el microcontrolador amb l'ordinador, ens trobem amb la problemàtica que tenen diferents nivells de tensió.

Tecnologia TTL:

TTL (del anglès *Transistor-Transistor Logic*, "Lògica Transistor a Transistor") es una tecnologia de construcció de circuits electrònics digitals, en els que els elements d'entrada a la ret lògica son transistors, així com els elements de sortida del dispositiu. La seva tensió d'alimentació característica es troba entre els 4,75V i els 5,25V. Com es un rang molt petit, els nivells lògics venen definits per el rang de tensió establerts entre 0,2V y 0,8V per l'estat L (baix) i 2,4V i Vcc per l'estat H (alt).

La velocitat de transmissió entre els estats lògics es la seva millor característica, encara que fa que augmenti tant el seu consum que també la converteix en la pitjor.

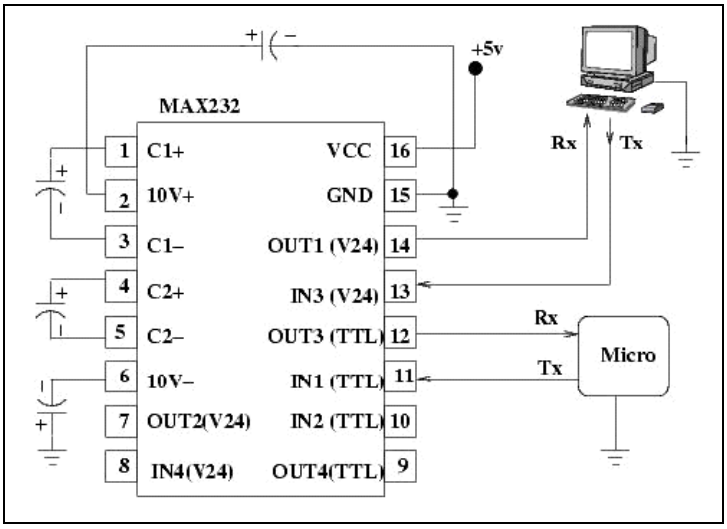


RS-232	TTL
+15V	+5V
-15V	0V

Taula 9: Nivells de Voltatge

Il·lustració 28: Configuració SERIE

Per solucionar la diferència de voltatges d'una tecnologia a l'altre fem servir un "transceiver" 232 que facilita l'adaptació.



Il·lustració 29: Esquema de connexió SERIE

A l'ora de muntar aquest circuit cal mira el Data Sheet per veure els valors dels condensadors [12], [14].

Capítol 4:

µC (R8C RENESAS), INTRODUCCIÓ AL PIC

4. µC (R8C Renesas), Introducció al pic

Renesas es un dels principals fabricants de semiconductors del mon, es una companyia japonesa de Hitachi i Mitsubishi Electric fundada en el 2003.

Dedicada als mercats d'automoció, mòbils, digital, industrials i PC/AV, Renesas te una línia extensa de MCU.

A mode de familiaritzar-nos amb els microcontroladors Renesas i la comunicació via SERIE i CAN hem treballat amb els següents MCU's.

4.1. R8C/11 Group

Dintre de la família R8C/11, el microcontrolador que utilitzem es el R5F21114FP amb una ROM de 16K bytes, una RAM de 1K byte, l'encapsula't es el 32P6U-A amb memòria Flash [5].

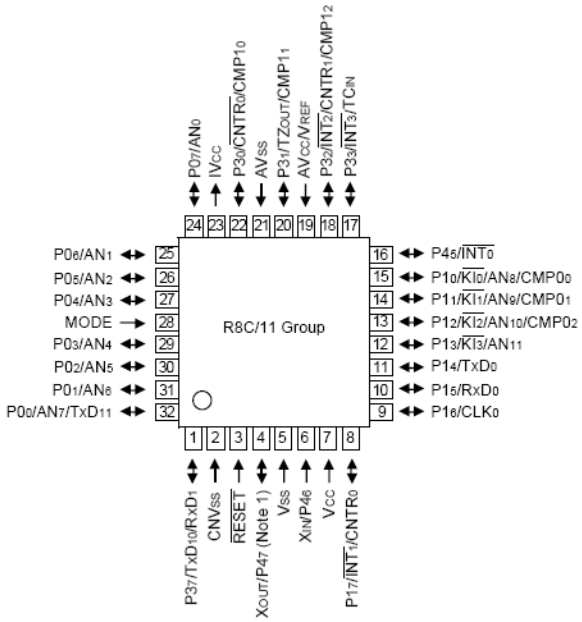
Per tenir una idea més concreta del que es pot fer amb aquest microcontrolador, a la següent taula es troben totes les seves característiques.

Item		Performance
CPU	Number of basic instructions	89 instructions
	Shortest instruction execution time	50 ns (f(XIN) = 20 MHz, VCC = 3.0 to 5.5 V) 100 ns (f(XIN) = 10 MHz, VCC = 2.7 to 5.5 V)
	Operating mode	Single-chip
	Address space	1M bytes
	Memory capacity	See Table 1.2.
Peripheral function	Interrupt	Internal: 11 sources, External: 5 sources, Software: 4 sources, Priority level: 7 levels
	Watchdog timer	15 bits x 1 (with prescaler)
	Timer	Timer X: 8 bits x 1 channel, Timer Y: 8 bits x 1 channel, Timer Z: 8 bits x 1 channel (Each timer equipped with 8-bit prescaler) Timer C: 16 bits x 1 channel Circuits of input capture and output compare.
	Serial I/O	•1 channel Clock synchronous, UART •1 channel UART
	A-D converter	10-bit A-D converter: 1 circuit, 12 channels
	Clock generation circuit	2 circuits •Main clock generation circuit (Equipped with a built-in feedback resistor) •Ring oscillator (high speed, low speed) On High-speed ring oscillator the frequency adjustment function is usable.
	Oscillation stop detection function	Stop detection of main clock oscillation
	Voltage detection circuit	Included
	Power on reset circuit	Included
	Port	Input/Output: 22 (including LED drive port), Input: 2 (LED drive I/O port: 8, max. 20 mA)

Electrical characteristics	Power supply voltage	Vcc = 3.0 to 5.5 V (f(XIN) = 20 MHz) Vcc = 2.7 to 5.5 V (f(XIN) = 10 MHz)
	Power consumption	Typ. 9 mA (Vcc = 5.0 V, (f(XIN) = 20 MHz, High-speed mode) Typ. 5 mA (Vcc = 3.0 V, (f(XIN) = 10 MHz, High-speed mode) Typ. 35 □A (Vcc = 3.0 V, Wait mode, Peripheral clock off) Typ. 0.7 □A (Vcc = 3.0 V, Stop mode)
Flash memory	Program/erase voltage	Vcc = 2.7 to 5.5 V
	Number of program/erase	100 times
Operating ambient temperature		-20 to 85 °C -40 to 85 °C (option)

Taula 10: Línies generals de funcionament del R8C/11

La connexió de les potes del R5F21114FP, es molt important, i s'ha de tenir en compte a l'hora de programar, ja que un mateix pin pot tenir diferents funcions, i depenent de que tinguem connectat, s'haurà de configurar els registres de diferent forma.



Il·lustració 30: Configuració del pins

A mode d'exemple, podem veure que la UART, que son les entrades i sortides de comunicació sèrie, es troben als pins 10(RxD0), 11(TxD0), 1(TxD10/RxD1) i 32(TxD11). Però també veiem que aquests pins tenen altres possibles connexions, per això s'ha de configurar els registres adequadament.

A la següent taula es troben definits tots els noms de les possibles configuracions dels pins, indicant la funció i si son d'entrada o de sortida.

Signal name	Pin name	I/O type	Function
Power supply input	Vcc, Vss	Input	Apply 2.7 V to 5.5 V to the Vcc pin. Apply 0 V to the Vss pin.
IVcc	IVcc	Output	Connect this pin to Vss via a capacitor (0.1 □F).
Analog power supply input	AVcc, AVss	Input	These are power supply input pins for A-D converter. Connect the AVcc pin to Vcc. Connect the AVss pin to Vss. Connect a capacitor between pins AVcc and AVss.
Reset input	$\overline{\text{RESET}}$	Input	"L" on this input resets the MCU.
CNVss	CNVss	Input	Connect this pin to Vss via a resistor.(1)
MODE	MODE	Input	Connect this pin to Vcc via a resistor.
Main clock input	XIN	Input	These pins are provided for the main clock generating circuit input/output. Connect a ceramic resonator or a crystal oscillator between the XIN and XOUT pins. To use an externally derived clock, input it to the XIN pin and leave the XOUT pin open.
Main clock output	XOUT	Output	
$\overline{\text{INT}}$ interrupt input	$\overline{\text{INT0}}$ to $\overline{\text{INT3}}$	Input	These are $\overline{\text{INT}}$ interrupt input pins.
Key input interrupt input	$\overline{\text{KI0}}$ to $\overline{\text{KI3}}$	Input	These are key input interrupt input pins.
Timer X	CNTR0	Input/Output	This is the timer X I/O pin.
	$\overline{\text{CNTR0}}$	Output	This is the timer X output pin.
Timer Y	CNTR1	Input/Output	This is the timer Y I/O pin.
Timer Z	TZOUT	Output	This is the timer Z output pin.
Timer C	TCIN	Input	This is the timer C input pin.
	CMP00 to CMP03, CMP10 to CMP13	Output	These are the timer C output pins.
Serial interface	CLK0	Input/Output	This is a transfer clock I/O pin.
	RxD0, RxD1	Input	These are serial data input pins.
	TxD0, TxD10, TxD11	Output	These are serial data output pins.
Reference voltage input	VREF	Input	This is a reference voltage input pin for A-D converter. Connect the VREF pin to Vcc.
A-D converter	AN0 to AN11	Input	These are analog input pins for A-D converter.
I/O port	P00 to P07, P10 to P17, P30 to P33, P37, P45	Input/Output	These are 8-bit CMOS I/O ports. Each port has an input/output select direction register, allowing each pin in that port to be directed for input or output individually. Any port set to input can select whether to use a pull-up resistor or not by program. P10 to P17 also function as LED drive ports.
Input port	P46, P47	Input	These are input only pins.

Taula 11: Descripció dels pins

Entre els que fem servir, es troba l'A-D converter, connectat a un potenciòmetre, te dotze canals d'entrada, dels en fem servir AN0.

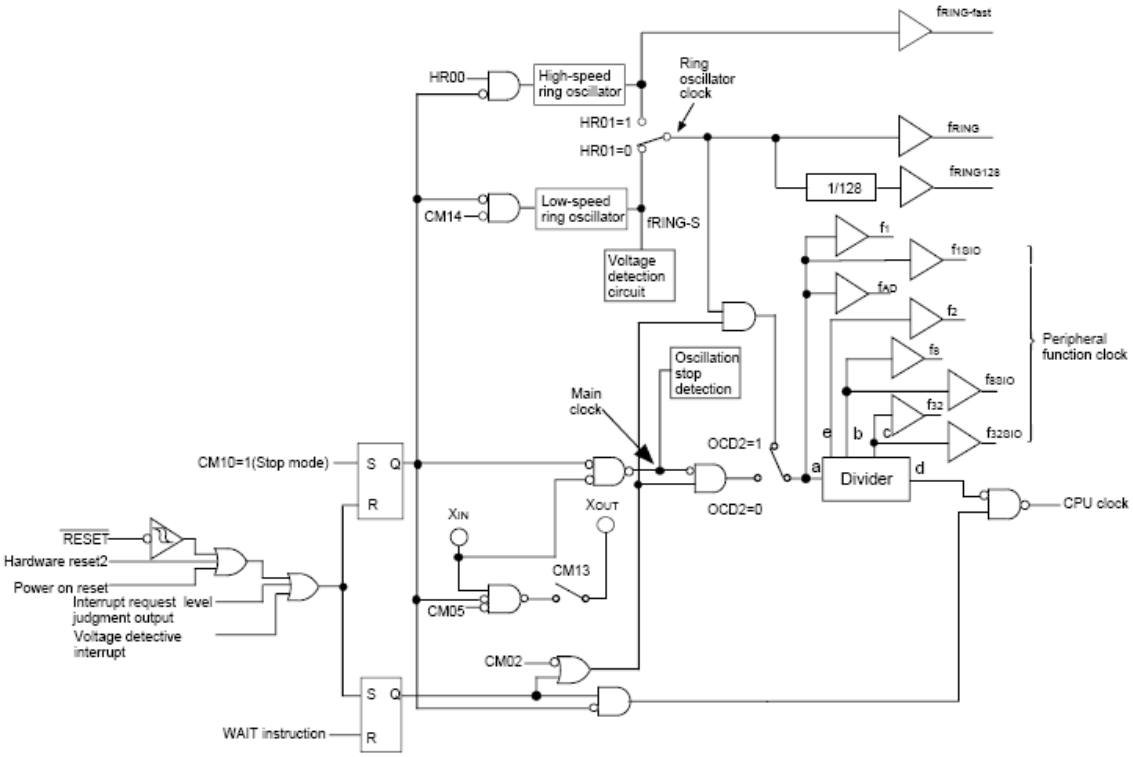
Els Timers que utilitzarem seran el TimerY i el TimerZ, per el primer utilitzem l'oscilador extern de 20MHz i per el TimerZ el valor obtingut del TimerY.

El Serial interface, també anomenat UART, es on es configuren els ports d'entrada i sortida del port SERIE RxD i TxD. Es poden fer servir dos ports diferents, nosaltres treballarem amb la UART0.

Per els I/O ports connectem els Led's indicadors.

4.1.1. Clock

La selecció del clock es important ja que determinarà la velocitat de treball per tot el programa.



Il·lustració 31: Circuit de generació del clock

L'oscil·lador permet treballar sense un cristall extern, es pot seleccionar a velocitat lenta 125KHz o ràpida 8MHz, variant les fonts des de f1 fins a f32 ajustarem aquests valors per que compleixin les nostres necessitats.

En el nostre cas treballarem amb un cristall extern de 20MHz.

Per configurar el clock amb el que vols treballar, has de seguir el diagrama modificant els registres necessaris.

Exemple per treballar amb un cristall extern:

CM02= 0; CM05= 0; CM06= 0; CM10= 0; CM13= 1; CM14= 0; CM15= 1; CM16= 0; CM17= 0;

4.1.2. Timers

El microcontrolador, té tres Timers de 8 bits i un de 16 bits. Els de 8 bits son el Timer X, Timer Y i Timer Z, i cada un te un prescaler de 8 bits. El Timer de 16 bits es el Timer C i té captura d'entrada i comparació de sortida. Tots aquests comptadors de temps funcionen independentment.

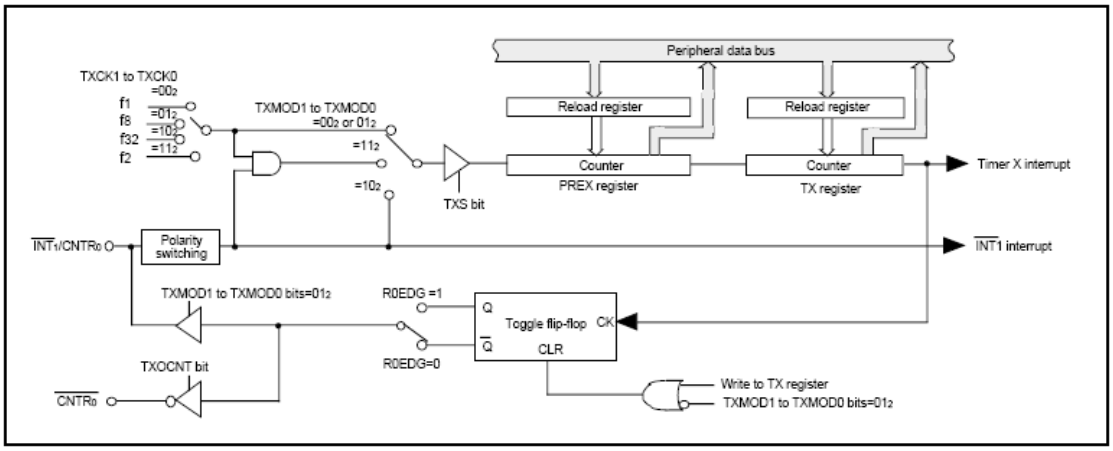
Item		Timer X	Timer Y	Timer Z	Timer C
Configuration		8-bit timer with 8-bit prescaler	8-bit timer with 8-bit prescaler	8-bit timer with 8-bit prescaler	16-bit free-run timer
Count		Down	Down	Down	Up
Count source		•f_1 •f_2 •f_8 •f_{32}	•f_1 •f_8 •f_{RING} •Input from $CNTR_1$ pin	•f_1 •f_2 •f_8 •Timer Y underflow	•f_1 •f_8 •f_{32} •$f_{RING-fast}$
Function	Timer mode	provided	provided	provided	not provided
	Pulse output mode	provided	not provided	not provided	not provided
	Event counter mode	provided	provided[1]	not provided	not provided
	Pulse width measurement mode	provided	not provided	not provided	not provided
	Pulse period measurement mode	provided	not provided	not provided	not provided
	Programmable waveform generation mode	not provided	provided	provided	not provided
	Programmable one-shot generation mode	not provided	not provided	provided	not provided
	Programmable wait one-shot generation mode	not provided	not provided	provided	not provided
	Input capture mode	not provided	not provided	not provided	provided
	Output compare mode	not provided	not provided	not provided	provided
Input pin		$CNTR_0$	$CNTR_1$	$\overline{INT_0}$	TC_{IN}
Output pin		$CNTR_0$ $\overline{CNTR_0}$	$CNTR_1$	TZ_{OUT}	$CMP0_0$ to $CMP0_2$ $CMP1_0$ to $CMP1_2$
Related interrupt		Timer X int $\overline{INT}1$ int	Timer Y int $\overline{INT}2$ int	Timer Z int $\overline{INT}0$ int	Timer C int $\overline{INT}3$ int compare 0 int compare 1 int
Timer stop		provided	provided	provided	provided

Taula 12: Timers

Els diagrames de blocs ens donen tota la informació que necessitem per programar els Timers.

El temporitzador X té cinc modes d'operació:

- Mode de temporitzador: El temporitzador utilitza una font interna de compte (font de rellotges).
- Mode de producció d'impuls: El temporitzador fa servir una font de compte interna i treu els impulsos amb la polaritat inversa del temporitzador.
- Mode de taulell d'esdeveniment: El temporitzador compta impulsos externs.
- Mode de càlcul d'amplada d'impuls: El temporitzador mesura l'amplada d'impuls d'un impuls extern.
- El temporitzador de mode: Mesura el període d'un impuls extern.



Il·lustració 32: Timer X

Exemple per aconseguir 5 segons:

El temporitzador X treballa amb l'oscil·lador intem, que depenent la configuració pot ser de 8MHz o de 125KHz, les fonts de correcció que se li poden aplicar son f1, f2, f8 o f32.

Seleccionem velocitat baixa de 125KHz i f32.

$$\text{rellotge intern a velocitat baixa} \cdot f32 = \frac{125000}{32} = 3906{,}25$$

$$\text{TimerX} = \frac{(PREX+1)\cdot(TX+1)}{\text{rellotge intern}} = 5 \text{ segons}$$

$$(PREX+1)\cdot(TX+1) = 3906{,}25 \cdot 5 = 19531{,}25$$

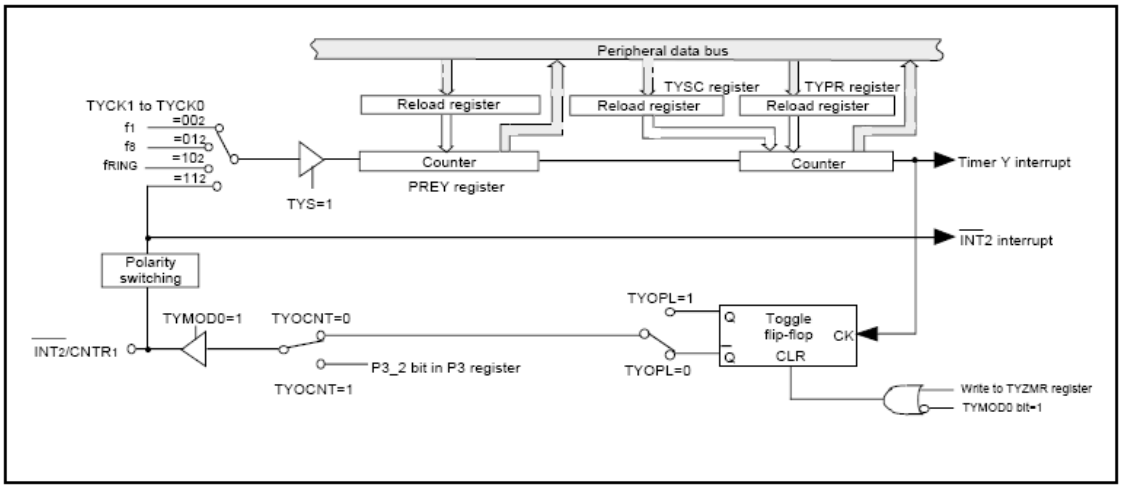
Si fixem PREX+1 a 100, TX+1 serà 195,3125

Al no sortir exacte aurem d'escollir entre $TX = \frac{194}{195}$

$$\text{TimerX} = \frac{(PREX+1)\cdot(TX+1)}{\text{rellotge intern}} = \frac{100\cdot 195}{3906{,}25} = 4{,}992 \text{ segons}$$

$$\text{TimerX} = \frac{(PREX+1)\cdot(TX+1)}{\text{rellotge intern}} = \frac{100\cdot 196}{3906{,}25} = 5{,}0176 \text{ segons}$$

Per configurar el Timer Y es fan servir els registres TYZMR, PREY, TYSC, TYPR, TYZOC, PUM, i YCSS.



Il·lustració 33: Timer Y

Exemple per aconseguir 0,02 segons:

El temporitzador Y treballa amb l'oscil·lador extem, la placa en té un de 20MHz, les fonts de correcció que se li poden aplicar son f1 f8 o fRING.

Seleccionem 20MHz i f8.

$$\text{Fext} = \text{rellotge extern} \cdot \text{f8} = \frac{20000000}{8} = 2.5\text{MHz}$$

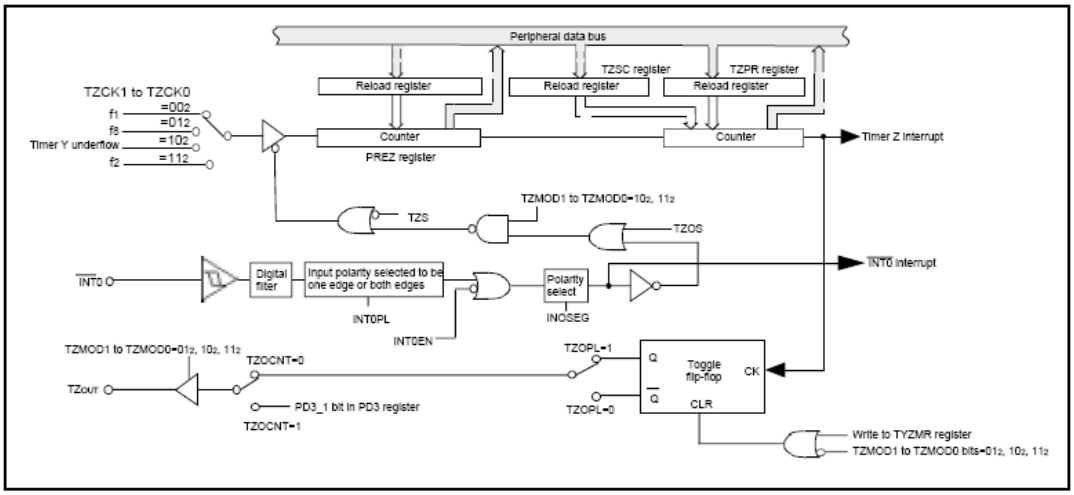
$$\text{TimerY} = \frac{(\text{PREY}+1)\cdot(\text{TYPR}+1)}{\text{Fext}} = \frac{250\cdot 200}{2500000} = 0.02 \text{ segons}$$

PREY = 249 i TYPR = 199

El temporitzador Z té quatre modes d'operació:

- Mode de temporitzador: El temporitzador utilitza una font interna de compte (font de rellotges) o la base de temps del Timer Y.
- Mode de generació de waveform programable: El temporitzador imprimeix impulsos d'una amplada donada successivament.
- Mode degeneració d'un dispar programable: Impuls a la sortida del temporitzador.
- Mode de generació d'una espera de dispar programable: Les sortides de temporitzador retardaven l'impuls de dispar.

Per configurar el Timer Z es fan servir els registres TYZMR, PREZ, TZSC, TZPR, TYZOC, PUM, i TCSS.



Il·lustració 34: Timer Z

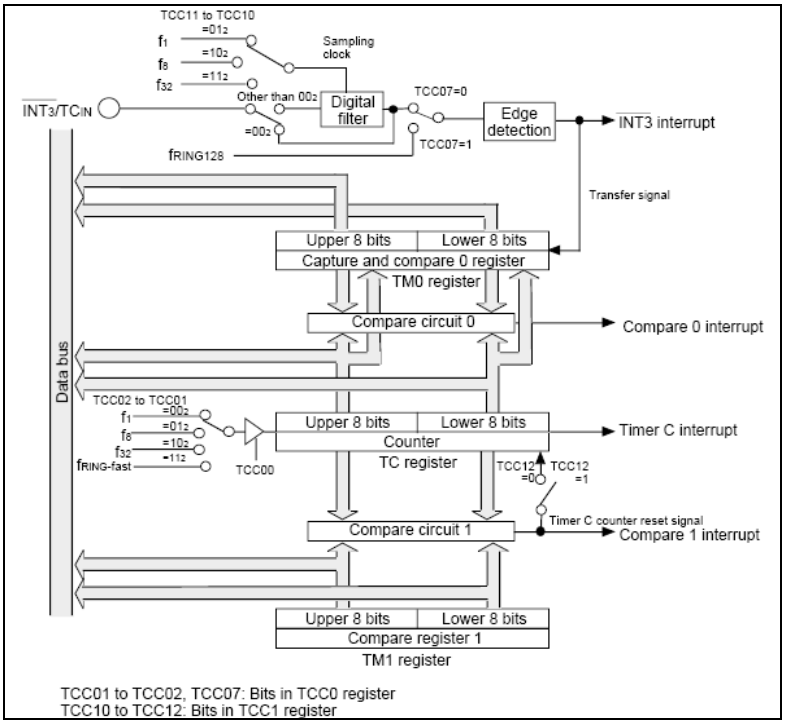
Exemple per aconseguir 1 segon:

Es pot escollir entre el rellotge intern o la base de temps creada amb el TimerY, escollim aquesta última creada de 0,02 segons.

$$\text{TimerZ} = \frac{(\text{PREZ}+1)\cdot(\text{TZPR}+1)}{1/\text{TimerY}} = \frac{1\cdot 50}{50} = 1 \text{ segon}$$

PREZ = 0 i TZPR=49

El temporitzador C es pot fer servir comptar ascendentment, o transferir el valor del registre TC al registre TM0 (el valor del registre TC es posa a "000016" quan s'atura un compte).



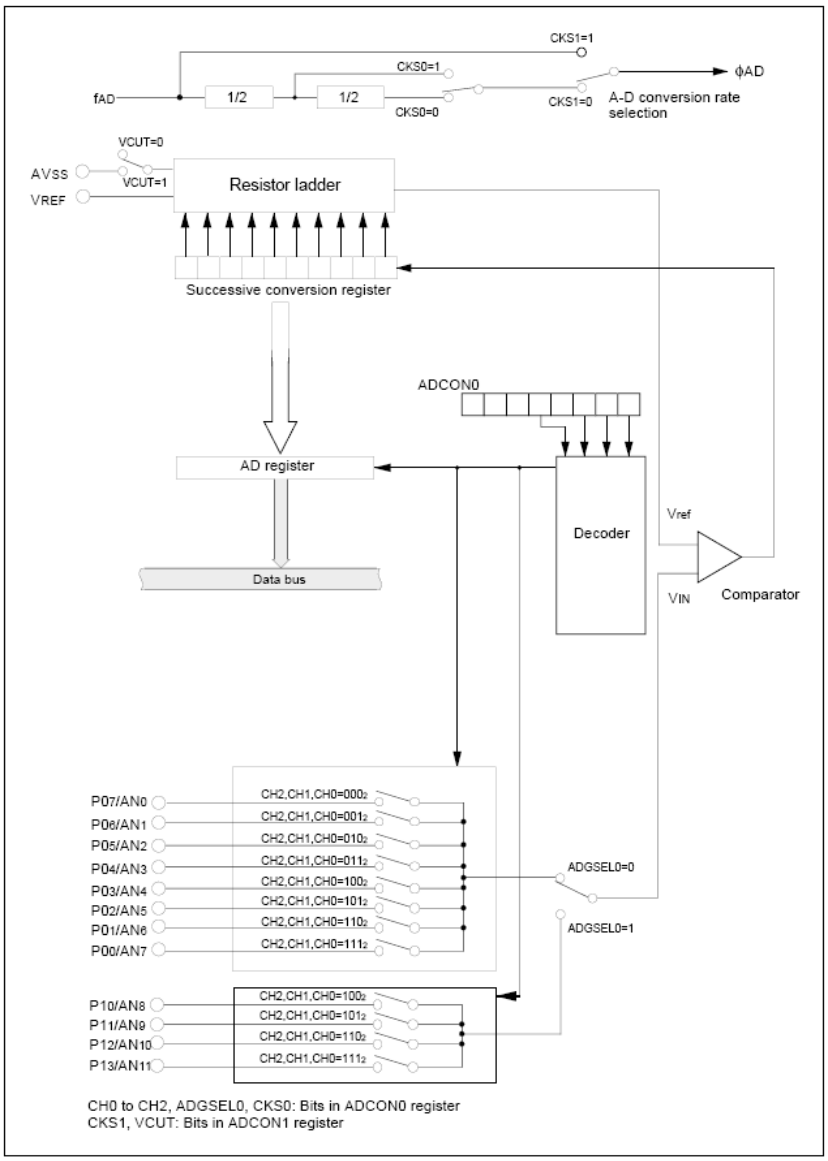
Il·lustració 35: Timer C

4.1.3. Conversió A-D

El convertidor A-D consisteix en un circuit successiu de conversió A-D amb una aproximació de 10 bits i un amplificador capacitiu. Les entrades analògiques comparteixen els pins de P00 fins a P07 i de P10 a P13. Per tant, s'ha de tenir en compte la declaració d'entrades i sortides segons quines facis servir (a "0" son declarades com entrades).

Si no fas servir el convertidor A-D, fixar VCUT bit a "0" (Vref desconnectat), per tal que no circuli corrent pel pin VREF, i reduir el consum d'energia.

El resultat de la conversió del A-D s'emmagatzema en el registre del AD.



Il·lustració 36: Diagrama del A-D

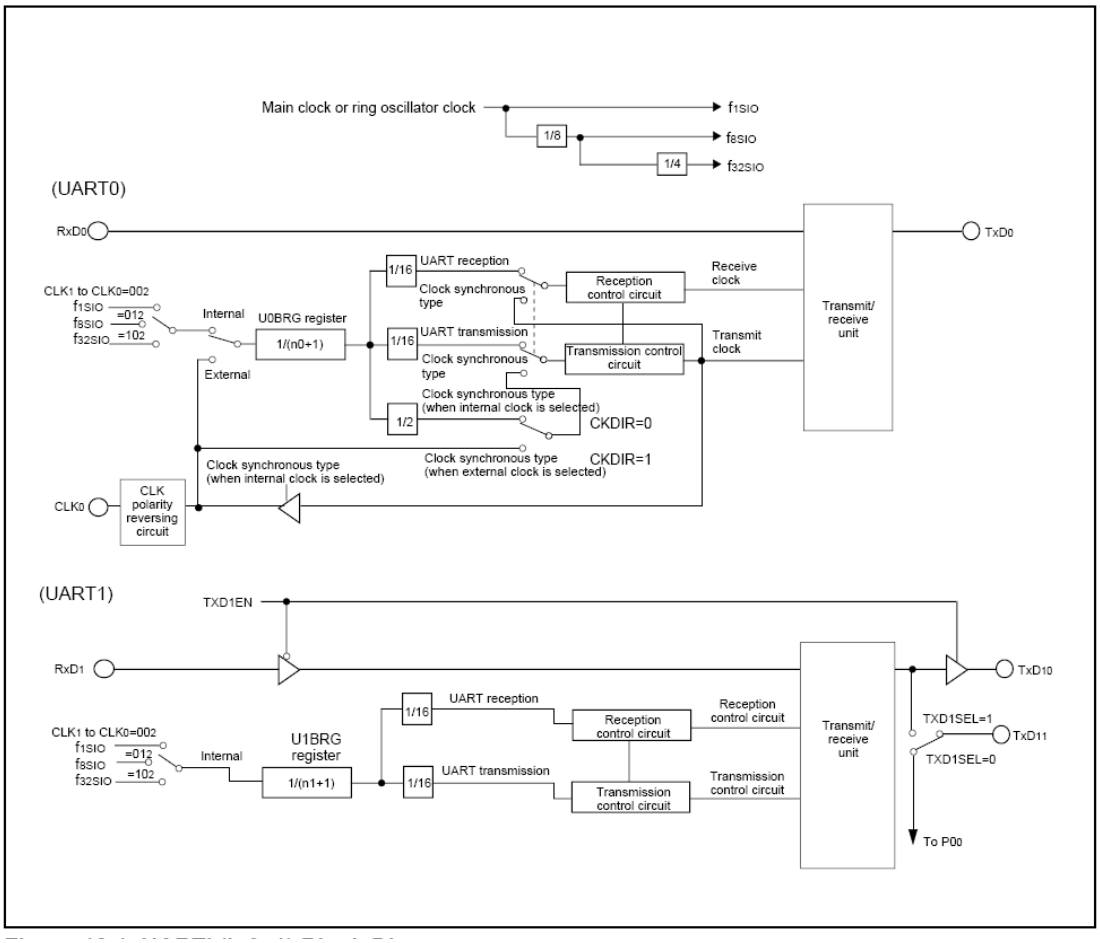
Per començar la conversió A-D s'ha de posar el bit ADST del ADCON0 a 1.

```
ADCON0_bit.ADST = 1;
```

El microcontrolador emmagatzema la conversió al ADL.

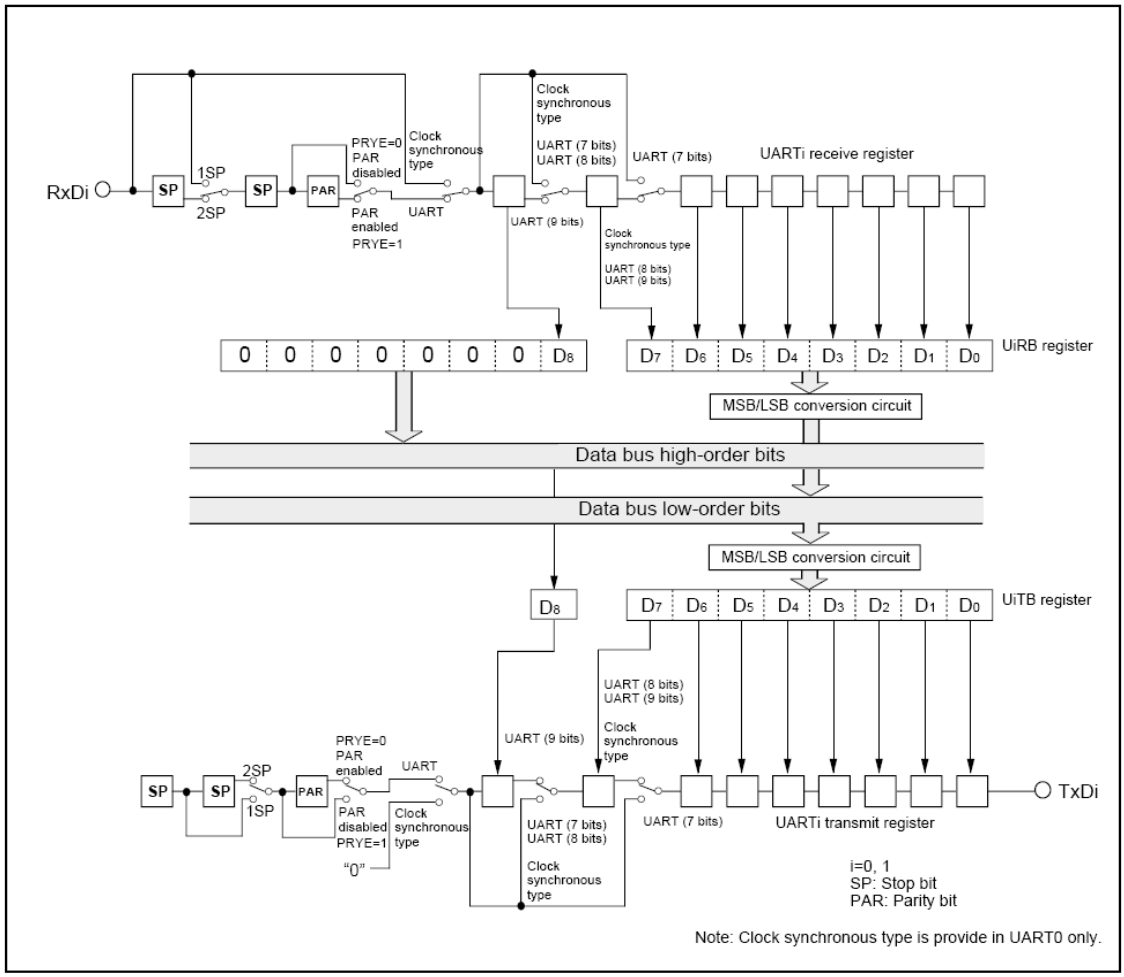
4.1.4. I/O UART

El serial I/O es configura en dos canals: UART0 i UART1. Aquests canals serveixen per realitzar la comunicació sèrie amb l'exterior.



Il·lustració 37: Diagrama de blocs de la UARTi(i=0, 1)

UART0 i UART1 tenen cadascun un comptador de temps exclusiu per generar el clock de transferència, així funcionen independentment l'un de l'altre.



Il·lustració 38: Funcionament de la UARTi

Cal establir el baud rate amb els registres U0BRG i U1BRG per fixar la velocitat de transmissió.

$$Formula: Tc = \frac{fj}{16 \cdot (n+1)}$$

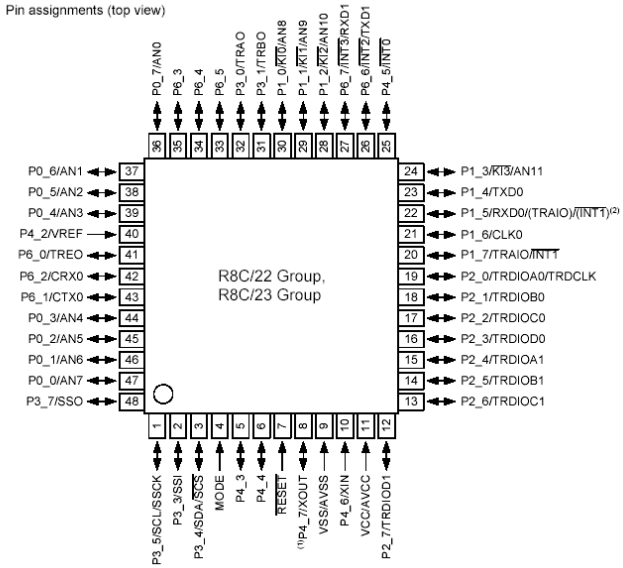
Les dades es transmeten per U0TB i U1TB i es reben als registres U0RB i U1RB.

4.2. R8C/23 Group

El gurp R8C/23 [8] te un ampli ventall de microcontroladors per escollir, el microcontrolador que te la placa de proves, i amb el que farem els programes es el següent:

Numero de serie	Memoria ROM	Memoria RAM	Tipus d'encapsulat	Comentaris
R5F212237JFP (D)	48 Kbytes	2.5 Kbytes	PLQP0048KB-A	Versió J

Taula 13: Característiques del R5F21227JFP



Il·lustració 39: Distribució dels pins del R8C/23

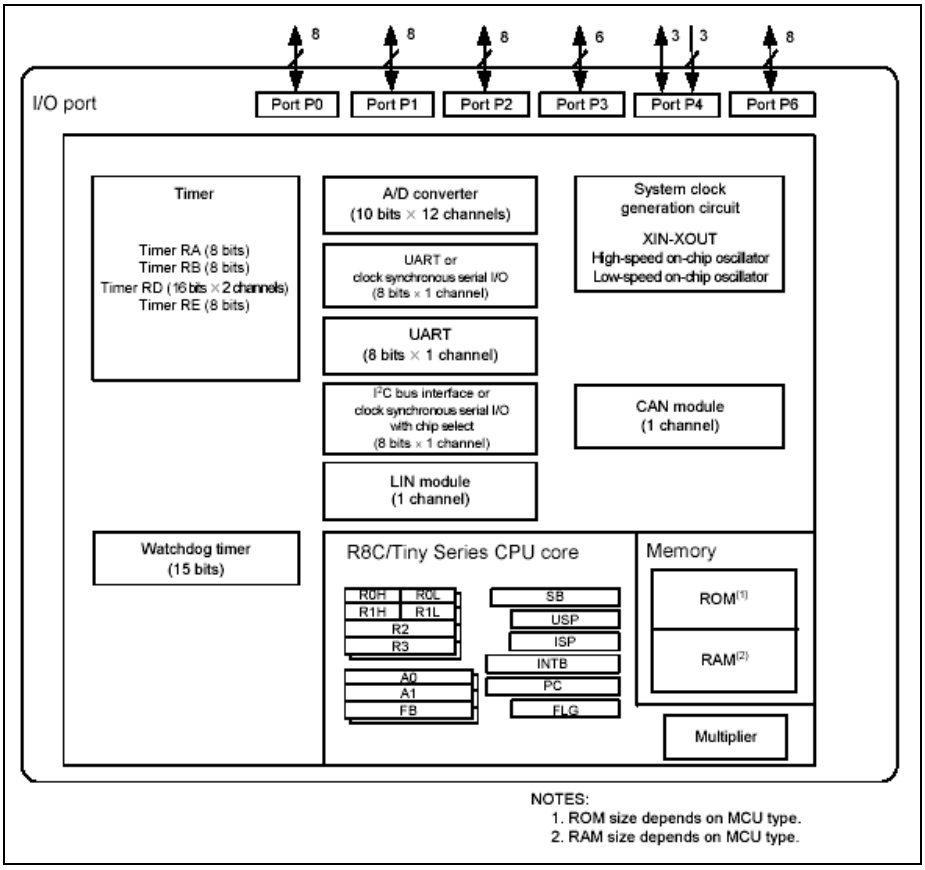
La següent taula conte les principals característiques del microcontrolador R5F21227JFP (D):

Item		**Especificacions**
CPU	Numero fundamental d'instruccions	89 instrruccions.
	Temps minim per d'instrcucció	50ns (f(XIN)=20MHz, Vcc 3.5 a 5.5 V). 50ns (f(XIN)=10MHz, Vcc 2.7 a 5.5 V).
	Capacitat de la Memoria	48Kbytes.
Funcions per periferics	Ports	41 pins de ports d'entrada – sortida i 3 pins de ports d'entrada.
	Timers	Timer RA: 8 bits x 1 canal. Timer RB: 8 bits x 1 canal. Timer RD: 16 bits x 2 canal. Timer RE: Comparador de 8 bits.
	Interficia bus serie	2 canals; UART0 i UART1
	Modul LIN	Hardware Lin: 1 canal.
	Modul CAN	1 can
	Conversor A/D	Conversor de 10 bits; 1 circuit amb 12 canals.
	Watchdog timer	
Característiques electriques	Font de voltatge	3.0 a 5.5 V(f(XIN)=20MHz). 2.7 a 5.5 V(f(XIN)=10MHz).

	Corrent nominal	Typ 12.5 mA (VCC=5 V, f(XIN)=20MHz). Typ 6.5 mA (VCC=5 V, f(XIN)=10MHz).
Memoria Flach		
Temperatura ambient de treball		-40 a 85 Cº.
Encapsulat		Modul de plastic LQFP de 48 pins.

Taula 14: Característiques detallades del R8C/23

Diagrama de blocs:



Il·lustració 40: Diagrama de blocs del microprocessador R8C/23

Les característiques principals d'aquest diagrama de blocs que hem utilitzat en el desenvolupament d'aquest projecte es descriuen en els apartats posteriors.

4.2.1. Clock

El microcontrolador pot obtenir la senyal de rellotge (clock) de tres circuits:

- XIN closck oscillation circuit
- Low speed on-chip oscillator

- High – speed on chip oscillator

La següent taula conte les especificacions dels circuits de clock:

Item	XIN Clock Oscillation Circuit	On-Chip Oscillator	
		High-Speed On-Chip Oscillator	Low-Speed On-Chip Oscillator
Use of Clock	• CPU clock source • Peripheral function clock source	• CPU clock source • Peripheral function clock source • CPU and peripheral function clock sources when XIN clock stops oscillating	• CPU clock source • Peripheral function clock source • CPU and peripheral function clock sources when XIN clock stops oscillating
Clock Frequency	0 to 20 MHz	Approx. 40 MHz(3)	Approx. 125 kHz
Connectable Oscillator	• Ceramic resonator • Crystal oscillator	–	–
Oscillator Connect Pins	XIN, XOUT(1)	–(1)	–(1)
Oscillation Stop, Restart Function	Usable	Usable	Usable
Oscillator Status After Reset	Stop	Stop	Oscillate
Others	Externally generated clock can be input(2)	–	–

Taula 15: Característiques del clock

En els programes creats, hem utilitzat principalment el circuit de rellotge “XIN Clock Oscillation circuit” , que com s’observa en la taula anterior pot generar una senyal de rellotge de 20Mhz.

La següent taula mostra d’una manera reduïda la configuració bàsica dels registres associats a aquest circuit de rellotge amb els corresponents valors de configuració perquè la font de rellotge del microcontrolador tingui una freqüència de 20MHz.

Nom del bit	Registre del al bit	Funcio de bit
cm05 = 0;	CM0 (Sistem clock control register 0)	Engega o para l’oscilació del clock.
cm13 = 1	CM1 (Sistem clock control register 1)	Hbilita els pins del microchip, per que els utilizi el cristall exterior
cm15 = 1		Seleccionem el valor per el qual dividim la senyal de rellotge. Podem dividir pels valors 1,2,4 i 16.
cm16 = 0;		
cm17 = 0;		
ocd2 = 0;	OCD (Oscillation stop detection register)	Seleccionem el circuit de rellotge.

Taula 16: Configuració del clock

4.2.2. Timers

El microcontrolador esta compost per dos rellotges de 8 bits amb 8 bits de prescaler, dos rellotges de 16 bits i un rellotge amb 4 bits i 8 bits de comptador.

Els rellotges de 8-bits mes 8 bits de prescaler s’anomenen “Timer RA” i “Timer RB”. Aquests tenen un registre de recarrega que memoritza el valor per defecte del comptador. El rellotge de 16-bits s’anomena “Timer RD”.

El rellotge que resta s'anomena "Timer RE", aquest te un comptador de 4 bits i un altre de 8 bits.

Tots els rellotges funcionen independentment.

Timer RA:

El Timer RA es un rellotge de 8 bits amb 8 bits de prescaler.

El rellotge i el prescaler tenen tots dos un registre de recarrega i un comptador. El registre de recarrega i el comptador esmentats estan situats a la mateixa posició de memòria, es a dir tenen la mateixa adreça. Es poden accedir a aquestes posicions mitjançant els registres TRAPRE i TRA.

El timer RA pot funcionar en 5 modes de operació:

- Timer mode: El rellotge conte amb un font de rellotge interna.
- Pulse output mode: Aquest mode pot generar un senyal de polsos, mitjançant l'underfolw del timer.
- Event counter mode: El rellotge conte polsos externs.
- Pulse width measurement mode: El rellotge mesura l'amplada de pols provenint de una senyal de pols extern.
- Pulse period measurement mode: El rellotge mesura el període de pols provenint de una senyal de pols extern.

4.2.3. Conversió A-D

El convertidor d'analògic – digital te una precisió aproximada de 10 bits. L'entrada analògica del convertidor comparteix els pins amb el ports del P0_0 a P0_7, P1_0 a P1_3, per tant al utilitzar aquets pins, hem d'assegurar que els bits dels ports corresponents es fixen a 0 (mode d'entrada).

Sinó fem servir el convertidor analògic/digital, s'ha de fixar el bit de VCUT del registre ADCON1 a 0, de manera que cap corrent flueixi pel pin Vref i així ajudi a reduir el consum.

El resultat de la conversió s'emmagatzema al registre AD.

La següent taula enumera el funcionament del convertidor analògic/digital

Item	Performance
A/D Conversion Method	Successive approximation (with capacitive coupling amplifier)
Analog Input Voltage(1)	0 V to AVCC
Operating Clock ϕAD(2)	4.2 V ≤ AVCC ≤ 5.5 V f1, f2, f4, fOCO-F 2.7 V ≤ AVCC < 4.2 V f2, f4, fOCO-F
Resolution	8 bit or 10 bit is selectable
Absolute Accuracy	AVCC = Vref = 5 V, ϕAD = 10MHz • 8-bit resolution ±2 LSB • 10-bit resolution ±3 LSB AVCC = Vref = 3.3 V, ϕAD = 10MHz • 8-bit resolution ±2 LSB • 10-bit resolution ±5 LSB
Operating Mode	One-shot and repeat modes(3)
Analog Input Pin	12 pins (AN0 to AN11)
A/D Conversion Start Condition	• Software trigger Set the ADST bit in the ADCON0 register to 1 (A/D conversion starts) • Capture Timer RD interrupt request is generated while the ADST bit is set to 1
Conversion Rate Per Pin	• Without sample and hold function 8-bit resolution: 49ϕAD cycles, 10-bit resolution: 59ϕAD cycles • With sample and hold function 8-bit resolution: 28ϕAD cycles, 10-bit resolution: 33ϕAD cycles

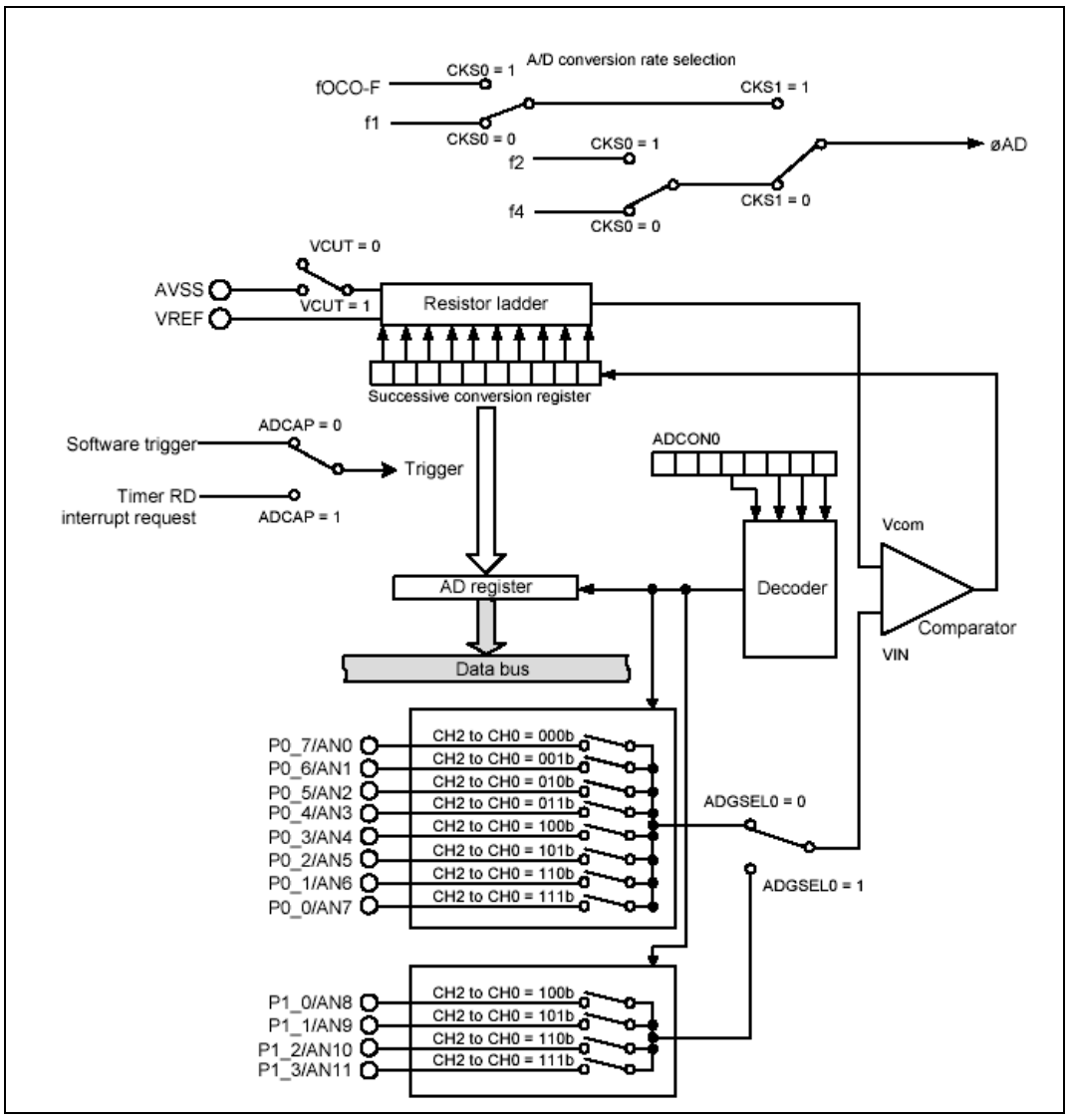
Taula 17: Rendiment del A/D

Tenim 12 canals d'entrada al convertidor analògic – digital, concretament del canal 0 (AN0) fins al canal 11 (AN11). L'usuari pot escollir el canal del qual vol transformar les dades mitjançant els bits CH2, CH1 i CH0 del registre ADCON 0, s'ha de tenir en compte que si volem escollir un canal entre AN0 i AN7 hem de posar el bit AGSEL del registre ADCON0 a 0, per afegiment, si volem escollir un canal entre AN8 i AN11 hem de posar l'esmenta't bit a 1.

El valor dels canals per on entre en valor analògic es comparat amb un valor de referència, per fer-ho necessitem ajustar el bit VCUT del registre ADCON1 a 1, per contra, si posem aquest bit a 0, el valor analògic que volem convertir, amb referència a Vcc, en aquest cas 5 volts aproximadament.

El resultat de la comparació passa a un registre, on mitjançant conversions successives, obtenim el valor en digital, aquest s'emmagatzema al registre AD. Aquest registre esta connecta al bus de dades.

En la següent il·lustració podrem observar gràficament el funcionament del converso A/D mitjançant un diagrama de blocs.



Il·lustració 41: Diagrama de blocs del convertidor A/D

Nosaltres hem utilitzat principalment el canal 8 del convertidor, perquè es on esta connectat el potenciòmetre de la placa de prova, seguidament mostrem una taula on es pot observar d'una manera reduïda la configuració necessària per obtenir el valor digital convertit a partir del valor analògic del potenciòmetre.

Nom del bit	Registre del bit	Funció de bit
ch0=0	ADCON0 (A/D control registre 0)	Seleccionem el canal de conversió, AN8.
ch1=0		
ch2=1		
adgsel0=0		
Adcap=0		
Adst=1		Iniciem la conversió.
Csk0=0	ADCON0(A/D control registre 0)	Seleccionem la freqüència de conversió.

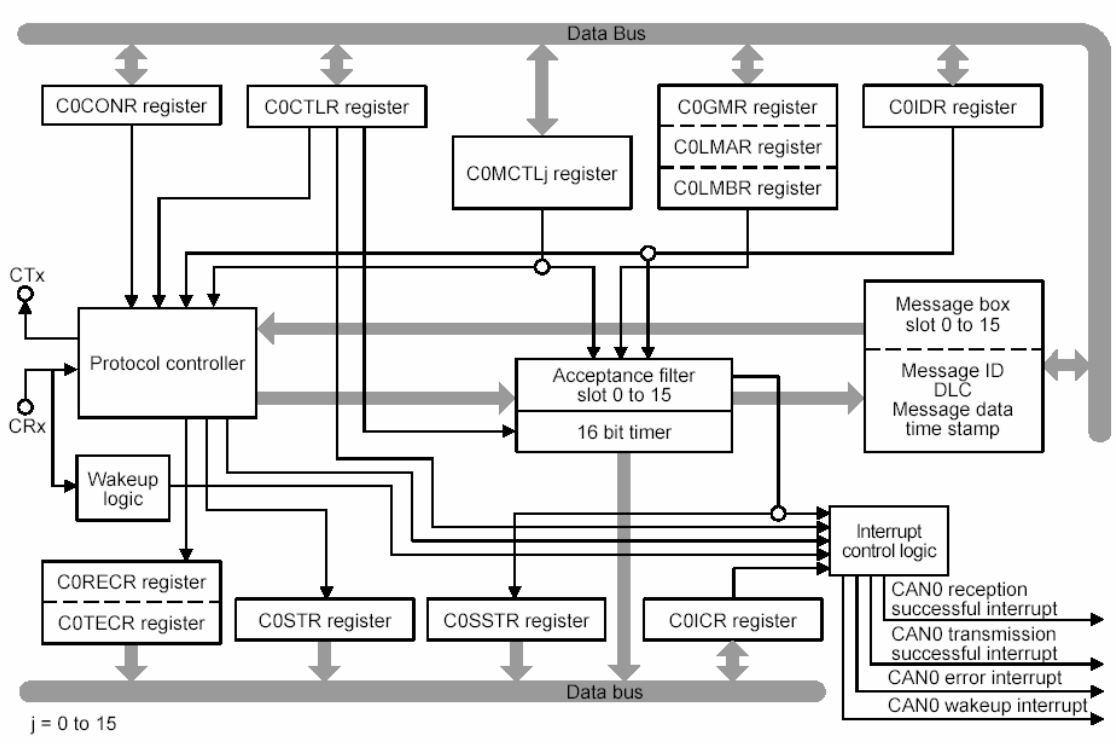
Csk1=0	ADCON1(A/D control registre 1)	
Bits=0		Seleccionem conversions de 8 bits.

Taula 18: Registres del A/D

4.2.4. CAN

El mòdul de CAN del MUC es una implementació del control de comunicació pel protocol CAN 2.0B.

Aquest pot enviar i rebre missatges en format estàndard i extens. Segurament descriurem mitjançant un diagrama de blocs les principals característiques d'aquest mòdul.



Il·lustració 42: Diagrama de blocs del CAN

Descripció individual dels blocs del diagrama:

CTx/Rx: pins entrada – sortida de CAN.

Protocol controller: Aquest controlador dirigeix el bus d'arbitratge i el protocol de servei de CAN.

Ex. Bit timing, stuffing, error status etc.

Message box: Aquest bloc de memòria consta de 16 ranures o espais, que es poden configurar com a transmissor o receptor. Cada espai conte un identificador (ID) individual, un codi de la mida de les dades que tindrà el missatge, un camp de dades de 8 bits i un "time stamp".

Acceptence filter : Aquest bloc funciona filtren els identificadors (ID) dels missatges rebuts. Per fer aquesta tasca de filtratge, s'utilitzen els registres C0GMR, C0LMAR i C0LMBR.

16 bit timer: S'utilitza per la funció "time stamp". Quan rebem un missatge i aquest s'emmagatzema a la memòria, el valor del timer es guardat al "timer stamp".

Wake up function: La interrupció de "CAN 0 wake up" es genera mitjançant un missatge des de el CAN bus.

Interrupt generator function: Els esdeveniments d'interrupció provenen del mòdul CAN 0. Els tipus d'interrupció poden ser per recepció, transmissió, error o "wake up".

El mòdul CAN disposa de 15 espais (slots) per emmagatzemar els missatges que enviarem o rebrem, el marc on s'inclouen aquets 15 espais s'anomena "CAN message box", en la següent taula podem observar l'adreça de memòria de cada slot i la forma d'accedir-hi.

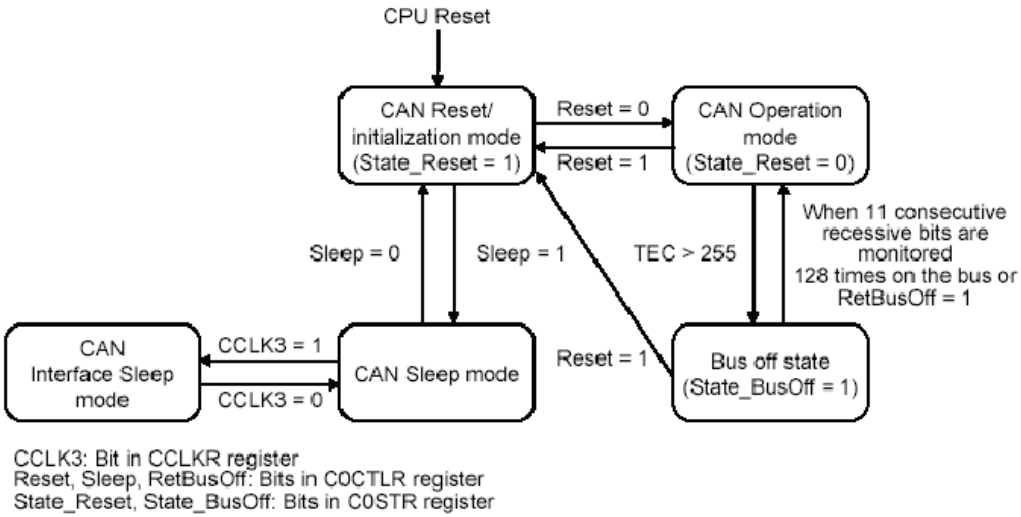
Address	Message Content (Memory Mapping)	
CAN0	Byte access (8 bits)	Word access (16 bits)
1360h + n • 16 + 0	SID10 to SID6	SID5 to SID0
1360h + n • 16 + 1	SID5 to SID0	SID10 to SID6
1360h + n • 16 +2	EID17 to EID14	EID13 to EID6
1360h + n • 16 + 3	EID13 to EID6	EID17 to EID14
1360h + n • 16 + 4	EID5 to EID0	Data Length Code (DLC)
1360h + n • 16 + 5	Data Length Code (DLC)	EID5 to EID0
1360h + n • 16 + 6	Data byte 0	Data byte 1
1360h + n • 16 + 7 ⋮ 1360h + n • 16 + 13	Data byte 1 ⋮ Data byte 7	Data byte 0 ⋮ Data byte 6
1360h + n • 16 + 14	Time stamp high-order byte	Time stamp low-order byte
1360h + n • 16 + 15	Time stamp low-order byte	Time stamp high-order byte

n: Slot number, n = 0 to 15

Taula 19: CAN capça de missatges

El mòdul de CAN pot entrar en quatre modes d'operació.

- CAN *reset/initialization* mode.
- CAN *sleep* mode.
- CAN *operation* mode.
- CAN *interface sleep* mode.



Il·lustració 43: Transicions entres els modes d'operació

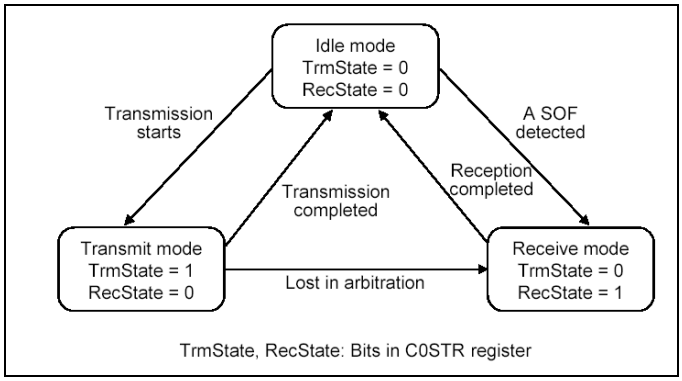
CAN *reset/initialization* mode:

El mòdul CAN pot entrar en mode *reset/initialization,* amb un reset a la CPU o be ajustant el bit de reset mitjançant el registre C0CTLR. Quant el bit anterior esmentat esta a 1, el bit "State_Reset bit" del registre C0STR també s'ha de posar a 1.

Quan el mòdul CAN es troba en aquest mode d'operació, passen diverses coses:

- La comunicació CAN no es possible.
- Si el modul CAN passa a mode *reset/initialization*duran la transmissió de un missatge, aquest es mantindra al mode *CAN opertaion* fins que la transmissió s'hagi completat.
- Els registres, C0IDR, C0MCTLi (i = 0 to 15), C0ICR, C0STR, C0RECR i C0TECR es reeinicien. Tots aquets registres estan bloqugats per evitar la modificacio de la CPU.
- Els registres C0CTLR, C0CONR, C0GMR, C0LMAR i C0LMBR, i el "CAN0 message box" retenen el seu contingut perque aquest estigui disponible per a l'acces de la CPU.

CAN *operation* mode:



Il·lustració 44: Mode d'operació del CAN

El modeul CAN pot entrar en mode *operation* quan el bit de reset del registre C0CTLR es posa a 0, quan aixo passa el bits "State_Reset" del registre C0STR es posa a 0. El modul CAN executa les funcions segments després de que onze bits consecutius siguin detectats en el mode *operation*.

- El modul CAN pot rebre i transmetre un missatge.
- El modul controla "error status" mitjançant un contatje dels errors de transmissió i recepció. La comunicació CAN depen de "l'status error". El modul es posat en un dels tres submodes del mode *operation.*
- Idle mode: tots els nodes destan parats.
- Receive mode: El node pot rebre un missatge transmès per un altre node.
- Transmit mode: El node pot transmetre un missatge. Aquest node pot rebre el missatge que acaba de transmetre quan el bit "LoopBack" del registre C=CTRL. Es posa a 1 (mode de llaç tancat).

CAN *sleep* mode:

El mòdul CAN entra en mode sleep quan el bit "sleep" del registre C0CTLR es posa a 1. Es pot entrar en aquest mode a partir del mode *reset/initialization.*

En aquest mode el consum d'energia es pot reduir perquè el rellotge o "clocck" no prové del mòdul CAN.

CAN *interface sleep* mode:

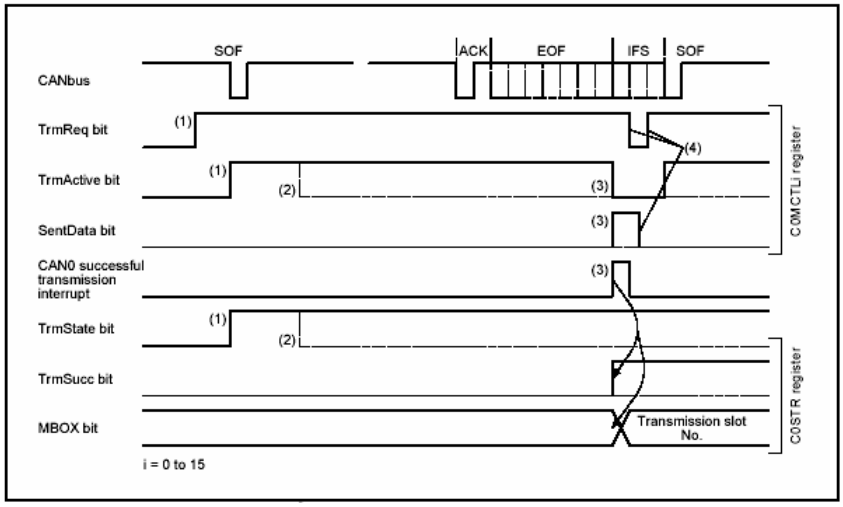
El mòdul CAN pot entrar en mode *interface sleep* quant el bit CCLKR3 del registre CCLKR es posa a 1.

S'entra en aquest mode d'operació a partir solsament del mode *sleep.*

En aquest mode el consum d'energia pot ser reduït perquè la font de rellotge no prové de la interfície de la CPU del mòdul CAN.

Diagrames de transmissió:

Quan la transmissió d'un missatge CAN es fa correctament, tot un seguit de bits es posen a 0 i a 1, en aquest diagrama podrem explicar quins son els bits que actuen quan el missatge s'ha transmès correctament i quins, quan el missatge no s'ha transmès satisfactòriament.



Il·lustració 45: Gràfica de transmissió de trames CAN

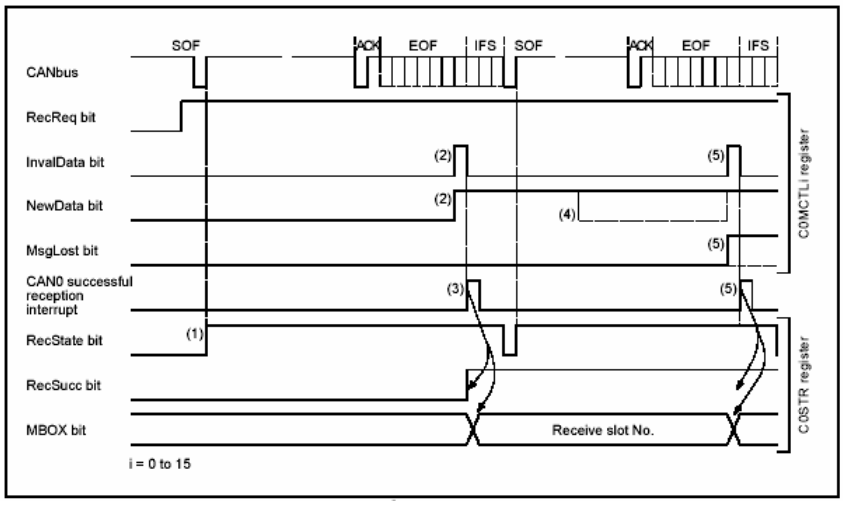
Quan s'inicia la transmissió d'una trama, el primer bit que actua és el bit TrmReq, aquest bit es posa a 1, perquè pugui transmetre el bus CAN ha d'estar lliure, quan el bus esta disponible el bit TrmActive de registre C0MCTL es posa a 1, al mateix temps també passa a 1 el bit TrmState conforma s'està transmetent.

Si el missatge es perd durant la transmissió, els bits TrmActive i TrmState passen de 1 a 0 al mateix temps.

Quan la transmissió es completada amb èxit, s'activen immediatament els bits SentData i la interrupció associada a la transmissió de trames CAN, posteriorment aquest bits es tornen a posar a 0 i també es posen a 0 el bit SentData de l registre C0STR i seguidament el bit TrmActive del registre C0MCTL.

Diagrames de recepció:

Quan la transmissió d'un missatge CAN es fa correctament, tot un seguit de bits es posen a 0 i a 1, en aquest diagrama podrem explicar quins son els bits que actuen quan el missatge s'ha transmès correctament i quins quan el missatge no s'ha transmès satisfactòriament.



Il·lustració 46: Gràfica de recepció de trames CAN

El bit encarregat de comunicar que s'ha rebut una trama CAN, s'anomena RecReq i pertany al registre de control del mòdul CAN, quan l'esmenta't bit es posa a 1 significa que hem rebut un missatge, al mateix temps el bit RecState també es posarà a 1 conforma s'està reben una trama CAN.

Si el missatge que s'ha rebut no complís el protocol CAN o tingues algun error, s'activaria el bit InvalData, si el missatge no te errors i compleix el protocol CAN, s'activaria el bit NewData, que ens informa que ha arribat una dada nova, immediatament l'esmenta't bit es posa a 0 altre vegada, en el flanc de baixada es posen a 1 el bit RecState i s'activa la interrupció associada a la recepció de missatges de CAN, posteriorment podem estreure el missatge dels slots de recepció on s'ha guardat, el bit que ens indica en quin slot esta ubicat s'anomena MBOX.

Capítol 4:

DESENVOLUPAMENT DEL PROJECTE I APLICACIONS

5. Desenvolupament del projecte i Aplicacions

5.1. Primeres proves amb la placa 1

Amb la R8C 3-D Board [6], hem provat la comunicació sèrie amb el PC, fent servir el valor d'un potenciòmetre per enviar-ho com a dada i veure'l mitjançant l'HyperTerminal i el LabView.

Il·lustració 47: Organigrama comunicació sèrie

5.1.1. Funcionament del programa

Aquest programa s'encarrega d'agafar el valor del sensor en voltatge (en aquest cas un potenciòmetre, amb el que obtenim diferents voltatges, variant el valor de resistència), convertir el voltatge obtingut (analògic) en un valor digital, i mitjançant el port sèrie enviar el resultat de la conversió, a l'ordinador.

Establim un temporitzador a un segon, i el fem servir per crear una interrupció que s'anirà repetint aquest temps, on farem la nova conversió i l'enviament. D'aquesta manera, obtindrem el nou estat del potenciòmetre cada segon.

Una vegada a l'ordinador fem servir l'HyperTerminal o el LabView per veure i administrar les dades.

5.1.2. Programa Comentat

Per començar incloem les llibreries que farem servir. Les llibreries son les corresponents al microcontrolador R5F2111 e inclouen les definicions de les entrades i sortides i tots els registres que es poden fer servir.

```
#include <ior8c11_13.h>

#include <intrinsics.h>
```

Les variables globals que farem servir, son:

```
unsigned char ValorAD, Trama[4];

int NT=0, DIV=0x3E8, i=0,a=0,b=0;
```

El tipus unsigned char es de 8 bits, el seu rang va de 0 a 255 i es fa servir per números petits i caràcters. Nosaltres el fem servir per les variables "ValorAD", on registrarem el valor obtingut de la convessió d'analogic a digital, i en "Trama[4]", que es una matriu amb quatre elements, i que farem servir per enviar element a element el contingut de "ValorAD".

El tipus int es de 32 bits, el seu rang va de -2147883648 a 2147883647 i es fa servir per números petits i control de bucles. Nosaltres el fem servir per les variables "NT" i "DIV" que ens ajuden a trossejar el valor de "ValorAD" i emmagatzemar-ho a "Trama", "a" "b" i "i" les utilitzem per controlar bucles.

Programa principal:

void main (**void**)

{

Una vegada dins del programa principal ens encarreguem de la configuració del registres i ports.

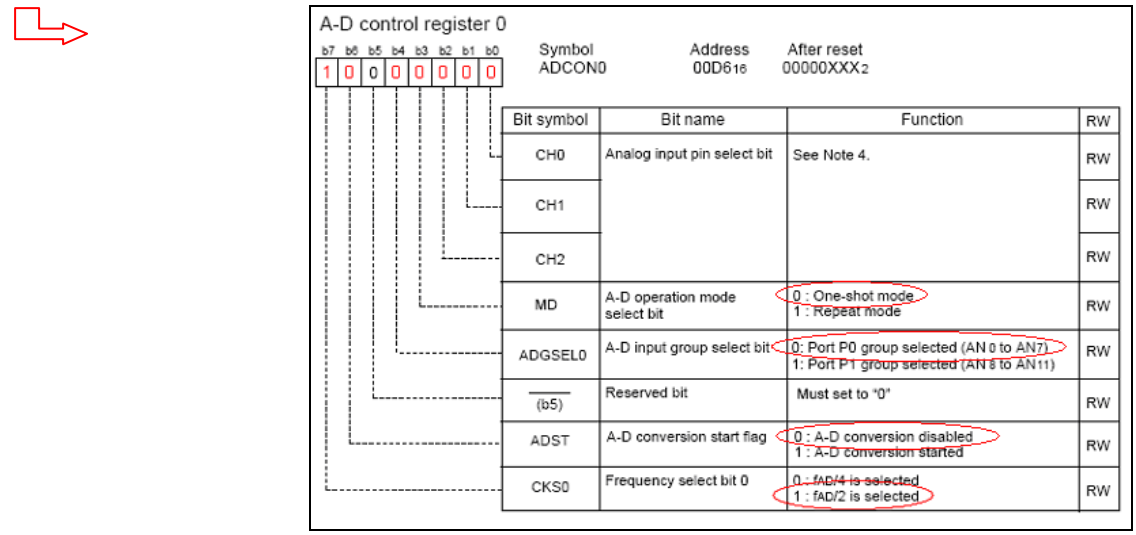
/* PORTS D'ENTRADA I SORTIDA */

PD1 = 0xFF; Declarem totes les direccions del Port 1 com a sortides.

P1 = 0x00; Posem tots els LED's en off.

/* REGISTRES DEL ADCON */

ADCON0 = 0x80;

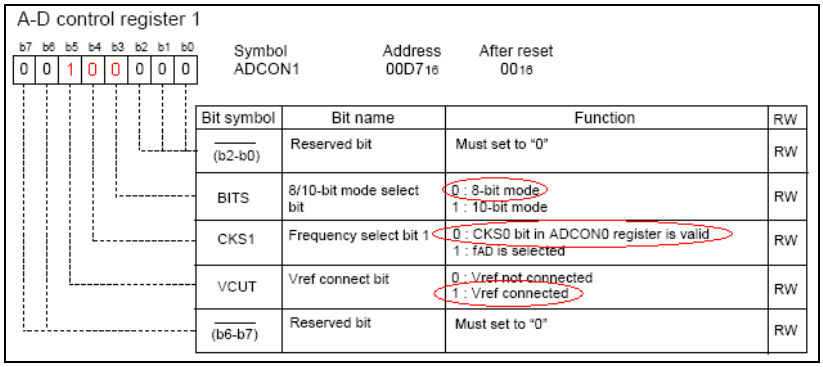


Il·lustració 48: ADCON0

Escollim la manera de fer la conversió d'analògic a digital i la deixem desconnectada per el moment.

Només volem que faci la conversió quan li indiquem per tant "MD" a "0" fa que no sigui continua sinó que fa una única conversió cada vegada que habilitem "ADST", les connexions físiques son al port P0 pel que posem "ADSGSEL=" a "0".

ADCON1 = 0x20;



A-D control register 1

b7	b6	b5	b4	b3	b2	b1	b0
0	0	1	0	0	0	0	0

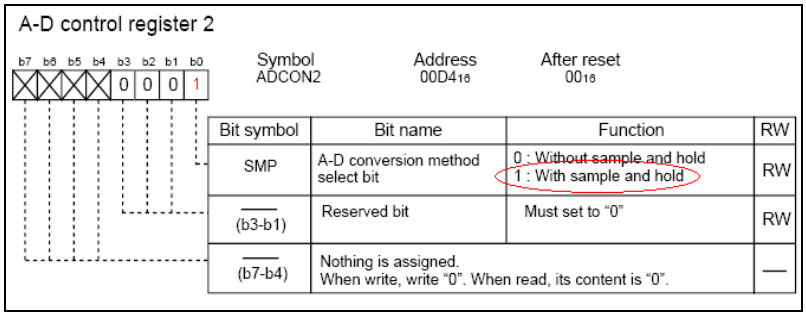
Symbol: ADCON1 — Address: 00D7₁₆ — After reset: 00₁₆

Bit symbol	Bit name	Function	RW
— (b2-b0)	Reserved bit	Must set to "0"	RW
BITS	8/10-bit mode select bit	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	0 : CKS0 bit in ADCON0 register is valid 1 : fAD is selected	RW
VCUT	Vref connect bit	0 : Vref not connected 1 : Vref connected	RW
— (b6-b7)	Reserved bit	Must set to "0"	RW

Il·lustració 49: ADCON1

Volem conversions de 8 bits, que treballi amb el rellotge CKS0 i que tingui com a referència Vref.

ADCON2 = 0x01;



A-D control register 2

b7	b6	b5	b4	b3	b2	b1	b0
X	X	X	X	X	0	0	1

Symbol: ADCON2 — Address: 00D4₁₆ — After reset: 00₁₆

Bit symbol	Bit name	Function	RW
SMP	A-D conversion method select bit	0 : Without sample and hold 1 : With sample and hold	RW
— (b3-b1)	Reserved bit	Must set to "0"	RW
— (b7-b4)	Nothing is assigned. When write, write "0". When read, its content is "0".		—

Il·lustració 50: ADCON2

Amb mostra i control.

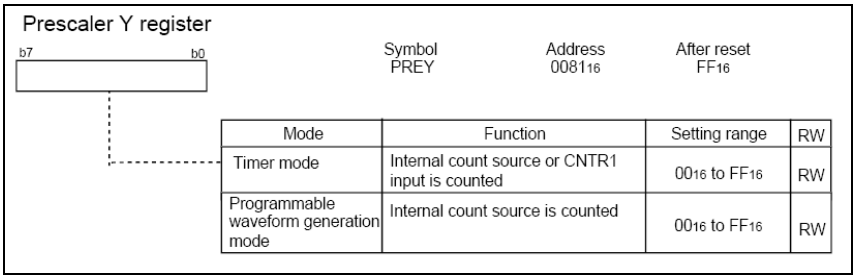
```
/* TIMERS */

    /* REGISTROS DEL TIMER Y */
```

Ajustem el TimerY a 0,01segons

$$\text{TimerY} = \frac{(\text{nPREY}+1)\cdot(\text{mTYPR}+1)}{\text{Fi}} = \frac{100\cdot 200}{2000000} = 0.01 \text{ Segons}$$

Per aconseguir aquest valor de temps, hem d'ajustar els registres de la formula als valors obtinguts.

PREY = 100-1;



Prescaler Y register

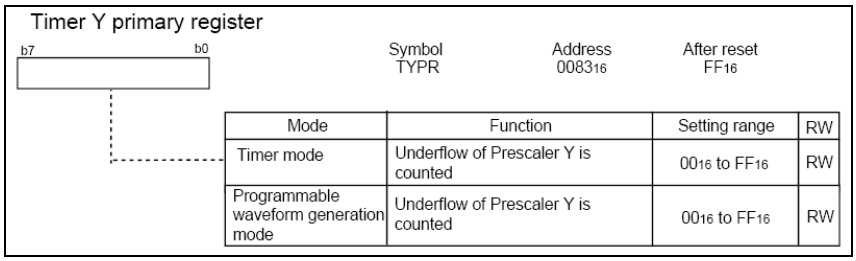
b7 b0

Symbol PREY | Address 008116 | After reset FF16

Mode	Function	Setting range	RW
Timer mode	Internal count source or CNTR1 input is counted	0016 to FF16	RW
Programmable waveform generation mode	Internal count source is counted	0016 to FF16	RW

Il·lustració 51: PREY

TYPR = 200-1;



Timer Y primary register

b7 b0

Symbol TYPR | Address 008316 | After reset FF16

Mode	Function	Setting range	RW
Timer mode	Underflow of Prescaler Y is counted	0016 to FF16	RW
Programmable waveform generation mode	Underflow of Prescaler Y is counted	0016 to FF16	RW

Il·lustració 52: TYPR

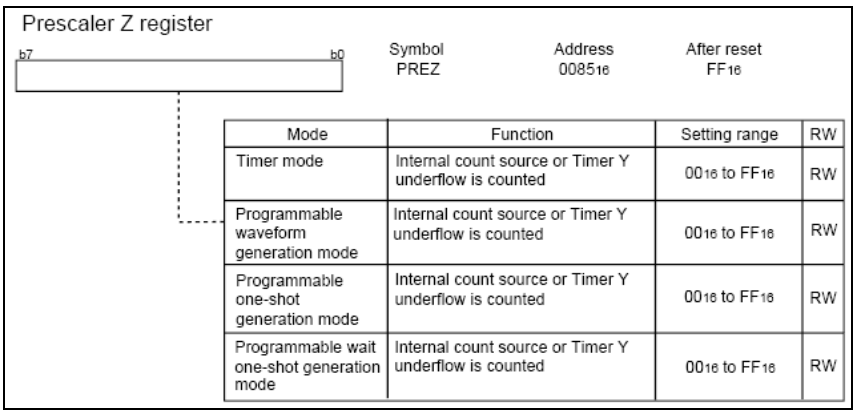
/* REGISTROS DEL TIMER Z */

Ajustem el TimerZ a 1segon, fent servir el valor de temporització obtingut amb el TimerY.

$$\text{TimerZ} = \frac{(\text{nPREZ}+1)\cdot(\text{mTZPR}+1)}{\text{Fi}} = \frac{1\cdot 100}{100} = 1 \text{ Segon}$$

Per aconseguir aquest valor de temps, hem d'ajustar els registres de la formula als valors obtinguts.

PREZ = 0x00;



Prescaler Z register

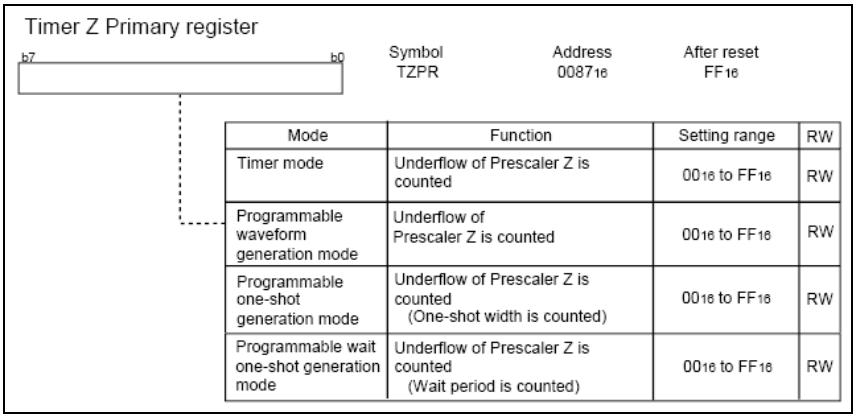
b7 b0

Symbol: PREZ | Address: 008516 | After reset: FF16

Mode	Function	Setting range	RW
Timer mode	Internal count source or Timer Y underflow is counted	0016 to FF16	RW
Programmable waveform generation mode	Internal count source or Timer Y underflow is counted	0016 to FF16	RW
Programmable one-shot generation mode	Internal count source or Timer Y underflow is counted	0016 to FF16	RW
Programmable wait one-shot generation mode	Internal count source or Timer Y underflow is counted	0016 to FF16	RW

Il·lustració 53: PREZ

TZPR = 100-1;



Timer Z Primary register

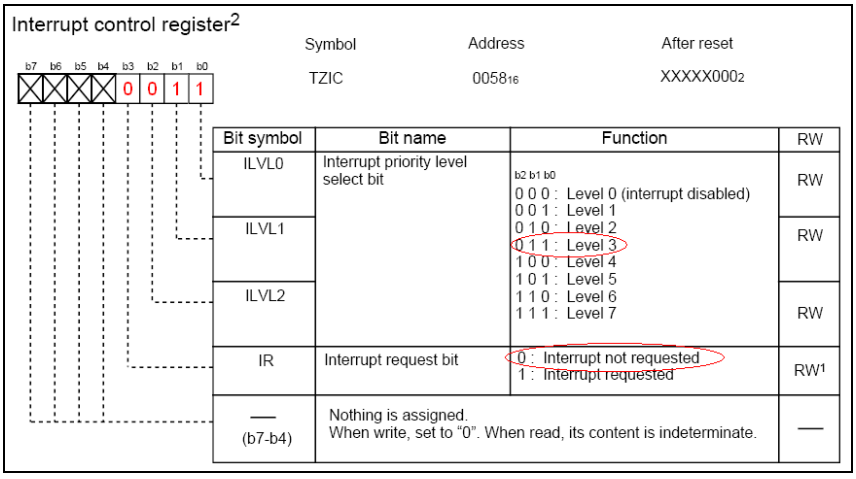
b7 b0

Symbol: TZPR | Address: 008716 | After reset: FF16

Mode	Function	Setting range	RW
Timer mode	Underflow of Prescaler Z is counted	0016 to FF16	RW
Programmable waveform generation mode	Underflow of Prescaler Z is counted	0016 to FF16	RW
Programmable one-shot generation mode	Underflow of Prescaler Z is counted (One-shot width is counted)	0016 to FF16	RW
Programmable wait one-shot generation mode	Underflow of Prescaler Z is counted (Wait period is counted)	0016 to FF16	RW

Il·lustració 54: TZPR

TZIC = 0x03;



Interrupt control register[2]

b7 b6 b5 b4 b3 b2 b1 b0: X X X X 0 0 1 1

Symbol: TZIC | Address: 005816 | After reset: XXXXX0002

Bit symbol	Bit name	Function	RW
ILVL0	Interrupt priority level select bit	b2 b1 b0 0 0 0 : Level 0 (interrupt disabled) 0 0 1 : Level 1 0 1 0 : Level 2 0 1 1 : Level 3 1 0 0 : Level 4 1 0 1 : Level 5 1 1 0 : Level 6 1 1 1 : Level 7	RW
ILVL1			RW
ILVL2			RW
IR	Interrupt request bit	0 : Interrupt not requested 1 : Interrupt requested	RW[1]
— (b7-b4)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—

Il·lustració 55: TZIC

Assignem un nivell d'interrupció per els temporitzadors.

Amb “TCSS” i ”TYZMR” organitzem els registres que comparteixen els temporitzadors Y i Z per tal que el temporitzador Z utilitzí el el temps del temporitzador Y per aconseguir un segon.

TCSS = 0x24;

Timer count source setting register

b7	b6	b5	b4	b3	b2	b1	b0
0	0	1	0	0	1	0	0

Symbol: TCSS — Address: 008E16 — After reset: 0016

Bit symbol	Bit name	Function	RW
TXCK0	Timer X count source select bit[1]	b1 b0 0 0 : f1 0 1 : f8 1 0 : f32 1 1 : f2	RW
TXCK1			RW
TYCK0	Timer Y count source select bit[1]	b3 b2 0 0 : f1 0 1 : f8 1 0 : fRING 1 1 : Selects input from CNTR1 pin	RW
TYCK1			RW
TZCK0	Timer Z count source select bit	b5 b4 0 0 : f1 0 1 : f8 1 0 : Selects Timer Y underflow 1 1 : f2	RW
TZCK1			RW
— (b7-b6)	Reserved bit	Set to “0”	RW

Il·lustració 56: TCSS

TYZMR = 0x88;

/* REGISTROS DE ENVIO */

Per fer la comunicació amb el PC necessitem fer servir la UART, que es com anomena el microcontrolador al conjunt de registres d’entrades i sortides destinades a la comunicació sèrie.

Es pot treballar amb la UART0 o la UART1, nosaltres ho fem amb la UART0.

/* UART0 */

Aquí es determina la forma de treball de la UART, nosaltres hem escollit treballar amb la font interna f1 a 9600 b/s, 8 bits de dades, 2 bits de stop i sense paritat, i utilitzant TXEPT per saber l’estat d’enviament.

Una vegada establert sabem com volem treballar, cal modificar els registres per aquesta configuració y fer el càlcul per saber com establir la velocitat de 9600 b/s.

$$Formula: Tc = \frac{fj}{16 \cdot (n+1)}$$

on: $fj = 20MHz \cdot f1 = 20MHz$, Tc = 9600 i n = U0BRG que pot ser de 0x00 fins a 0xFF

$$9600 = \frac{20MHz}{16 \cdot (n+1)};$$

$$n = \frac{20MHz}{9600 \cdot 16} - 1 = \boxed{130,2}$$

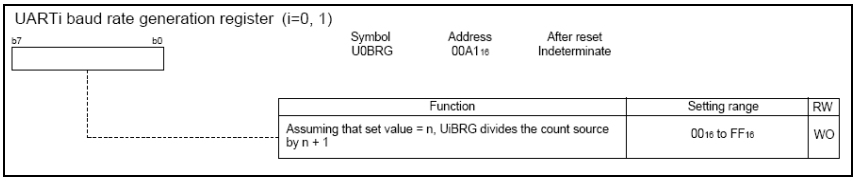
Al no sortir un numero decimal, provarem quin es el mes adequat:

$$n = 131 = 0x83 \rightarrow \frac{20MHz}{16\cdot(131+1)} = 9469,69$$

$$n = 130 = 0x82 \rightarrow \frac{20MHz}{16\cdot(130+1)} = 9541,98$$

$$n = 129 = 0x81 \rightarrow \frac{20MHz}{16\cdot(129+1)} = \boxed{9615,38}$$

U0BRG=0X81;



Il·lustració 57: U0BRG

UARTi transmit/receive mode register (i=0, 1)

b7 b6 b5 b4 b3 b2 b1 b0: 0

Symbol U0MR | Address 00A016 | After reset 0016

Bit symbol	Bit name	Function	RW
SMD0	Serial I/O mode select bit	b2 b1 b0 0 0 0 : Serial I/O disabled 0 0 1 : Clock synchronous serial I/O mode 1 0 0 : UART mode transfer data 7 bits long 1 0 1 : UART mode transfer data 8 bits long 1 1 0 : UART mode transfer data 9 bits long Must not be set except above	RW
SMD1			RW
SMD2			RW
CKDIR	Internal/external clock select bit	0 : Internal clock 1 : External clock	RW
STPS	Stop bit length select bit	0 : One stop bit 1 : Two stop bits	RW
PRY	Odd/even parity select bit	Effective when PRYE = 1 0 : Odd parity 1 : Even parity	RW
PRYE	Parity enable bit	0 : Parity disabled 1 : Parity enabled	RW
(b7)	Reserved bit	Must set to "0"	RW

Il·lustració 58: U0MR

U0MR_bit.SMD0 = 1;
U0MR_bit.SMD1 = 0;
} Transmetem 8bits

U0MR_bit.SMD2 = 1;

U0MR_bit.CKDIR = 0; Definim rellotge intern

U0MR_bit.STPS = 1; Stop bit a 2

U0MR_bit.PRYE = 0; Paritat inhabilitada

UARTi transmit/receive control register 0 (i=0, 1)

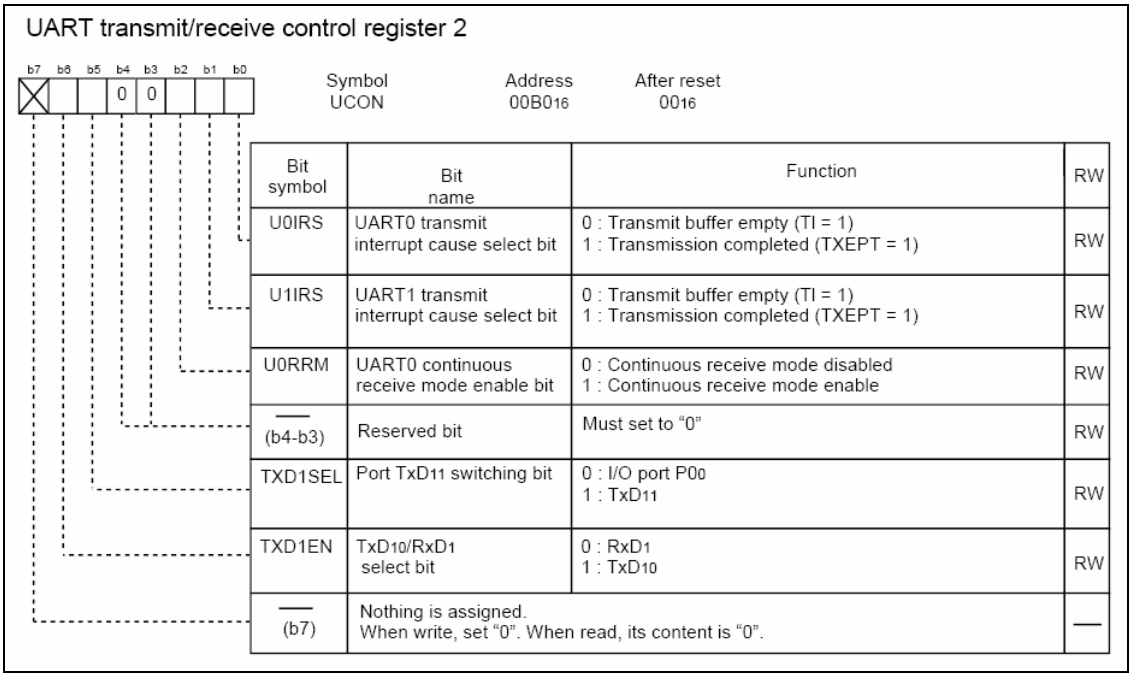
b7	b6	b5	b4	b3	b2	b1	b0
			X		0		

Symbol U0C0 — Address 00A4₁₆ — After reset 08₁₆

Bit symbol	Bit name	Function	RW
CLK0	BRG count source select bit	b1 b0 0 0 : f1SIO is selected 0 1 : f8SIO is selected 1 0 : f32SIO is selected 1 1 : Avoid this setting	RW
CLK1			RW
(b2)	Reserved bit	Must set to "0"	RW
TXEPT	Transmit register empty flag	0 : Data present in transmit register (during transmission) 1 : No data present in transmit register (transmission completed)	RO
(b4)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—
NCH	Data output select bit	0 : TxDi pin is CMOS output 1 : TxDi pin is N-channel open-drain output	RW
CKPOL	CLK polarity select bit	0 : Transmit data is output at falling edge of transfer clock and receive data is input at rising edge 1 : Transmit data is output at rising edge of transfer clock and receive data is input at falling edge	RW
UFORM	Transfer format select bit	0 : LSB first 1 : MSB first	RW

Il·lustració 59: U0C0

```
U0C0_bit.CLK0 = 0;
U0C0_bit.CLK1 = 0;      } f1
U0C0_bit.UFORM = 0;     Transmetem des del bit alt
```



UART transmit/receive control register 2

b7	b6	b5	b4	b3	b2	b1	b0
X			0	0			

Symbol: UCON — Address: 00B016 — After reset: 0016

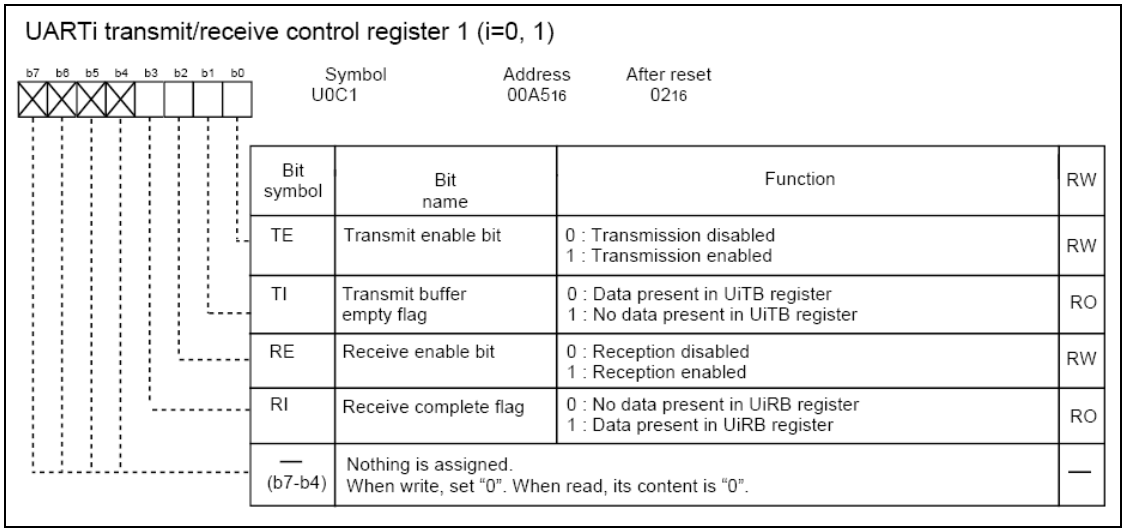
Bit symbol	Bit name	Function	RW
U0IRS	UART0 transmit interrupt cause select bit	0 : Transmit buffer empty (TI = 1) 1 : Transmission completed (TXEPT = 1)	RW
U1IRS	UART1 transmit interrupt cause select bit	0 : Transmit buffer empty (TI = 1) 1 : Transmission completed (TXEPT = 1)	RW
U0RRM	UART0 continuous receive mode enable bit	0 : Continuous receive mode disabled 1 : Continuous receive mode enable	RW
— (b4-b3)	Reserved bit	Must set to "0"	RW
TXD1SEL	Port TxD11 switching bit	0 : I/O port P00 1 : TxD11	RW
TXD1EN	TxD10/RxD1 select bit	0 : RxD1 1 : TxD10	RW
— (b7)	Nothing is assigned. When write, set "0". When read, its content is "0".		—

Il·lustració 60: UCON

UCON_bit.U0IRS = 1; Utilitzem TXEPT

UCON_bit.U0RRM=1; Recepció continua

UCON_bit.TXD1SEL = 1; Definim que P0.0 sigui TXD11



UARTi transmit/receive control register 1 (i=0, 1)

b7	b6	b5	b4	b3	b2	b1	b0
X	X	X	X	X			

Symbol: U0C1 — Address: 00A516 — After reset: 0216

Bit symbol	Bit name	Function	RW
TE	Transmit enable bit	0 : Transmission disabled 1 : Transmission enabled	RW
TI	Transmit buffer empty flag	0 : Data present in UiTB register 1 : No data present in UiTB register	RO
RE	Receive enable bit	0 : Reception disabled 1 : Reception enabled	RW
RI	Receive complete flag	0 : No data present in UiRB register 1 : Data present in UiRB register	RO
— (b7-b4)	Nothing is assigned. When write, set "0". When read, its content is "0".		—

Il·lustració 61: U0C1

U0C1_bit.TE = 1; Habilita la transmissió

U0C1_bit.RE = 0; Deshabilita la lectura

Habilitem les interrupcions per poder treballar amb ellas:

__enable_interrupt();

Una vegada configurat, creem un bucle infinit per tal de mantenir el programa en marcha.

```
        while(1);
}
```

Fora del programa principal definim que té que fer la interrupció.

```
#pragma vector = 24              Interrupció del Timer Z, s'executarà cada segon
__interrupt void TimerZ(void)
{
        ADCON0_bit.ADST = 1;        Començar la conversió A-D
        ValorAD = ADL;              ADL es on emmagatzema la conversió el micro
```

Per tenir un control visual, hem habilitat quatre led's que ens mostraran l'estat del ValorAD, quant més gran sigui el valor, mes led's encesos tindrem.

```
        if (ValorAD <= 0x1F)
        {
                P1=0x00;                    Led's OFF
        }
        else if ((ValorAD > 0x1F) && (ValorAD <= 0x3E))
        {
                P1 = 0x01; //PORT;          Led 0 ON
        }
        else if ((ValorAD > 0x5D) && (ValorAD <= 0x7C))
        {
                P1 = 0x03; //PORT;          Led's 0 i 1 ON
        }
        else if ((ValorAD > 0x9B) && (ValorAD <= 0xBA))
        {
                P1 = 0x07; //PORT;          Led's 0, 1 i 2 ON
        }
```

```
else if ((ValorAD > 0xD9) && (ValorAD <= 0xF8))
{
        P1 = 0x0F; //PORT;          Led's 0, 1, 2 i 3 ON
}
```

El led 6 ens informarà d'una nova conversió

```
if (P1_bit.P1_7==1)
{P1_bit.P1_7=0;}
else
{P1_bit.P1_7=1;}
```

Els valors convertits, encara no son vàlids per transmetre, els hem de seccionar i enviar per separat.

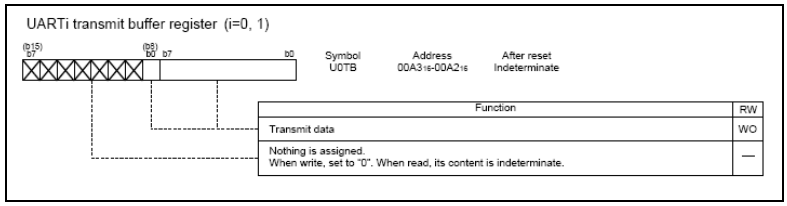
Utilitzant un **for** realitzem la següent operació quatre vegades, que serà el número de digits que enviarem.

```
i=0;
for(i=0;i<=3;i++)
{
        NT=ValorAD/DIV;                 DIV=0x3E8=1000
        Trama[i]=(unsigned char)NT;  Guardem els valors de NT a Trama
        ValorAD=ValorAD-(NT*DIV);       Traiem el valor ja enregistrat
        DIV=DIV/0x0A;                   Traiem un 0 a DIV
        Trama[i]=(Trama[i]+0x30);       Passem el valor de Trama al codi ACII
}
i=0;            Inicialitzem les variables als seus valors
DIV=0x3E8;
```

Enviem Trama un per un a l'ordinador fent servir un retard.

```
for(i=0;i<=3;i++)
{
        while(U0C0_bit.TXEPT == 0){}
        do
        {
```

```
            a++;
            b=0;
            do
            {b++;}
            while(b<30000);
        }
        while(a<49);

        U0TB=Trama[i];          Enviem Trama
```



Il·lustració 62: U0TB

```
    }
```

El led 5 ens informarà d'una nova dada transmesa

```
     if (P1_bit.P1_6==1)
     {P1_bit.P1_6=0;}
     else
     {P1_bit.P1_6=1;}
}
```

Aquesta funció es el primer que s'executa i configura els registres del clock

```
unsigned char __low_level_init(void)
{
```

PRCR = 0x01; Traiem la protecció per poder escriure en CM0, CM1 i OCD

Protect register

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	1

Symbol: PRCR — Address: 000A16 — After reset: 00XXX0002

Bit symbol	Bit name	Function	RW
PRC0	Protect bit 0	Enable write to CM0, CM1, OCD, HR0, HR1 registers 0 : Write protected 1 : Write enabled	RW
PRC1	Protect bit 1	Enable write to PM0, PM1 registers 0 : Write protected 1 : Write enabled	RW
PRC2	Protect bit 2	Enable write to PD0 register 0 : Write protected 1 : Write enabled	RW
PRC3	Protect bit 3	Enable write to VCR2, D4INT registers 0 : Write protected 1 : Write enabled	RW
(b5-b4)	Reserved bit	When write, should set to "0"	RW
(b7-b6)	Reserved bit	When read, its content is "0".	RO

Il·lustració 63: PRCR

CM0 = 0x08;

System clock control register 0

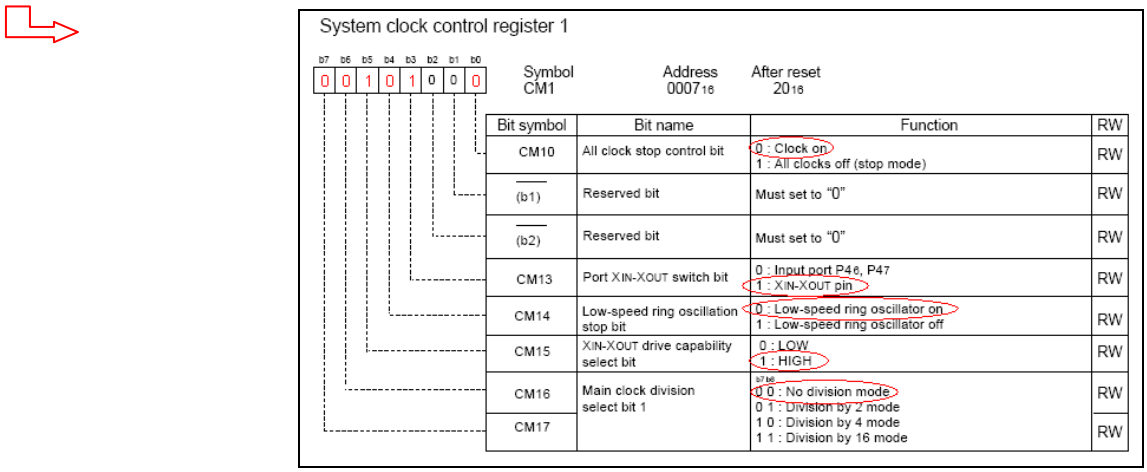
b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	1	0	0	0

Symbol: CM0 — Address: 000616 — After reset: 6816

Bit symbol	Bit name	Function	RW
(b1-b0)	Reserved bit	Set to "0"	RW
CM02	WAIT peripheral function clock stop bit	0 : Do not stop peripheral function clock in wait mode 1 : Stop peripheral function clock in wait mode	RW
(b3)	Reserved bit	Set to "1"	RW
(b4)	Reserved bit	Set to "0"	RW
CM05	Main clock (XCIN-XCOUT) stop bit	0 : On 1 : Off	RW
CM06	Main clock division select bit 0	0 : CM16 and CM17 valid 1 : Divide-by-8 mode	RW
(b7)	Reserved bit	Set to "0"	RW

Il·lustració 64: CM0

```
CM1  = 0x28;
```



System clock control register 1

b7	b6	b5	b4	b3	b2	b1	b0
0	0	1	0	1	0	0	0

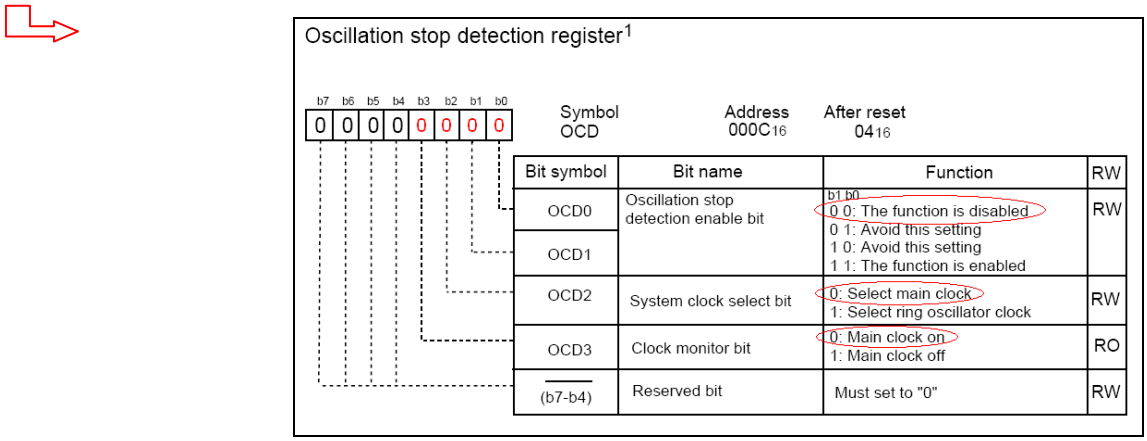
Symbol: CM1 — Address: 0007₁₆ — After reset: 20₁₆

Bit symbol	Bit name	Function	RW
CM10	All clock stop control bit	0 : Clock on 1 : All clocks off (stop mode)	RW
(b1)	Reserved bit	Must set to "0"	RW
(b2)	Reserved bit	Must set to "0"	RW
CM13	Port XIN-XOUT switch bit	0 : Input port P46, P47 1 : XIN-XOUT pin	RW
CM14	Low-speed ring oscillation stop bit	0 : Low-speed ring oscillator on 1 : Low-speed ring oscillator off	RW
CM15	XIN-XOUT drive capability select bit	0 : LOW 1 : HIGH	RW
CM16	Main clock division select bit 1	b7 b6 0 0 : No division mode 0 1 : Division by 2 mode 1 0 : Division by 4 mode 1 1 : Division by 16 mode	RW
CM17			RW

Il·lustració 65: CM1

Control dels temporitzadors activat, oscil·lador extern, velocitat baixa, capacitat de funcionament del cristall alt i sense divisió del rellotge.

```
OCD  = 0x00;
```



Oscillation stop detection register[1]

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	0

Symbol: OCD — Address: 000C₁₆ — After reset: 04₁₆

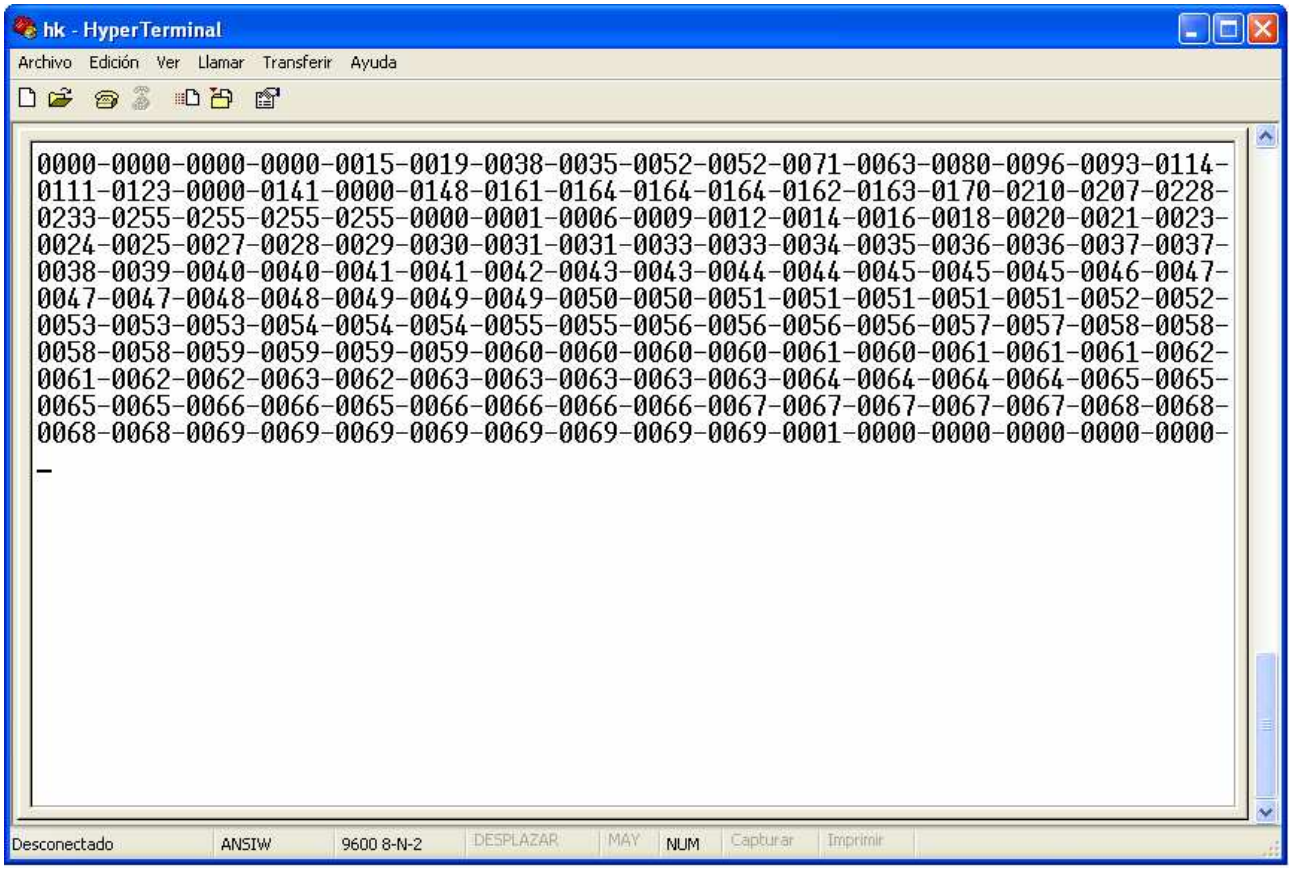
Bit symbol	Bit name	Function	RW
OCD0	Oscillation stop detection enable bit	b1 b0 0 0: The function is disabled 0 1: Avoid this setting 1 0: Avoid this setting 1 1: The function is enabled	RW
OCD1			
OCD2	System clock select bit	0: Select main clock 1: Select ring oscillator clock	RW
OCD3	Clock monitor bit	0: Main clock on 1: Main clock off	RO
(b7-b4)	Reserved bit	Must set to "0"	RW

Il·lustració 66: OCD

Oscil·lador sempre en funcionament i rellotge principal seleccionat i activat.

```
PRCR  = 0x00;          Protegim CM0, CM1 i OCD

return 1;

}
```

5.1.3. Visualització al HyperTerminal



Il·lustració 67: Resultats a l'HyperTerminal

Modificant una mica el programa, per tal de que al final de trama inclogui un guio, veiem les dades del sensor resistiu que hem anat variant. per tal que es tingui diferents valors.

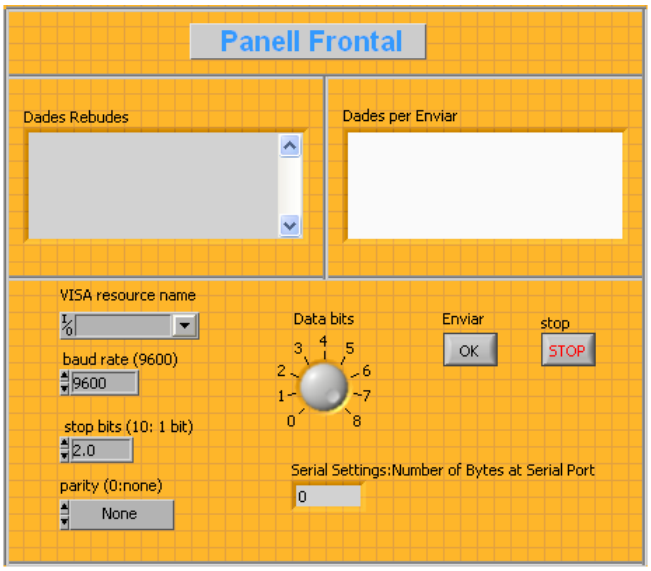
El HiperTerminal l'hem fet servir sobre tot per comprovar que tant la configuració com l'enviament de dades eren correctes. D'aquesta manera i una vegada feta la comprovació, ja podem passar a crear un programa amb el LabView on puguem tindre mes opcions.

5.1.4. Visualització al LabView

Es molt important,establir la mateixa configuració de comunicació en el panell frontal que la que tenim configurada per programa al microcontrolador. En el nostre cas fixem el volum de dades a 8 bits, el baud rate a 9600, 2 bits de stop i sense paritat. El port COM que tenim que escollir es pel que hem connectat la placa a l'ordinador, fent servir l'opció de refrescar, ens dirà quins estan disponibles.

En el panell frontal visualitzem les dades que ven arribant i tenim l'opció d'escriure dades per enviar-li al microcontrolador polsant OK.

El boto stop, atura la comunicació.



Il·lustració 68: Panell Frontal comunicació sèrie

- Al diagrama de blocs fem les connexions adients per lligar els blocs:

VISA serial. Ens serveix per crear la connexió amb les diferents opcions de comunicació.

VISA W. Serveix per enviar el que conte el string.

VISA R. Serveix mostrar el que rebem per canal RxD.

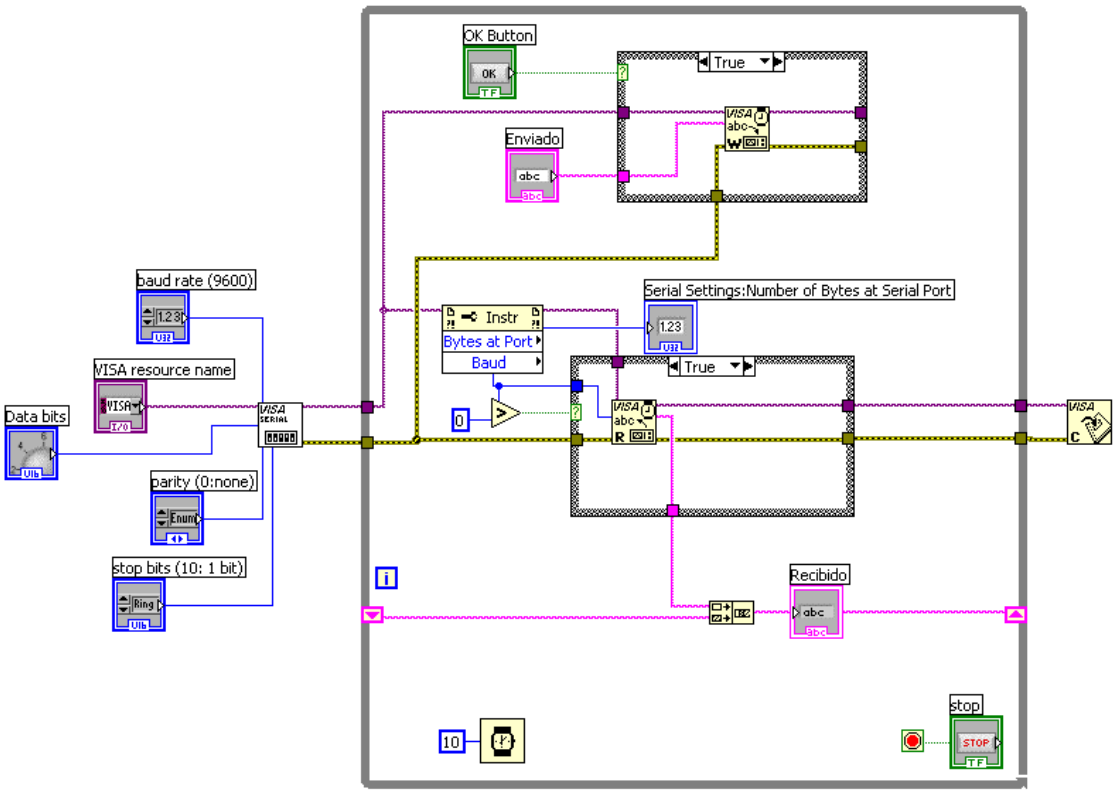
VISA C. Serveix tancar la connexió.

- Definició dels colors:

El blau indica que tracta dades decimals.

El verd valors booleans.

El rosa tracta amb textes

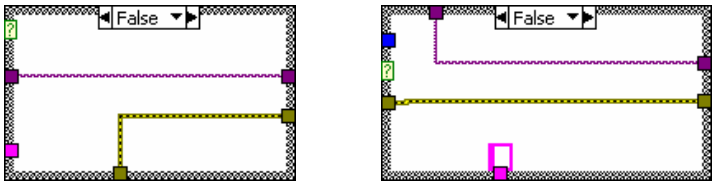


Il·lustració 69: Diagrama de blocs comunicació sèrie

- Funcions:

El rectangle exterior es un bucle que s'aturara quan es polsa l'estop.

Els dos interns son if. Depenent si la condició es compleix o no, actuara el true o el false.



5.2. Aplicacions CAN amb la placa 2

5.2.1. Utilització del Converso A/D

L'objectiu d'aquesta aplicació es la utilització del converso analògic – digital, per poder convertir dades analògiques, en aquest cas provinents del potenciòmetre de la placa, a digitals.

Es una aplicació que serà útil per si en un futur necessitem utilitzar el converso.

El programa segueix la mateixa estructura que el programa de l'aplicació del Timer A, així doncs començarem amb la instrucció del programa principal:

► void main(void)

Seguidament després d'aquesta instrucció cridem a les funcions que configuraran els registres corresponents al sistema analògic – digital. La primera funció que cridem es la següent:

► f_adcon0();

Aquesta funció s'ocupa de configurar el registre ADCON 0, que podem veure en la fiura següent:

Symbol	Address	After Reset	
ADCON0	00D6h	00h	
Bit Symbol	Bit Name	Function	RW
CH0	Analog input pin select bit	Refer to (4)	RW
CH1			RW
CH2			RW
MD	A/D operation mode select bit(2)	0 : On-shot mode 1 : Repeat mode	RW
ADGSEL0	A/D input group select bit(4)	0 : Selects port P0 group (AN0 to AN7) 1 : Selects port P1 group (AN8 to AN11)	RW
ADCAP	A/D conversion automatic start bit	0 : Starts in softw are trigger (ADST bit) 1 : Starts in timer RD (complementary PWM mode)	RW
ADST	A/D conversion start flag	0 : Disables A/D conversion 1 : Starts A/D conversion	RW
CKS0	Frequency select bit 0	[When CKS1 in ADCON1 register = 0] 0 : Select f4 1 : Select f2 [When CKS1 in ADCON1 register = 1] 0 : Select f1(3) 1 : Select fOCO-F	RW

Il·lustració 70: ADCON 0

Les dades analògiques que volem convertir a digital provenen del potenciòmetre de la placa, aquest esta connectat al canal AN8, per tant hem de configurar els bits CH0, CH1 i CH2 per seleccionar l'entrada analògica correcta. Les següents instruccions corresponen aquesta tasca:

- ch0=0
- ch1=0

→ Entrada analògica (AN8)

- ch2=1
- adgsel0=1

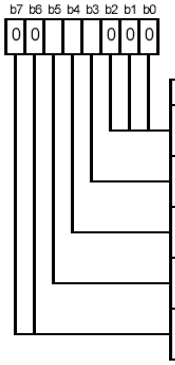
En aquesta funció també seleccionem la forma en que el timer començarà les conversions, mes concretament nosaltres seleccionem començar les instruccions per software, mitjançant el bit ADST. La instrucció per fer-ho es la següent:

- adcap=0;

La segona funció de configuració que cridem es la següent:

► f_adcon1()

Amb aquesta funció el tipus de conversió que volem, pot ser una conversió de 8 bits o be una de 10 bits. La següent il·lustració mostra el registre que configurem:



Symbol: ADCON1 — Address: 00D7h — After Reset: 00h

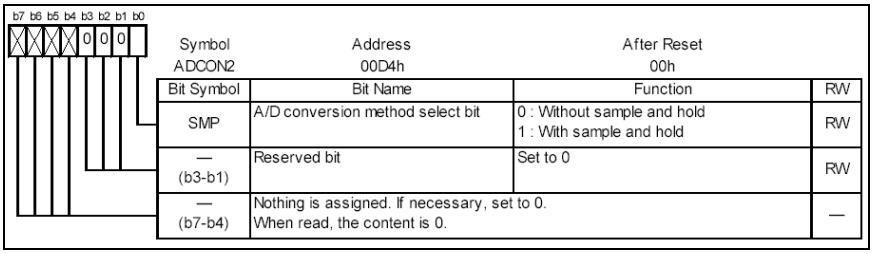
Bit Symbol	Bit Name	Function	RW
— (b2-b0)	Reserved bit	Set to 0	RW
BITS	8/10-bit mode select bit(2)	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	Refer to a description of the CKS0 bit in the ADCON0 register function	RW
VCUT	Vref connect bit(3)	0 : Vref not connected 1 : Vref connected	RW
— (b7-b6)	Reserved Bit	Set to 0	RW

Il·lustració 71: ADCON 1

Les instruccions per configurar el mode de conversió 8 bits, la freqüència de conversió i el voltatge de referència son les següents:

- bits=0
- cks1=0
- vcut=1

La última funció que cridem, configura el registra ADCON 2:



Symbol: ADCON2 — Address: 00D4h — After Reset: 00h

Bit Symbol	Bit Name	Function	RW
SMP	A/D conversion method select bit	0 : Without sample and hold 1 : With sample and hold	RW
— (b3-b1)	Reserved bit	Set to 0	RW
— (b7-b4)	Nothing is assigned. If necessary, set to 0. When read, the content is 0.		—

Il·lustració 72: ADCON 2

A la configuració d'aquest registre li correspon la següent instrucció:

- smp=1

Si posem el bit smp a 1, com en la instrucció anterior reduïm substancialment el numero de cicles de rellotge que tarda el converso en passar les dades d'analògic a digital, si posem un 0 en el bit esmentat tardarem mes cicles en fer la conversió però la conversió serà mes exacta.

Un cop configurats els registres pertinents, procedim a la creació d'un bucle infinit on inclourem un sistema de visió, del valor del registre on el converso emmagatzema el resultat de la conversió, mitjançant els quatre dels que disposem a la placa.

Les instruccions del programa són les següents:

- adst=1;

Amb aquesta instrucció iniciem la conversió analògica- digital del valor que rep per l'entrada corresponent el canal 8 del converso. El bit que activem per fer aquesta tasca correspon al registre ADCON 0.

Un cop la conversió ha finalitzat, les dades s'emmagatzemen al registre AD, del qual agafem la part baixa, que correspon a la variable adl.

Mitjançant les següents instruccions comparem el valor de la variable adl a una sèrie de valors compresos entre els valors hexadecimals 0x00 i 0xFF.

```
▪ if (adl<=0x1f)
        ▪{
        ▪LED0=LED_OFF;
        ▪LED1=LED_OFF;
        ▪ LED2=LED_OFF;
        ▪ LED3=LED_OFF;
        ▪ adst=0;
        ▪ }
```

En les instruccions anteriors, programem, per que el valor adl, si es menor que el valor 0x01f apagui els leds 0,1,2 i 3.

```
        ▪ else if ((adl>0x1f)&&(adl<=0x3e))
        ▪ {
        ▪ LED0=LED_ON;
        ▪ LED1=LED_OFF;
        ▪ LED2=LED_OFF;
        ▪ LED3=LED_OFF;
        ▪ adst=0;
        ▪ }
```

En les instruccions anteriors, programem, per que el valor adl, si es menor que el valor 0x01f apagui els leds 1,2 i 3 i encengui el led 1. Abans de sortir de la rutina de condició parem la conversió mitjançant el bit adst, simplement l'hem d'ajustar a 0.

```
        ▪ else if ((adl>0x5d)&&(adl<=0x7c))
        ▪ {
        ▪ LED0=LED_ON;
        ▪ LED1=LED_ON;
        ▪ LED2=LED_OFF;
```

▪ LED3=LED_OFF;

▪ adst=0;

▪ }

En les instruccions anteriors, la condició es que el valor adl estigui compres entre els valors 0x5d i 0x7c que estan en format hexadecimal, si aquesta condició es compleix s'encendran els leds 0 i 1, i s'apagaran els leds 2 i 3.programem. Com en l'apartat anterior, abans de sortir de la rutina de condició parem la conversió mitjançant el bit adst, simplement l'hem d'ajustar a 0.

▪ else if ((adl>0x9b)&&(adl<=0xba))

▪ {

▪ LED0=LED_ON;

▪ LED1=LED_ON;

▪ LED2=LED_ON;

▪ LED3=LED_OFF;

▪ adst=0;

▪ }

En les instruccions anteriors, la condició es que el valor adl estigui compres entre els valors 0x09 i 0xba que estan en format hexadecimal, si aquesta condició es compleix s'encendran els leds 0,1 i 1, i s'apagaran els led 3. Com en l'apartat anterior, abans de sortir de la rutina de condició parem la conversió mitjançant el bit adst, simplement l'hem d'ajustar a 0.leds 0 i 1, i s'apagaran els leds 2 i 3. Com en l'apartat anterior, abans de sortir de la rutina de condició parem la conversió mitjançant el bit adst, simplement l'hem d'ajustar a 0.

Per últim si el valor del registre adl esta entre 0xd9 i 0xf8 encendrem tots els leds.

▪ else if ((adl>0xd9)&&(adl<=0xf8))

▪ {

▪ LED0=LED_ON;

▪ LED1=LED_ON;

▪ LED2=LED_ON;

▪ LED3=LED_ON;

▪ adst=0;

▪ }

El programa que hem creat utilitza en definitiva els recursos de la placa, com son els leds, per visualitzar les dades digitals que hem utilitzat mitjançant el converso analògic – digital.

5.2.2. Utilització del Timer RA

L'objectiu d'aquesta aplicació és el d'utilitzar el Timer RA, en el mode d'operació "Timer mode", ja que ens pot ser d'utilitat per alguns programes on es requereixi algun tipus de mesura de temps.

El programa desenvolupat a continuació realitza la tasca de obrir i tancar un led, cada 1 segon aprox. Aquesta tasca el timer la realitzarà mitjançant una interrupció.

El programa anomenat TimerRA esta compost per diferents arxius, uns amb extensió ".h", aquests contenen les capçaleres de les funcions que utilitzarem, i els altres arxius tenen l'extensió ".c", aquests contenen les funcions i instruccions del programa.

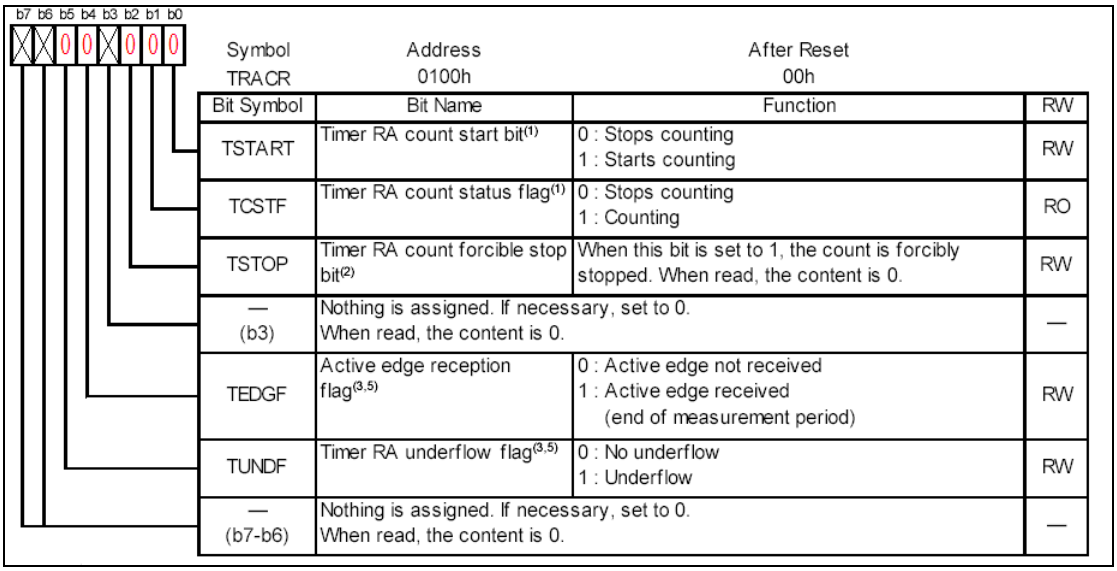
Comencem al programa principal representat en la següent instrucció;

► void main(void)

Des d'aquesta instrucció cridem les funcions de configuració del timer i també la funció que configurarà la interrupció del timerRA.

► f_tracr()

Aquesta funció configura el registre de control del Timer RA. El mostrem en la figura següent:



Symbol: TRACR — Address: 0100h — After Reset: 00h

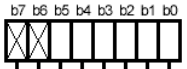
Bit Symbol	Bit Name	Function	RW
TSTART	Timer RA count start bit(1)	0 : Stops counting 1 : Starts counting	RW
TCSTF	Timer RA count status flag(1)	0 : Stops counting 1 : Counting	RO
TSTOP	Timer RA count forcible stop bit(2)	When this bit is set to 1, the count is forcibly stopped. When read, the content is 0.	RW
— (b3)	Nothing is assigned. If necessary, set to 0. When read, the content is 0.		—
TEDGF	Active edge reception flag(3,5)	0 : Active edge not received 1 : Active edge received (end of measurement period)	RW
TUNDF	Timer RA underflow flag(3,5)	0 : No underflow 1 : Underflow	RW
— (b7-b6)	Nothing is assigned. If necessary, set to 0. When read, the content is 0.		—

Il·lustració 73: TRACR

Inicialment posem tots els bits del registre a zero, ja que encara no volem que comenci a contar (bit0=0).

► f_traic()

Aquesta funció configura el registre de control del Timer RA. El mostrem en la figura següent:



Symbol: TRAIOC — Address: 0101h — After Reset: 00h

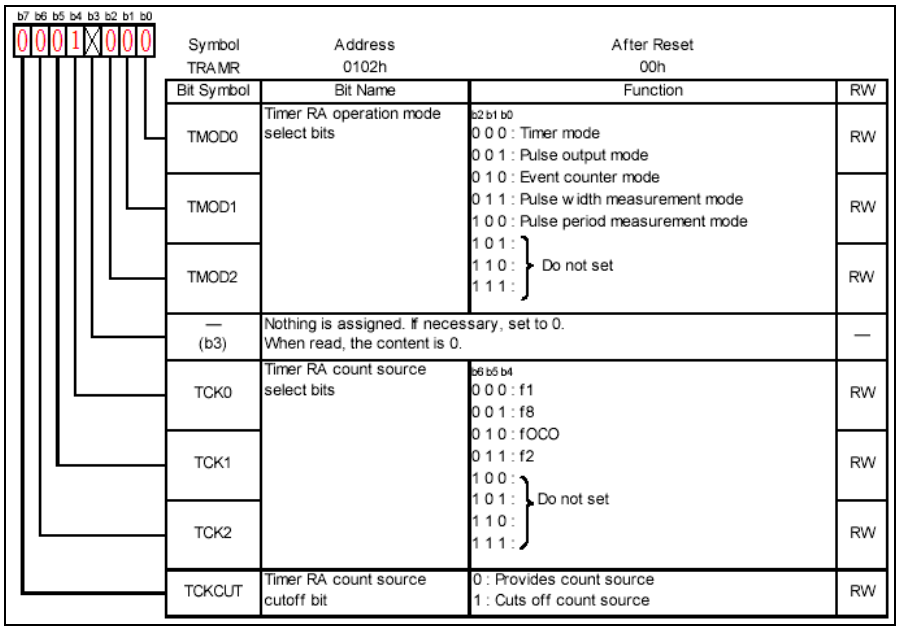
Bit Symbol	Bit Name	Function	RW
TEDGSEL	TRAIO polarity switch bit	Function varies depending on operation mode	RW
TOPCR	TRAIO output control bit		RW
TOENA	TRAO output enable bit		RW
TIOSEL	INT1/TRAIO select bit		RW
TIPF0	TRAIO input filter select bits		RW
TIPF1			RW
— (b7-b6)	Nothing is assigned. If necessary, set to 0. When read, the content is 0.		—

Il·lustració 74: TRAIOC

Aquest registra s'utilitza per, i per aixo el posem tot a zero.

► f_tramr()

Aquesta funció configura el registre de modes d'operació del Timer RA. El mostrem en la figura següent:



Symbol: TRAMR — Address: 0102h — After Reset: 00h

Bit Symbol	Bit Name	Function	RW
TMOD0	Timer RA operation mode select bits	b2 b1 b0 0 0 0 : Timer mode 0 0 1 : Pulse output mode 0 1 0 : Event counter mode 0 1 1 : Pulse width measurement mode 1 0 0 : Pulse period measurement mode 1 0 1 : } Do not set 1 1 0 : 1 1 1 :	RW
TMOD1			RW
TMOD2			RW
— (b3)	Nothing is assigned. If necessary, set to 0. When read, the content is 0.		—
TCK0	Timer RA count source select bits	b6 b5 b4 0 0 0 : f1 0 0 1 : f8 0 1 0 : fOCO 0 1 1 : f2 1 0 0 : } Do not set 1 0 1 : 1 1 0 : 1 1 1 :	RW
TCK1			RW
TCK2			RW
TCKCUT	Timer RA count source cutoff bit	0 : Provides count source 1 : Cuts off count source	RW

Il·lustració 75: TRAMR

Posant els bits TMOD, TMOD1 i TMOD2, seleccionem el mode de treball del Timer RA, en concret el "Timer mode".

Els bits restants, els configurem amb els següents valors; els bits TCK1 i TCK2 a 0 i el bit TCK0 el posem a 1, d'aquesta manera dividirem la font de rellotge per 8.

Les instruccions que realitzen aquesta tasca són:

- tmod1_tramr=0
- tmod2_tramr=0

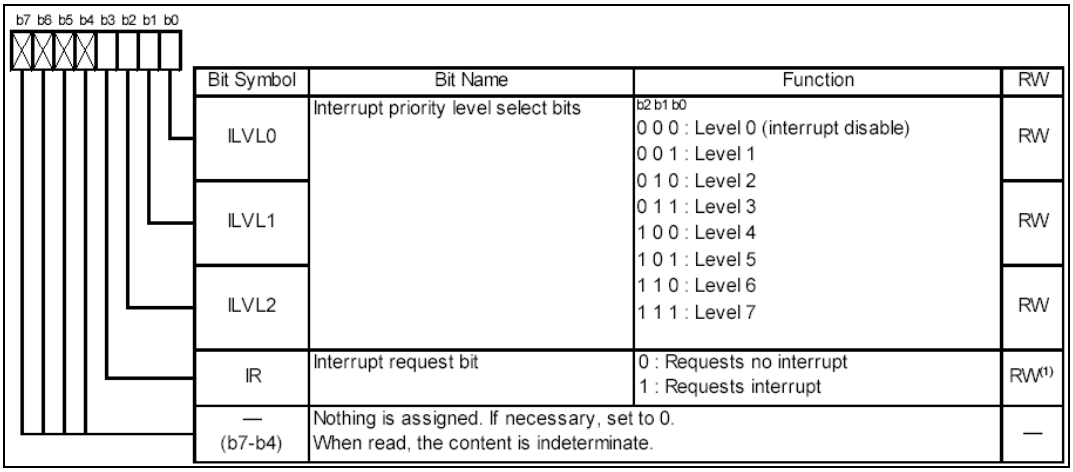
→ Mode d'operació.

- tmod0_tramr=0

- tck0_tramr=1
- tck1_tramr=0 → Divisor de la font de rellotge.
- tck2_tramr=0
- tckcut_tramr=0

► Interrupcio_TimerRA()

En aquesta funció configurarem tant el nivell de prioritat de la interrupció del timer RA, com l'habilitació de la interrupció pròpiament esmentada. Els registres que necessitem configurar es mostren en la següent figura:



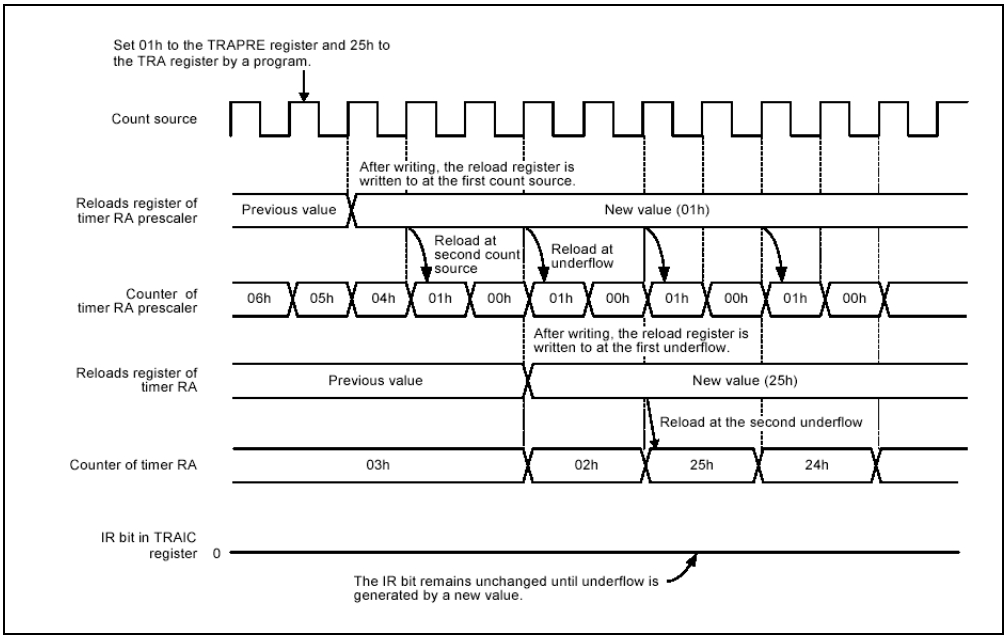
Bit Symbol	Bit Name	Function	RW
ILVL0	Interrupt priority level select bits	b2 b1 b0 0 0 0 : Level 0 (interrupt disable) 0 0 1 : Level 1 0 1 0 : Level 2 0 1 1 : Level 3 1 0 0 : Level 4 1 0 1 : Level 5 1 1 0 : Level 6 1 1 1 : Level 7	RW
ILVL1			RW
ILVL2			RW
IR	Interrupt request bit	0 : Requests no interrupt 1 : Requests interrupt	RW(1)
— (b7-b4)	Nothing is assigned. If necessary, set to 0. When read, the content is indeterminate.		—

Il·lustració 76: Registre de control d'interrupció

La funció Interrupcio_TimerRA configurarà els bits del registres de la il·lustració, perquè la interrupció del timer RA tingui nivell de prioritat 3. També habilitarem les interrupcions posant un 1 al bit IR. Les instruccions per configurar son les següents:

- ilvl0_traic=1
- ilvl1_traic=0 → nivell d'interrupció.
- ilvl2_traic=0
- ir_traic=1 → Habilita la resposta a la interrupció.

Exemple de l'operació de rescriure el valor de compte del timer A.



Il·lustració 77: Organigrama del Timer RA

El bit TSTART del registre TRACR es l'encarregat de inicia el decrement del valor del timer. El valor del timer es configura des dels registres de recarrega del " timer RA precaler" i " tmer RA", quan aquests valors son recarregats es transmeten als registres " timer RA precaler" i " tmer RA" amb un retard de 2 cicles de rellotge aproximadament.

Quan el valor del registre TRA, que es va decrementant, arriba a 0, la interrupció associada del timer RA s'activa.

5.2.3. Comunicació CAN

5.2.3.1. Enviament de dades del R8C/23 al LabView

El programa principal consta de dos parts. Una part es la programació en ANSI C mitjançant el compilador del microchip perque acondicioni les dades del sensor, creï un missatge que compleixi el protocol CAN amb les dades dels sensor en format hexadecimal i les enviï al bus CAN. L'altre part esta programada mitjançant LabView i el programa creat ens permet monotoritzar les dades que rebem del microchip per mitja del bus CAN, per fer-ho el programa LabView utilitza la PCMECIA NICAN2.2.

L'organigrama següent mostra gràficament el funcionament de l'aplicació:

Il·lustració 78: Organigrama comunicació CAN

El primer que hem fet es incloure els arxius que contenen les funcions que el programa principal utilitzarà, això ho farem de la següent manera:

```
#include "sfr_r823.h"

#include "rskR8C23def.h"

#include "CANte.h"

#include "ad.h"

#include "main.h"
```

En l'arxiu "CANte.h", hi ha l'estructura d'un missatge estàndard, aquest en ha servirà per emmagatzemar el missatges que rebem i també per crear-nos de nous i així poder-los enviar.

Després d'incloure els arxius.h, hem de definir les variables que farem servir en el programa, en la següent taula es pot observar les principals variables que declarem i una descripció de la funció que tindran.

variables	Descripció
struct SMSCAN sensor1;	La variable sensor es del tipus SMSCAN, i contindra tres camps;id,lengh i dada
struct SMSCAN *s1;	Aquesta variable es un punter que apunta a una estructura SMSCAN
unsigned short nm=0;	Aquesta variable representa el numero de missatge, pot anar de 0 a 15.
int lp_dlc=4;	
int n;	

Taula 20: Variables

Declarades totes les variables, procedim a escriure el programa. El programa principal està precedit per la seguent declaració;

```
void main (void){
```

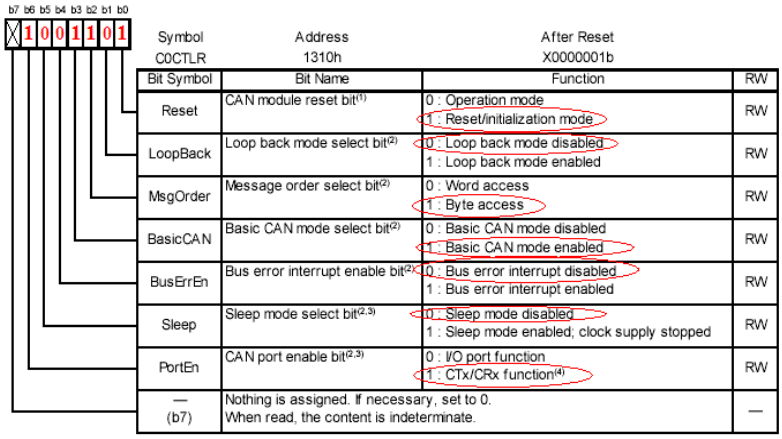
Necessitem fer que el punter declarat com s1 apunti a l'estructura declarada com a sensor1, d'aquesta manera podrem passar a les funcions que ho necessitin, la variable punter s1 com a paràmetre. Això ho aconseguim mitjançant la següent instrucció.

```
s1=&sensor1;
```

Primer inicialitzem les variables, cridem les funcions que estan relacionades amb l'inicialitzacio de bits, registres del mòdul CAN, i el convertidor analògic/digital. La primera funció que cridem es la següent:

```
Configuracio()
```

El registre C0CTLR esta directament relacionat amb la funció configuració. La següent figura mostra aquest registre:



Symbol	Address	After Reset
C0CTLR	1310h	X0000001b

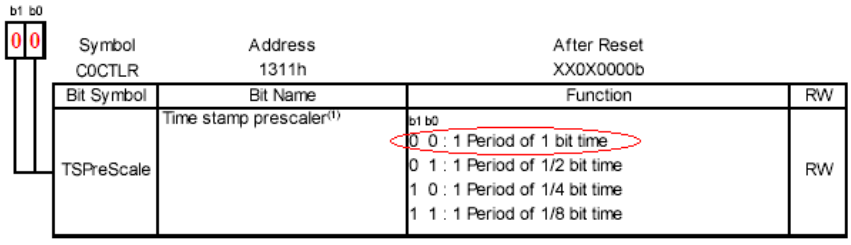
Bit Symbol	Bit Name	Function	RW
Reset	CAN module reset bit(1)	0 : Operation mode 1 : Reset/initialization mode	RW
LoopBack	Loop back mode select bit(2)	0 : Loop back mode disabled 1 : Loop back mode enabled	RW
MsgOrder	Message order select bit(2)	0 : Word access 1 : Byte access	RW
BasicCAN	Basic CAN mode select bit(2)	0 : Basic CAN mode disabled 1 : Basic CAN mode enabled	RW
BusErrEn	Bus error interrupt enable bit(2)	0 : Bus error interrupt disabled 1 : Bus error interrupt enabled	RW
Sleep	Sleep mode select bit(2,3)	0 : Sleep mode disabled 1 : Sleep mode enabled; clock supply stopped	RW
PortEn	CAN port enable bit(2,3)	0 : I/O port function 1 : CTx/CRx function(4)	RW
— (b7)	Nothing is assigned. If necessary, set to 0. When read, the content is indeterminate.		—

Il·lustració 79: C0CTLR

Aquest registra el configurem a traves de les següents instruccions:

- c0ctlr_addr.b.reset = 1 → Portem a reset el mòdul CAN.
- c0ctlr_addr.b.sleep = 0 → Desactivem el mode d'operació "sleep".
- c0ctlr_addr.b.porten = 1 → Desactivem els ports i Activem la transmissió/recepció de missatges CAN.
- c0ctlr_addr.b.loopback = 0 → Desactivem el sistema de llaç tancat.
- c0ctlr_addr.b.msgorder = 1 → Ajustem l'accés tip byte.
- c0ctlr_addr.b.basicCAN = 1 → Habilitem el sistema "basic CAN".
- c0ctlr_addr.b.buserren = 0 → Inhabilitem les interrupcions del bus CAN.

La figura següent també correspon al registre C0CTLR.



Symbol	Address	After Reset
C0CTLR	1311h	XX0X0000b

Bit Symbol	Bit Name	Function	RW
TSPreScale	Time stamp prescaler(1)	b1 b0 0 0 : 1 Period of 1 bit time 0 1 : 1 Period of 1/2 bit time 1 0 : 1 Period of 1/4 bit time 1 1 : 1 Period of 1/8 bit time	RW

Utilitzem la següents instruccions per programar-lo:

- c0ctlr_addr.b.tsprescale = 0

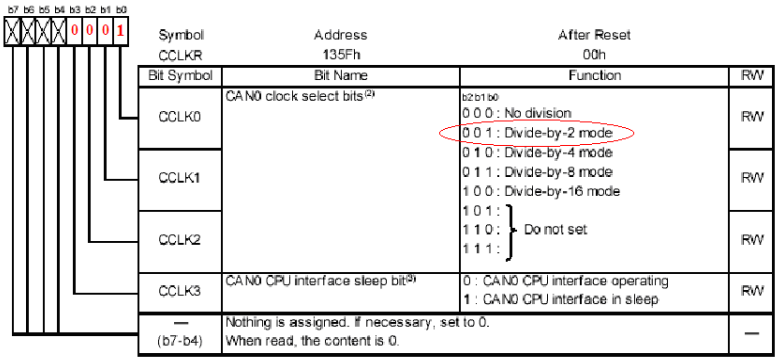
Dintre de la configuració, cridem la funció anomenada Baudrate.

Baudrate()

Que serveix per configurar la velocitat de les dades que circularen pel bus CAN.

Per aconseguir-ho aquesta funció ha d'ajustar els valors de dos registres:

El registre CCLKR. La figura següent mostra aquest registre.



Symbol	Address	After Reset
CCLKR	135Fh	00h

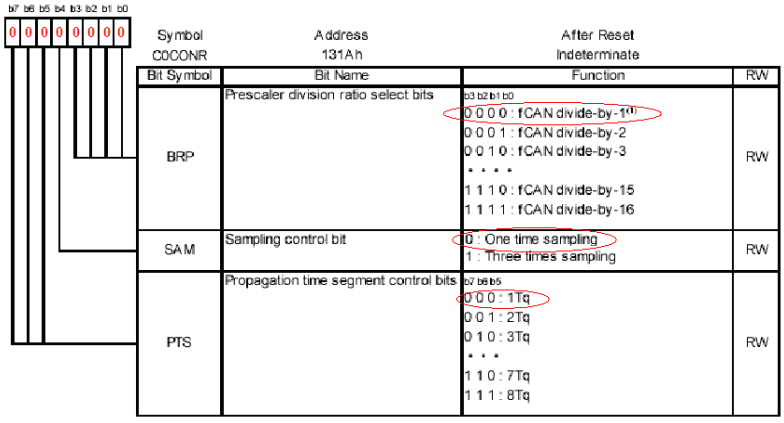
Bit Symbol	Bit Name	Function	RW
CCLK0	CAN0 clock select bits(2)	b2 b1 b0 0 0 0 : No division 0 0 1 : Divide-by-2 mode	RW
CCLK1		0 1 0 : Divide-by-4 mode 0 1 1 : Divide-by-8 mode 1 0 0 : Divide-by-16 mode	RW
CCLK2		1 0 1 : 1 1 0 : 1 1 1 : } Do not set	RW
CCLK3	CAN0 CPU interface sleep bit(3)	0 : CAN0 CPU interface operating 1 : CAN0 CPU interface in sleep	RW
— (b7-b4)	Nothing is assigned. If necessary, set to 0. When read, the content is 0.		—

Il·lustració 80: CCLKR

Les instruccions per ajusta el registra de la figura son les següents:

- prc0 = 1 → Inhabilita la protecció del registre, per poder-hi escriure.
- cclkr &= ~0x0F → Neteja el registre, mitjançant un and lògica.
- cclkr |= 0x01 → Dividim per dos la freqüència del rellotge.
- prc0 = 0 → Inhabilitem l'escriptura, habilitem altre vegada la protecció.

El registre C0CONR. Les figures següents contenen aquest registre.



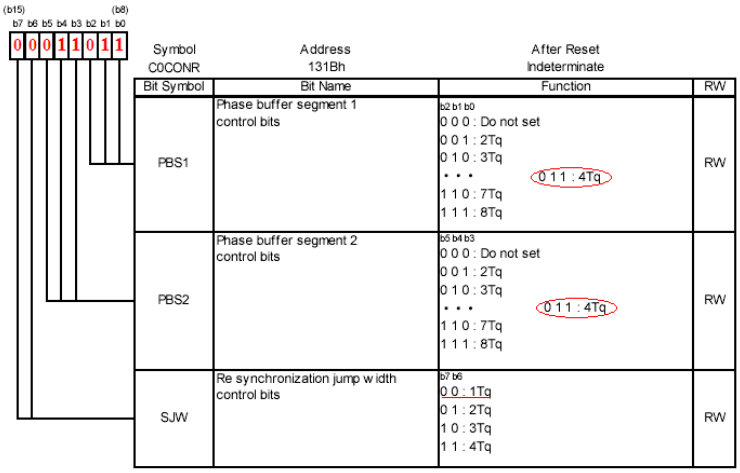
Symbol	Address	After Reset
C0CONR	131Ah	Indeterminate

Bit Symbol	Bit Name	Function	RW
BRP	Prescaler division ratio select bits	b3 b2 b1 b0 0 0 0 0 : fCAN divide-by-1(1) 0 0 0 1 : fCAN divide-by-2 0 0 1 0 : fCAN divide-by-3 • • • • 1 1 1 0 : fCAN divide-by-15 1 1 1 1 : fCAN divide-by-16	RW
SAM	Sampling control bit	0 : One time sampling 1 : Three times sampling	RW
PTS	Propagation time segment control bits	b7 b6 b5 0 0 0 : 1Tq 0 0 1 : 2Tq 0 1 0 : 3Tq • • • 1 1 0 : 7Tq 1 1 1 : 8Tq	RW

Il·lustració 81: C0CONR 1

Les instruccions per ajustar el registre de la figura son les següents:

- c0conr_addr.b.brp = 0
- c0conr_addr.b.sam = 0
- c0conr_addr.b.pts = 0 → Obtenim 1 Tq



Il·lustració 82: C0CONR 2

Les instruccions per ajustar el registre de la figura son les següents:

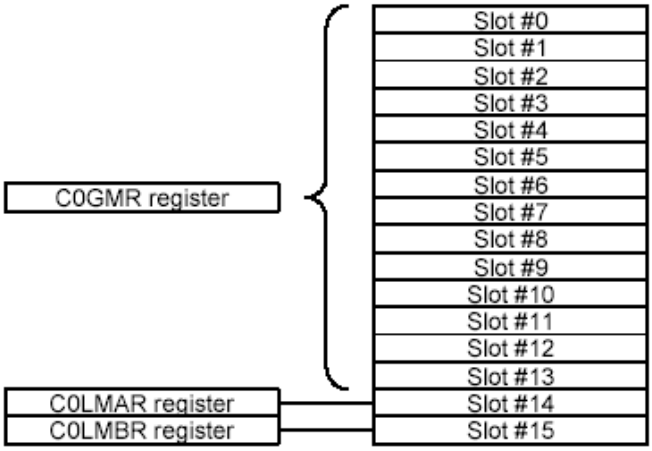
- c0conr_addr.b.pbs1 = 3 → Obtenim 4 Tq
- c0conr_addr.b.pbs2 = 3 → Obtenim 4 Tq
- c0conr_addr.b.sjw = 0 → Obtenim 1 Tq

Seguidament cridem la funció:

```
Filtre_id();
```

Aquesta funció esta relacionada amb els identificadors dels missatges que rebrem, depenen de l'argument que li passem podem rebre missatges amb qualsevol identificador o sols missatges amb un identificador en concret.

Es poden programar tants filtres com *slots* conte el "message box" del mòdul CAN. La següent figura mostra una relació entre els registres que conten la configuració del filtre i els *slots* del mòdul CAN d'aquest microcontrolador.



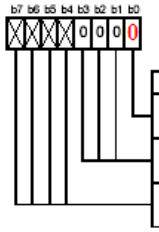
Nosaltres sols necessitem de moment permetre la recepció de qualsevol missatge amb qualsevol identificador, la instrucció que utilitzem es la següent:

- c0gmr.ba.sidh = (0x00>>6) & 0x1f → SID 10-6
- c0gmr.ba.sidl = 0x00 & 0x3f → SID 5-0

La ultima instrucció que cridarem per inicialitzar registres, esta relacionada amb el convertidor analògic/digital;

ad_config();

En aquesta funció ens encarreguem d'ajustar tres registres, tot seguit mostrem de quines es tracten mitjançant les seves respectives figures.



Symbol: ADCON2 | Address: 00D4h | After Reset: 00h

Bit Symbol	Bit Name	Function	RW
SMP	A/D conversion method select bit	0 : Without sample and hold 1 : With sample and hold	RW
— (b3-b1)	Reserved bit	Set to 0	RW
— (b7-b4)	Nothing is assigned. If necessary, set to 0. When read, the content is 0.		—

Il·lustració 83: ADCON2

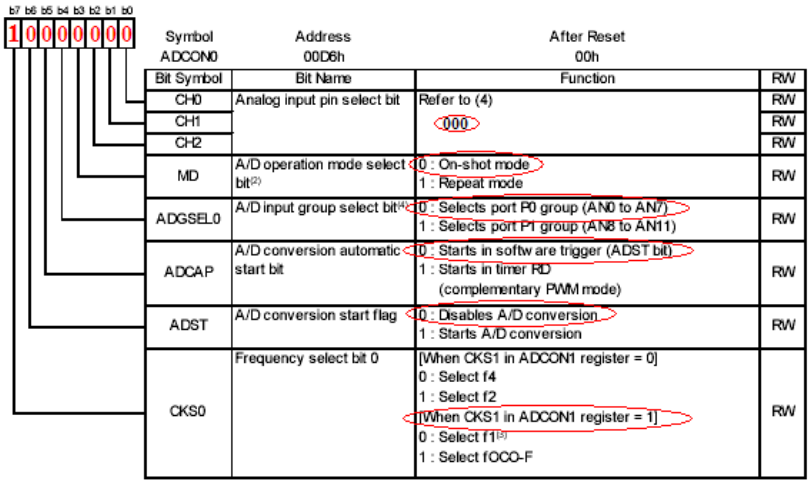
- adcon2=0x01

Symbol: ADCON1 | Address: 00D7h | After Reset: 00h

Bit Symbol	Bit Name	Function	RW
— (b2-b0)	Reserved bit	Set to 0	RW
BITS	8/10-bit mode select bit(2)	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	Refer to a description of the CKS0 bit in the ADCON0 register function	RW
VCUT	Vref connect bit(3)	0 : Vref not connected 1 : Vref connected	RW
— (b7-b6)	Reserved Bit	Set to 0	RW

Il·lustració 84: ADCON1

- adcon1=0x20



b7 b6 b5 b4 b3 b2 b1 b0	Symbol	Address	After Reset
1 0 0 0 0 0 0 0	ADCON0	00D6h	00h

Bit Symbol	Bit Name	Function	RW
CH0	Analog input pin select bit	Refer to (4)	RW
CH1		000	RW
CH2			RW
MD	A/D operation mode select bit(2)	0 : On-shot mode 1 : Repeat mode	RW
ADGSEL0	A/D input group select bit(4)	0 : Selects port P0 group (AN0 to AN7) 1 : Selects port P1 group (AN8 to AN11)	RW
ADCAP	A/D conversion automatic start bit	0 : Starts in softw are trigger (ADST bit) 1 : Starts in timer RD (complementary PWM mode)	RW
ADST	A/D conversion start flag	0 : Disables A/D conversion 1 : Starts A/D conversion	RW
CKS0	Frequency select bit 0	[When CKS1 in ADCON1 register = 0] 0 : Select f4 1 : Select f2 [When CKS1 in ADCON1 register = 1] 0 : Select f1(3) 1 : Select fOCO-F	RW

Il·lustració 85: ADCON0

- adcon0=0x01

Un cop configurat tot els registres, mitjançant les seves respectives funcions, tenim el programa preparat per complir la part mes important del projecte. Aquesta es repetirà indefinidament mitjançant un bucle, la següent instrucció es la encarregada de portar a terme aquesta tasca.

```
while(1)
```

Aquest bucle, igual que la funció *main*, conte dos còrtexs, un obert i un altre tancat. Dins d'aquests s'aniran executen el codi, el qual es repetirà eternament.

La primera instrucció que te el bucle activa el converso analògic - digital, mitjançant un bit del registre ADCON 0:

- adst=1

Un cop iniciada la conversió ja tenim les primeres dades del sensor, el pròxim pas consisteix en posar-les dins d'un missatge CAN:

- sensor1.id=1→Donem un valor al identificador del nostre missatge.
- sensor1.dlc=2 →Posem el número de dades.
- sensor1.data[0]=adl →Aquí carreguem la part alta de la conversió.
- sensor1.data[1]=adh →Aquí carreguem la part baixa de la conversió.

Un cop tenim el missatge format amb les seves corresponents dades, aquest ja esta preparat per ésser enviat, això ho aconseguim criden la funció següent:

```
Transmetre(1,s1);
```

Aquesta funció com el seu propi nom indica s'ocupa d'enviar els missatges CAN, per aconseguir-ho necessita que li donin dos paràmetres; un punter que apunti al missatge que volem enviar i el numero de l'*slot* des de el qual es vol enviar el missatge.

Aquest programa li passa el punter S1, que com hem explicat anteriorment conte l'adreça de l'estructura SMSCAN de la variable *sensor1*, la qual te les dades del converso.

El primer que fem es comprovar la disponibilitat del mòdul CAN per poder transmetre missatge, això ho fem mitjançant la següent instrucció:

- while(c0mctl[in_slot].transmit.trmactive){}

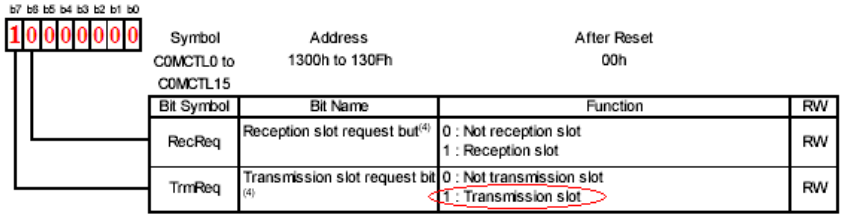
Aquesta instrucció va consultant el bit "trmActive" del registre C0MCTL, de manera que es queda en un bucle infinit fins que l'esmenta't bit es posa a 0 i sortim del bucle. Això significa el mòdul CAN ja ha acabat de transmetre i ja esta preparat per transmetre de nou.

El següent pas consisteix en carregar als registres que el mòdul CAN utilitza per enviar missatges al bus, amb els valors que hem passat com a paràmetres a la funció.

Les següents instruccions son les encarregades de fer aquesta tasca:

- c0slot[in_slot].ba.sidh = ((in_trm_data->id)>>6) & 0x1F
- c0slot[in_slot].ba.sidl = (in_trm_data->id) & 0x3F
- c0slot[in_slot].ba.dlc = in_trm_data->dlc
- for(lp_dlc=0; lp_dlc<(in_trm_data->dlc); ++lp_dlc){
- c0slot[in_slot].ba.data[lp_dlc] = in_trm_data->data[lp_dlc];}

Finalment ens queda donar la ordre al mòdul CAN perquè enviï les dades que te als registres C0slot [1], això ho fa amb el bit TrmReq del registre C0CMTL. En la figura següent es pot observar aquesta acció.



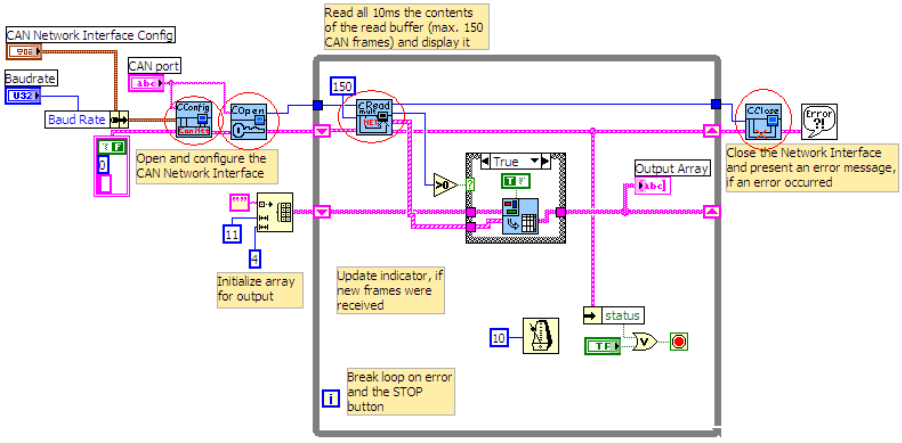
Il·lustració 86: C0CMTL

La instrucció que mostrem a continuació actua directament en el bit mostrar en la figura anterior:

- c0mctl[in_slot].byte=0x80

5.2.3.1.1. VISUALITZACIÓ AL LABVIEW

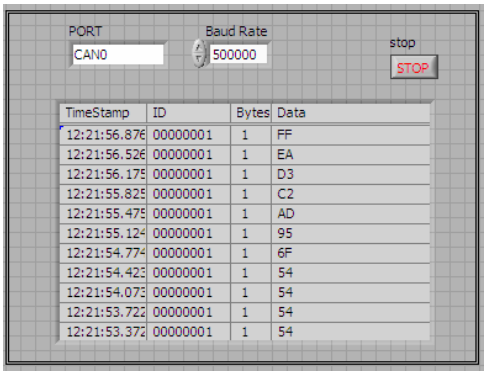
Labwieb es el programa que fem servir per visualitzar les dades que ens envia el microcontrolador Renesas mitjançant el bus CAN. Per realitzar-lo hem seguit una seria de passos:



Il·lustració 87: Diagrama de Blocs

En aquesta figura es pot apreciar l'esquemàtic que utilitza el "lectura.vi" per llegir les dades. Per poder-lo crear s'han d'instal·lar el NICAN 2.0 de National Instruments, on es poden trobar les llibreries necessàries per crear l'esquemàtic de la figura, més concretament podem observar que els símbols encerclats son els que necessiten les esmentades llibreries.

Les dades es podran observar en el següent pannell:



Il·lustració 88: Panell Frontal

En aquest panell, hi ha alguns punts a comentar:

PORT: Aquí es posa el port del qual vols llegir les dades, tenim dos possibilitats, port 0 o el port 1, ja que la PCMCIA sols te dos ports.

Baud rate: Hem d'especificar baudrate en Kbs de les dades que circulen pel bus i que nosaltres volem llegir.

TimeStamp: Es la l'hora en que rep les dades.

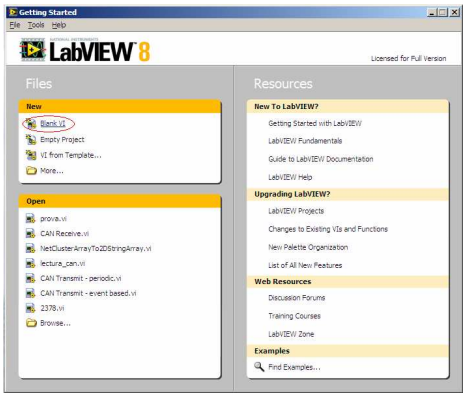
ID: Es l'identificador de la trama, pot tenir 11bits, es a dir que va de 0 a 2047 possibles identificadors.

Bytes: Conte la mida o numero de dades que contindrà el missatge, en aquesta cas dos.

Data: Aquí podem llegir les dades que estem reben del bus CAN.

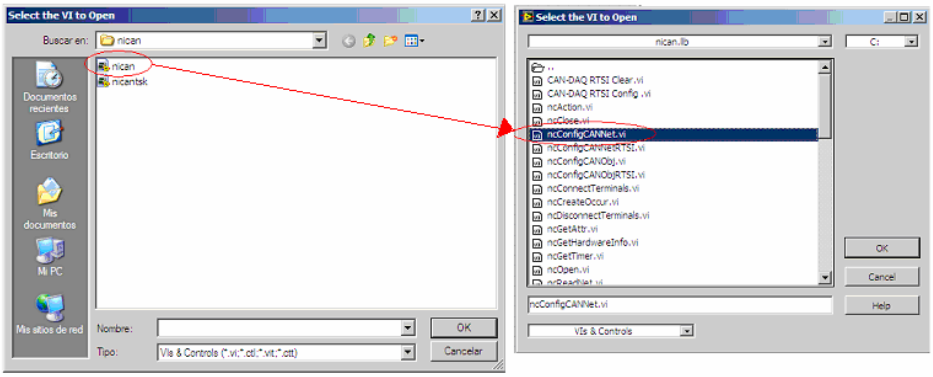
Seguidament mostrem d'una manera mes detallada com hem fet el programa de lectura i visualització de dades del bus CAN.

Primer obrim LabView i creem un projecte nou.



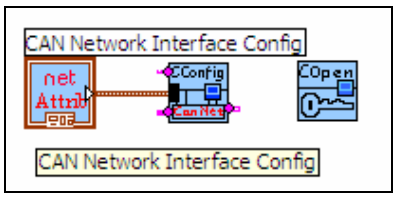
Il·lustració 89: Menú inicial del LabView

Ara procedirem a descarregar-nos dos arxius de la llibreria de nican, seran el "ncConficnicacn.vi" i el "ncOpen.vi", son necessaris per poder posar les característiques que tindran les dades del bus CAN, i pel port que aniran



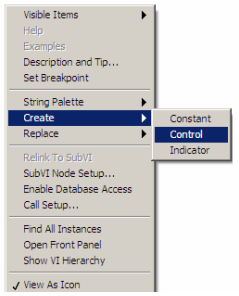
Il·lustració 90: Detall de la instal·lació d'una llibreria

Un cop seleccionats podem connectar.



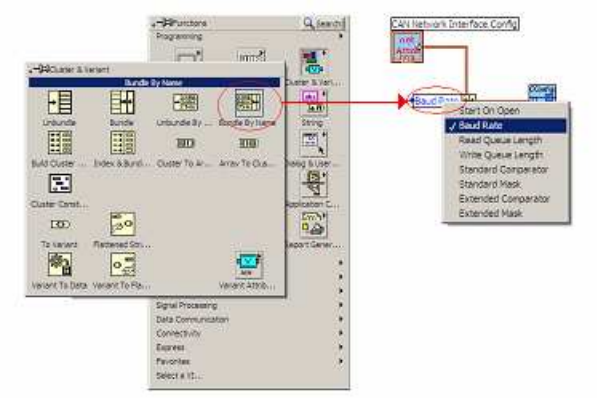
El "CAN Network Interface Config", s'aconsegueix de la següent manera:

Posant el ratolí sobre el símbol "Cconfig" i polsant amb el boto de la dreta a la connexió esmentada anteriorment, seleccionem control.



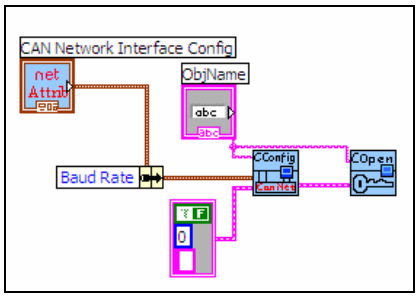
Ara per poder crear un control que ens permeti canviar el port del can del qual llegirem, hem de connectar entre si una estructura entre el "Cconfig" i el control "CAN Network Interface Config". Això ho farem de la seguen manera:

Seleccionem un bundle by Name, de la llibreria "Cluster & variables", després de connectar-la, fent doble click, seleccionem la opció Baud Rate.

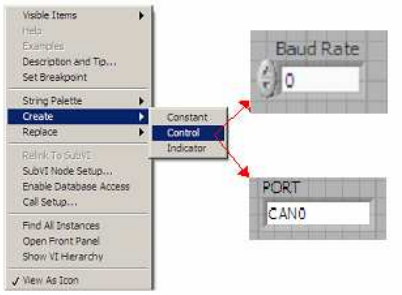


Il·lustració 91: Bundle by Name

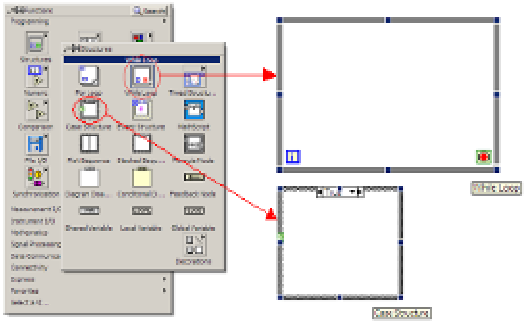
Ara procedim ha connectà el "Cconfig" amb el "Copen" i crear un control per configurar el baud Rrate.



Per aquest control, que es el que va unit al "Cconfig" i al "cOpen", podem escriure el valor en Kbs del Baut Rate. També creem un control per seleccionar el port.



Ara procedim a crear un estructura while, aquesta es un bucle continu, que ens permetrà llegir i visualitzar les dades indefinidament.

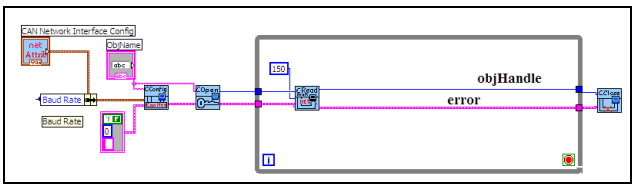


Il·lustració 92: Estructura while i case

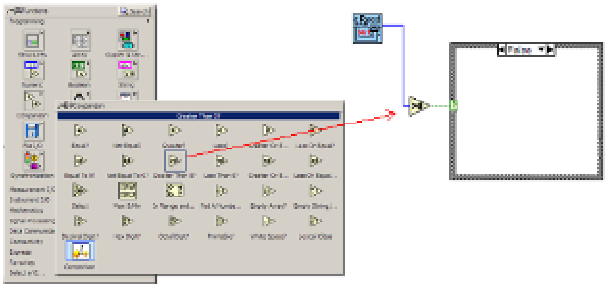
Ara posem dins del bucle un altre llibreria del NICAN 2.0, la que ens servirà per llegir les dades i poder-les tractar, “ncReadNetMult.vi”.

L’última llibreria que necessitem es la que tanca el circuit de comunicació CAN.

Despres d’obtenirla podem unir totes les llibreries mitjançant les connexions corresponents.

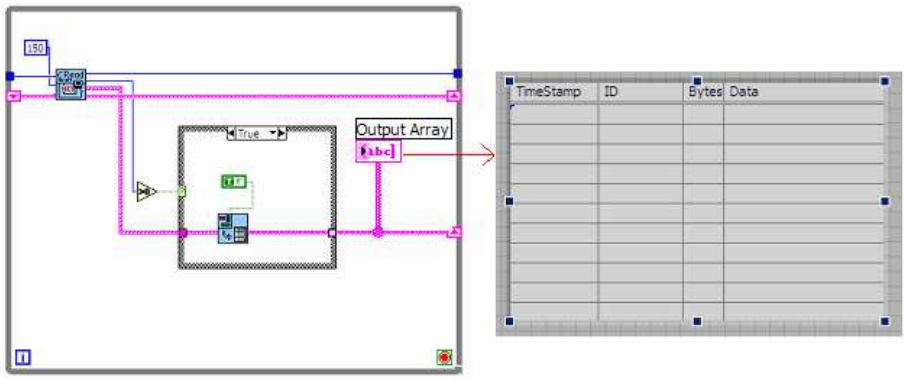


Ara necessitem visualitzar les dades, també utilitzarem un estructura case, ja que en cas que el bus no contingues dades, no necessitaríem visualitzares. L’estructura case s’ha de posar dins l’estructura while, utilitzarem aquesta estructura per condicionar la lectura de dades, de tal forma, que si les dades que rep son inferiors a zero, no s’activi.



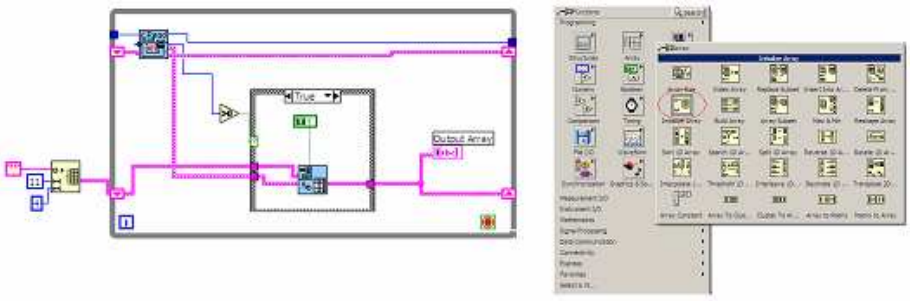
Il·lustració 93: Estructura case condicionada

Ara procedirem a fer el tractament i visualització de les dades que circulen pel bus, per aconseguir-ho necessitem un símbol de la llibreria nican, s'anomena "NetClusterArrayTo2DstringArray.vi". Un cop instal·lat procedim a fer les connexions corresponents.



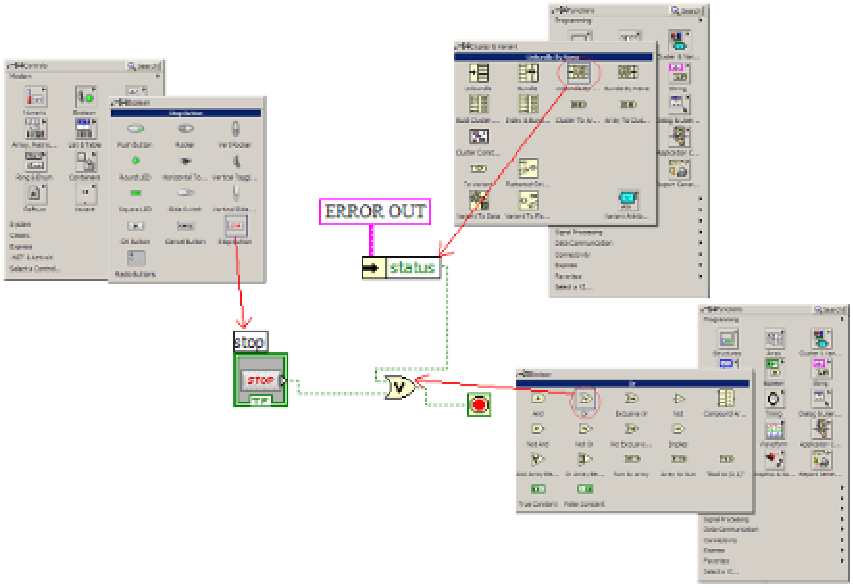
Il·lustració 94: Visualització de les dades del bus

El en el següent pas aconseguirem que les dades es vagin guardant, perquè si ho deixem així només es veuran les dades en temps real i no tindrem temps de llegir-les. Per fer-ho utilitzarem un Initialize Array, que connectarem al input array del "NetClusterArrayTo 2DstringArray.vi". Després de connectar i incloure les constants el circuit quedaria d'aquesta forma.



Il·lustració 95: Input Array

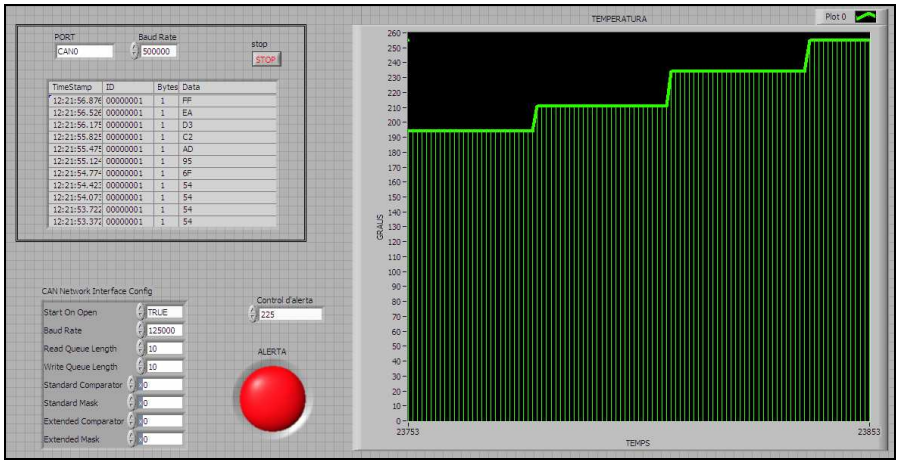
Per controlar el programa i també evitar que es saturi a causa d'algun error de la comunicació del bus CAN, controlarem el bucle mitjançant dos sistemes connectats amb un "or", un d'ells serà un boto de "stop" i l'altre, la línia d'error de la comunicació del bus CAN. El boto de stop, l'or els hem obtingut de la seguen forma:



Il·lustració 96: Disseny del control stop del bucle

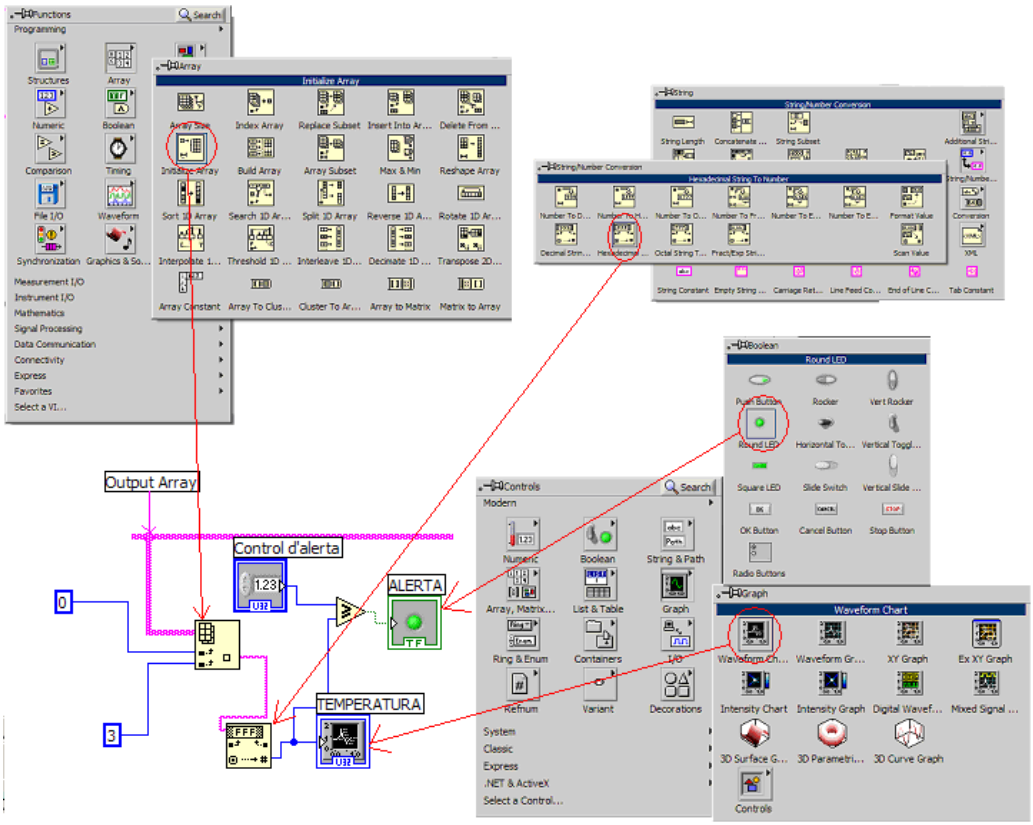
Per facilitar la comprensió de les dades que rebem, podem incloure en el panell frontal una gràfica, on d'una manera mes visual podem seguir l'evolució dels valors del sensor.

La següent figura mostra una forma de visualització de les dades del sensor.



Il·lustració 97: Panell frontal del projecte final

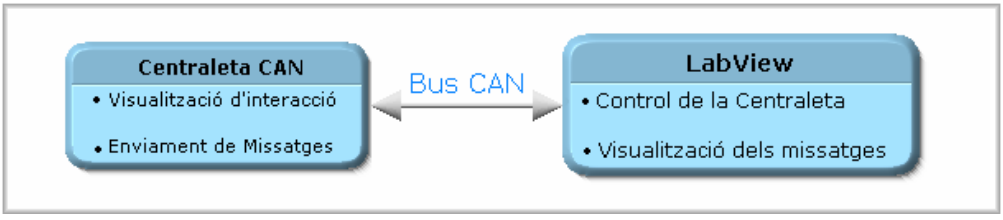
Per aconseguir visualitzar les dades amb una gràfica, hem de crear el següent circuit:



Il·lustració 98: Circuit de visualització amb gràfica

Podem observar que te un control de alerta, per si volem que el programa encengui un led si la dada que visualitzem sobrepassa un valor, que nosaltres podem ajustar mitjançant un control. Aquest es sols un exemple del tractament que es pot fer de les dades mitjançant el LabView, però existeixen altres formes de visualitzar i tractar les dades.

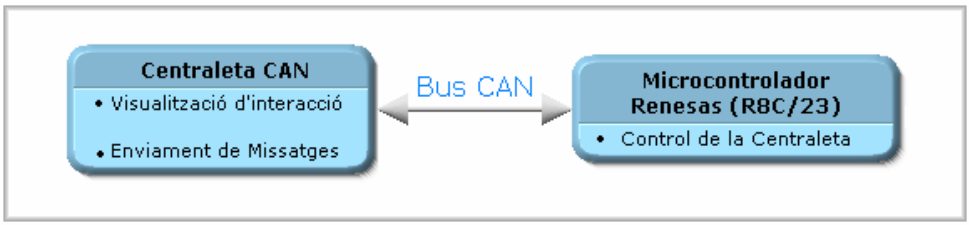
5.2.3.2. Centraleta CAN al LabView



Il·lustració 99: Organigrama Centraleta CAN

Per fer la comprovació del funcionament de la centraleta CAN, fem servir el LabView, amb un VI de lectura i escriptura de CAN. Una vegada estudiats quins son els missatges per actuar en la centraleta i quines característiques tenen (Baud rate, ID, Byts), podem prova a enviar dits missatges des de la placa Starter Kit R8C/23.

5.2.3.3. Control de la Centraleta CAN per el R8C/23



Il·lustració 100: Organigrama interacció Centraleta

L'objectiu d'aquesta pràctica es comprovar que els missatjes que creem amb el nostre codi compleixen el protol de CAN, per fer-ho hem utilitzat el projecte de fi de carrera titulat *monotorització de bus can amb compact_Rio* , que implementa una centraleta d'un choche seat ibiza.

El programa un cop executat permet activar mitjançant els switches de la placa els inetermitents de la centraleta i alguns dels leds de la placa renesas, la següent taula mostra la relació entre esl switches, els intermintens i els leds de la placa que controlem.

Switches de la placa	Centraleta can	Leds de la placa
SW1	Intermiten de l'esquerra	LED 1
SW2	Intermiten de la dreta	LED 2
SW3	Dos intermitens a la vegada	LED 3

Taula 21: Funcions dels polsadors

El codi del programa consta de diverses parts, les mes importants les comentem a continuació:

Segons les especificacions del projecte de la centraleta del automòbil seat ibiza, els missatges relacionats amb els intermitents tenen el identificador 470 i la mida de la dada es de 5 bytes, per tant nosaltres creem els missatges per comunicar-nos amb la centraleta de la següent manera:

```
missatge0.id=0x470;

missatge0.dlc=0x05;

missatge0.data[0]=0x01;

missatge0.data[1]=0x00;

missatge0.data[2]=0x00;

missatge0.data[3]=0x00;

missatge0.data[4]=0x00;
```

Com podem observar aquest es un dels tres missatges que enviem, aquest s'encarregarà d'activar el intermitent esquerra, per fer-ho tenim que enviar un 1, en la dada(0).

El següent pas un cop tenim les estructures dels missatges carregades amb les dades correctes, es enviar-les. Les enviem mitjançant els switches, per exemple si el sw1 esta polsat envia el missatge0 amb les dades corresponents al intermitent esquerra. En el seguen fragment de codi ho podem observar:

```
while(1)
    {
        if ((S1==0)&&(EnviarTrama==0))
        {
            while(S1==1){}
            pm=&missatge0;
            LED1=LED_ON;
            LED2=LED_OFF;
            LED3=LED_OFF;
            EnviarTrama=1;
        }
```

En el codi anterior podem veure que a part d'enviar el missatge, encenem el led corresponent de la placa, perquè visualment podem comprovar que hem entrar correctament al bucle on enviem el missatge del intermitent esquerra.

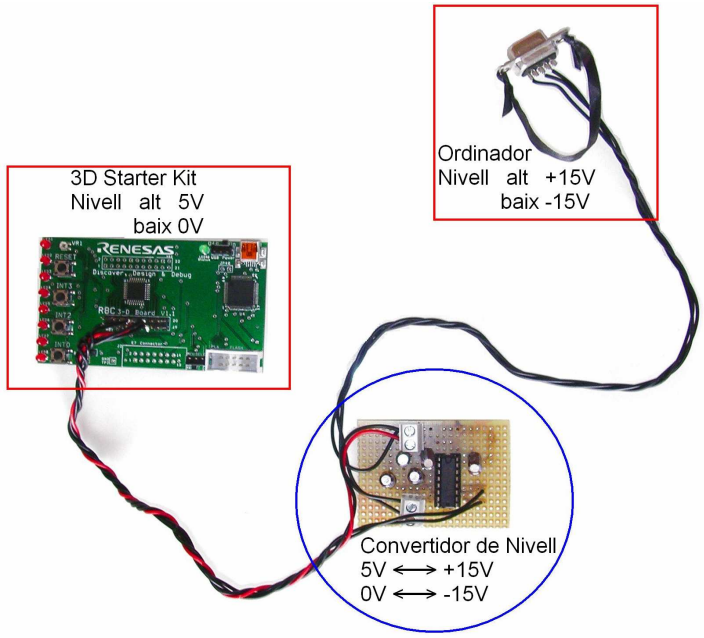
Amb aquest programa hem pogut comprovar, que els missatges que enviem compleixen el protocol CAN, i per tant ens podem relacionar amb altres aparells que estiguin connectats el bus CAN, mitjançant trames creades al nostre microchip.

Capítol 6:

DESENVOLUPAMENT DEL HARDWARE

6. Desenvolupament del Hardware

6.1. Placa convertidora de nivell



Il·lustració 101: Convertidor de nivell

Com s'especifica al punt 3 protocols de comunicació sèrie, els nivells de voltatge amb que treballen el microcontrolador i l'ordinador, son diferents.

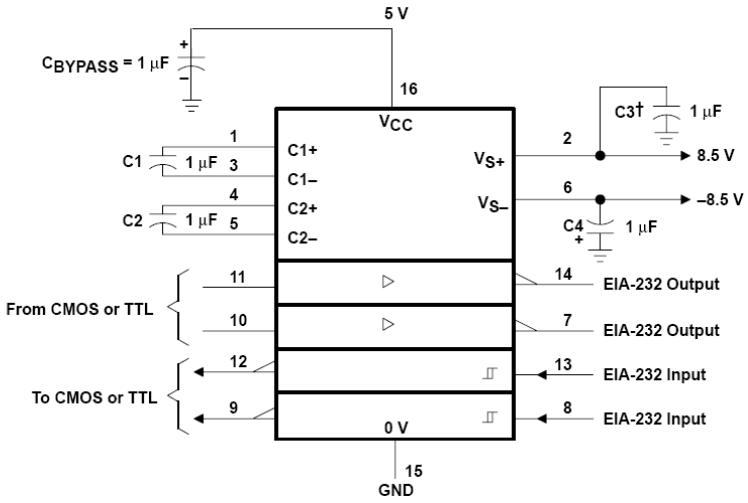
Mentre que per el microcontrolador el nivell alt "1" es aproximadament 5 volts i el nivell baix "0" es a 0 volts, per l'ordinador, el nivell alt es troba a 15 volts i el baix a -15 volts.

Per tal de que tots dos puguin treballar junts cal fer una conversió de voltatge, que es realitza amb el que en angles es diu un "transceiver".

La placa 3D Starter Kit no te incorporat aquest sistema de conversió, pel que tenim que muntar una placa externa que faci aquesta funció.

Ara l'ordinador podrà entendre perfectament les dades que li enviï el microcontrolador i aquest les rebudes de l'ordinador, això si, sempre que la configuració estigui ben fixada.

El chip utilitzat a estat el MAX232N 67FCT4K E4 en format DIP de 16 potes [12], condensadors electrolítics d'1uF



Il·lustració 102: Circuit típic del 232

Per mes informació consultar el Data sheet del MAX232 de Texas Instrument [12].

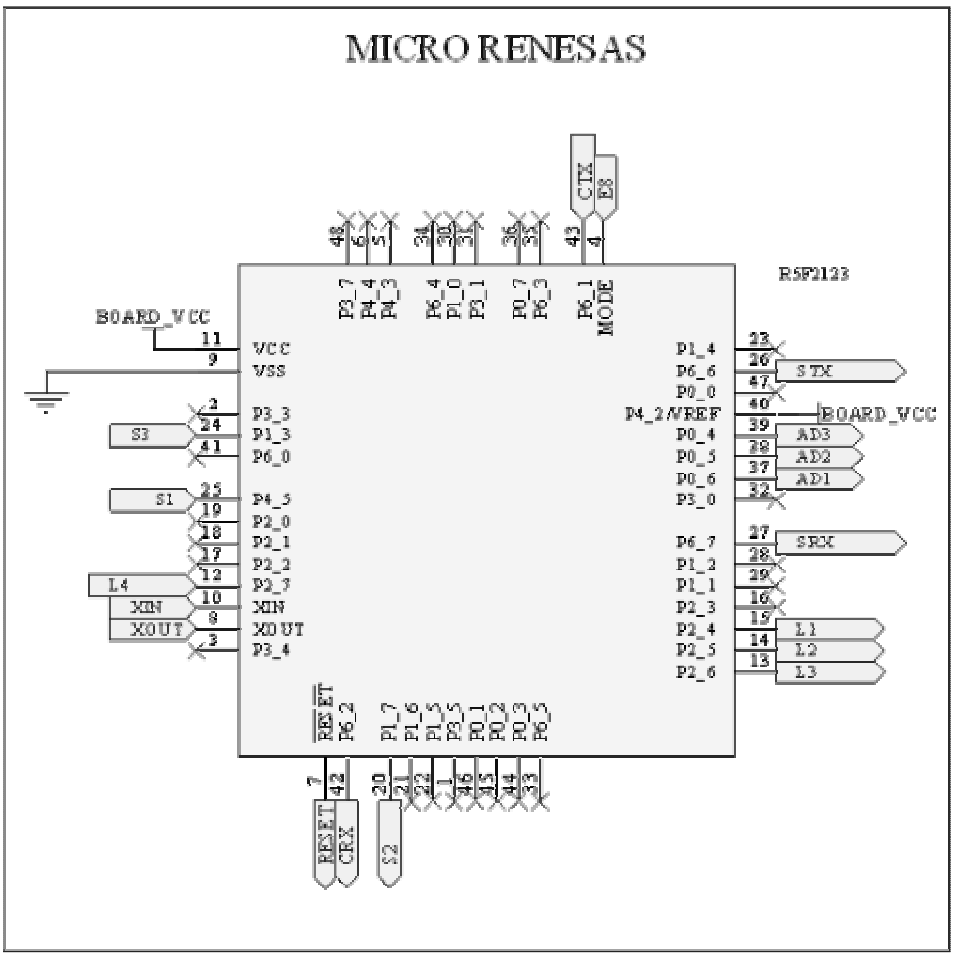
6.2. Placa dissenyada

La finalitat de la creació d'aquesta placa, es aplicar els coneixements adquirits en una única PBC que ens permeti els dos tipus de connexió.

Il·lustració 103: Organigrama comunicació CAN i SERIE amb la placa dissenyada

6.2.1. Esquemàtic

L'esquemàtic sencer es troba inclòs als annexes, aquí podem veure les seves parts i la funció que fan.

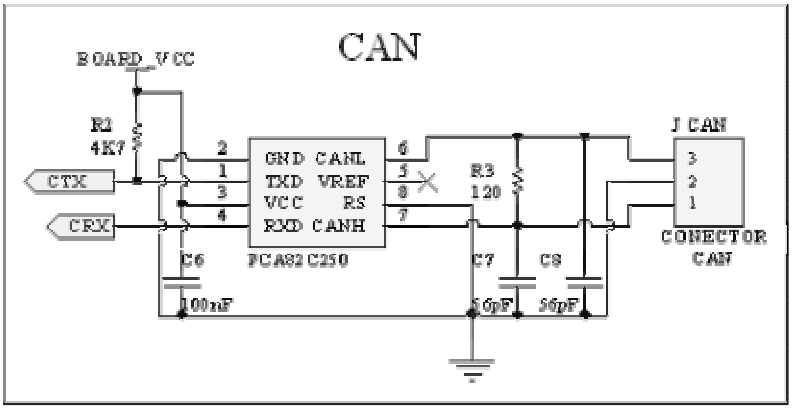


El cos del microprocessador [8] i les potes utilitzades.

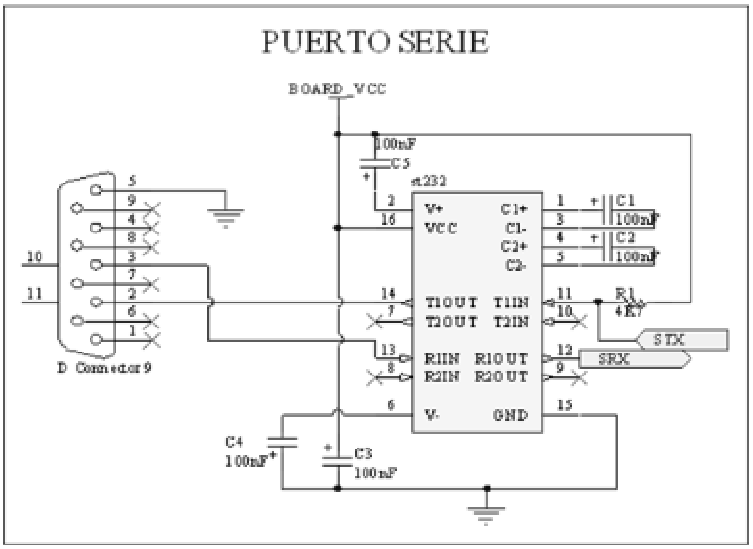
Pin	Nom	Connexió	Funció
4	MODE	E8	Programar el microcontrolador
7	RESET	RESET	Reset del circuit
8	XOUT	XOUT	Cristall extern
9	VSS	GND	Massa del circuit
10	XIN	XIN	Cristall extern
11	VCC	BOARD_VCC	Alimentació de la placa
12	P2_7	L4	Led 4
13	P2_6	L3	Led 3
14	P2_5	L2	Led 2
15	P2_4	L1	Led 1
20	P1_7	S2	Polsador 2
24	P1_3	S3	Polsador 3
25	P4_5	S1	Polsador 1
26	P6_6	STX	Transmissió sèrie
27	P6_7	SRX	Recepció sèrie
37	P0_6	AD1	Conversió A-D
38	P0_5	AD2	Conversió A-D
39	P0_4	AD3	Conversió A-D
40	P4_2/VREF	BOARD_VCC	Referència ca conversió A-D a BOARD_VCC
42	P6_2	CRX	Recepció CAN
43	P6_1	CTX	Transmissió CAN

Taula 22: Funció dels pins

El mòdul de CAN es basa en el Data Sheet del transceiver PCA82C250 [13].

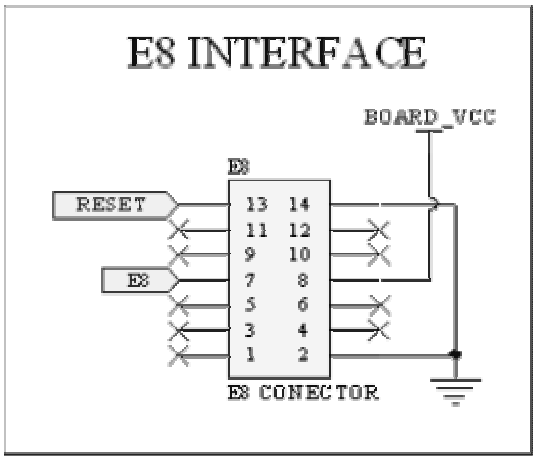


Del conector J CAN sortiran els canals L i H que enviarem a la PCMCIA CAN [11].

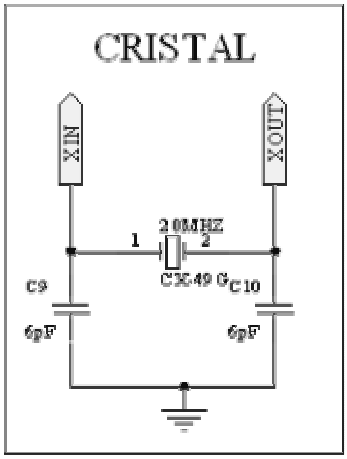


El circuit del port sèrie l'extraiem de les especificacions del Data Sheet del st232 [14].

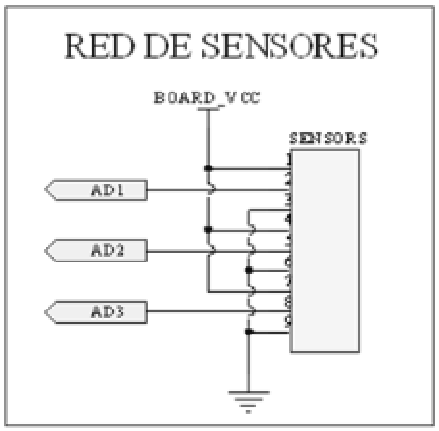
L'emulador del programa nomes fa servir les potes 2 i 14 per GND, la 8 per extreure l'alimentació del programador i alimentar la placa, la 7 per escriure el programa en memoria i la 13 per fer un reset per programa.



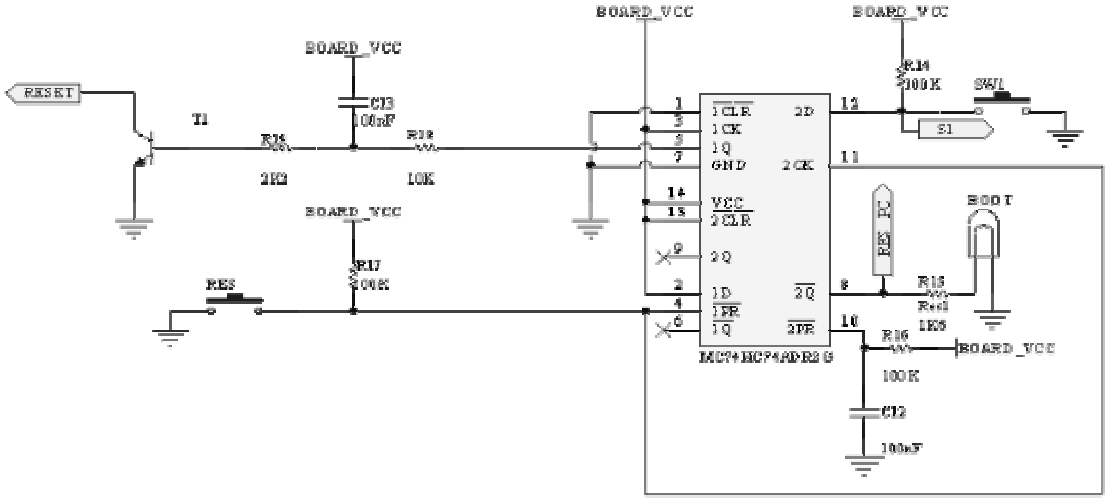
El cristall extern amb el que treballarem es de 20MHz:



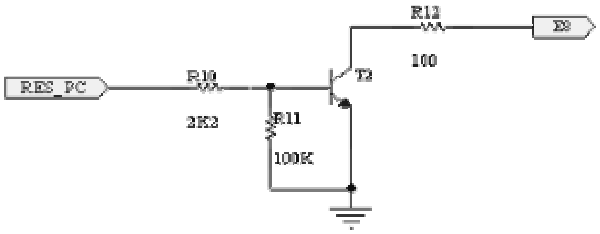
Per tal de fer proves habilitem tres entrades de conversió analògic/digital per connectar-hi 3 sensors.



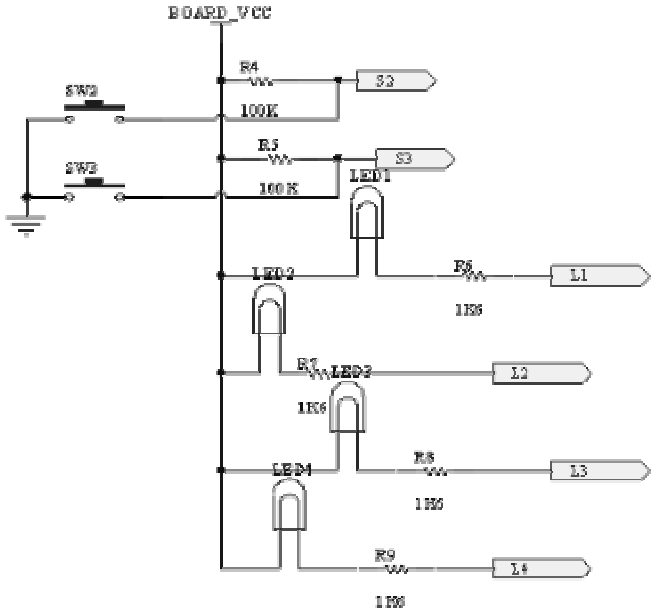
Per fer un circuit que treballi amb el reset de la placa i el del programador, fem servir el flip flop MC74HC74ADR2G [15].



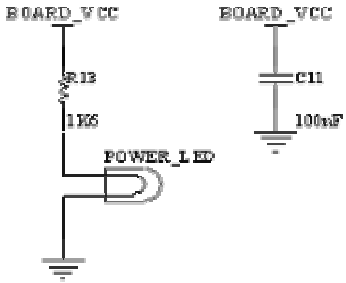
Un transistor ens connectarà la sortida del flip flop amb el microcontrolador.



Els Led's del circuit estan connectats a 5V, això farà que actuïn quan a Ln (n: de 1 fins a 4) estigui a zero lògic.



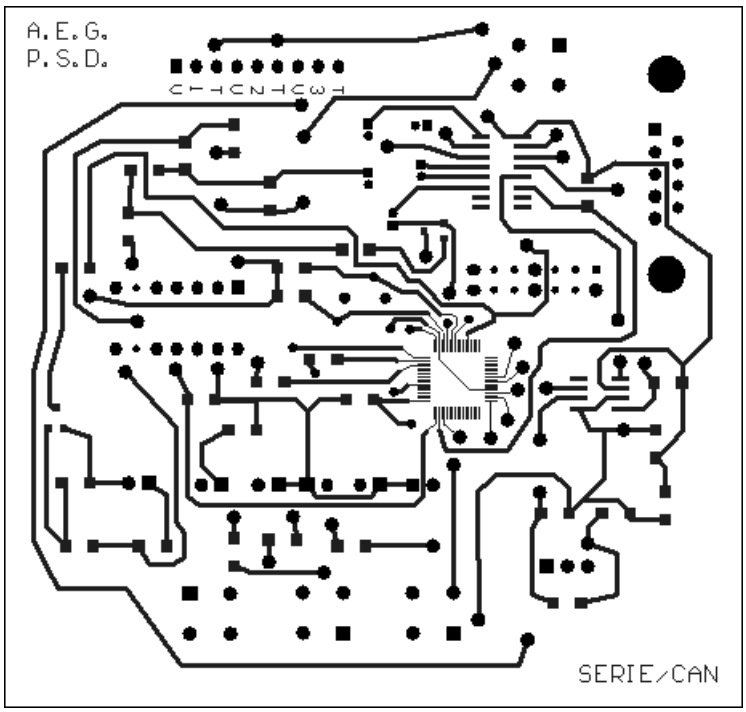
Per controlar que la placa està alimentada, afegim uln Led indicador.



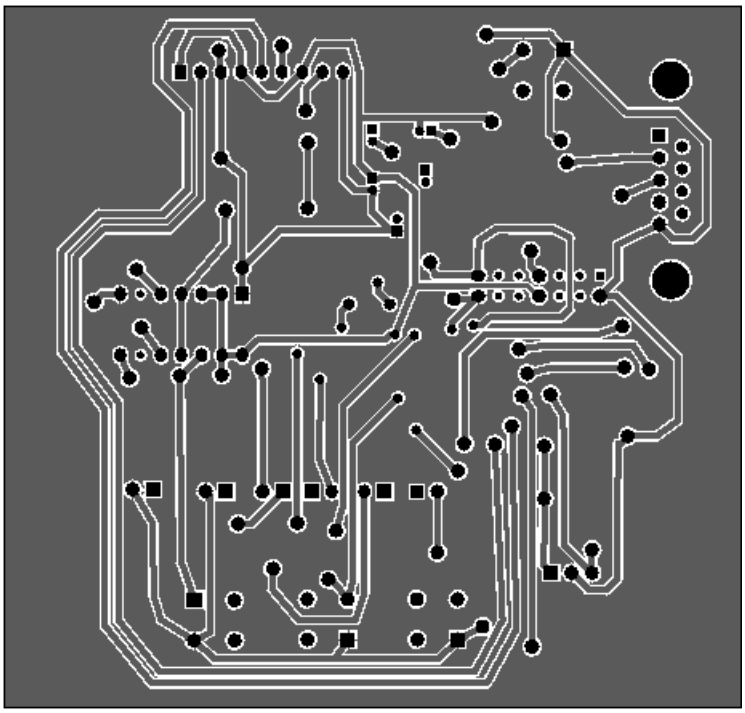
6.2.2. PCB

Per fer la placa hem utilitzat components SMD i DIP, per aquest motiu varem decidir fer la PCB de dues capes[4], [24].

La Top Layer o capa superior soldem els components SMD i les vies necessàries, mentre que a la Bootom Layer o capa inferior, soldem els altres i hi afegim un pla de massa.

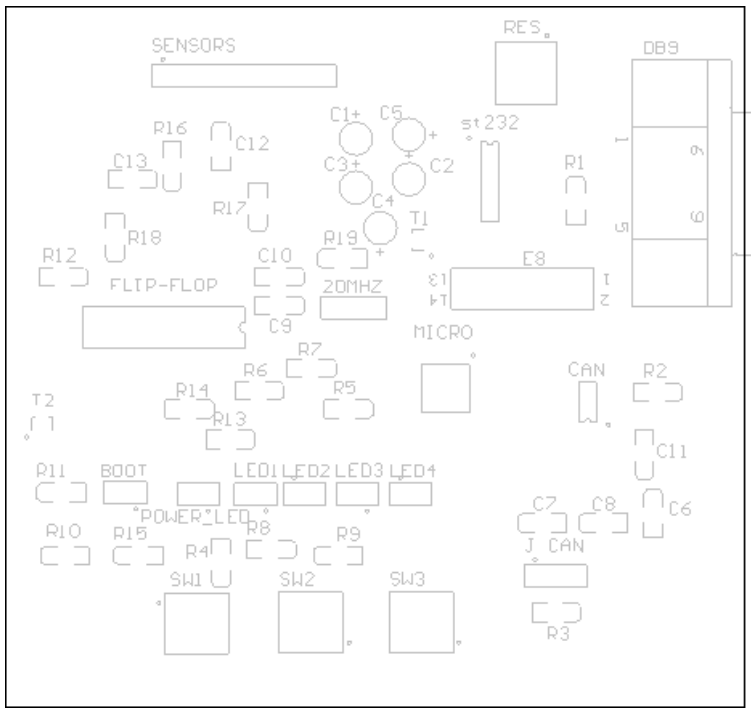


Il·lustració 104: Top



Il·lustració 105: Bottom

Seguint els esquemes i la taula de components tenim el circuit sencer amb la colocació correcte.



Il·lustració 106: Layout de Components

6.2.3. Llista de materials

Designator	Description	Footprint	Value
SENSORS	Header, 9-Pin	TIRA9PINES	
CAN	1ppm/°C, Low-Noise, +2.5V Voltage Reference	NSO8	
st232	+5V Powered RS-232 Driver/Receiver	SOIC16N	
MICRO	8-Bit Microcontroller	932-02	
FLIP-FLOP	330MHz, 4x1 Precision Video Multiplexer	DIP-14	
C10	Capacitor	DSO-F2/D6.1	6pF
C11	Capacitor	DSO-F2/D6.1	100nF
C12	Capacitor	DSO-F2/D6.1	100nF
C13	Capacitor	DSO-F2/D6.1	100nF
C6	Capacitor	DSO-F2/D6.1	100nF
C7	Capacitor	DSO-F2/D6.1	56pF
C8	Capacitor	DSO-F2/D6.1	56pF
C9	Capacitor	DSO-F2/D6.1	6pF
J CAN	Header, 3-Pin	pin3	
E8	Header, 7-Pin, Dual row	HDR2X7	
BOOT	Incandescent Bulb	PIN2	
LED1	Incandescent Bulb	PIN2	
LED2	Incandescent Bulb	PIN2	
LED3	Incandescent Bulb	PIN2	
LED4	Incandescent Bulb	PIN2	
POWER_LED	Incandescent Bulb	PIN2	
T1	NPN Bipolar Transistor	SO-G3/C2.5	
T2	NPN Bipolar Transistor	SO-G3/C2.5	
C1	Polarized Capacitor (Radial)	CAPPR1.5-4X5	100nF
C2	Polarized Capacitor (Radial)	CAPPR1.5-4X5	100nF
C3	Polarized Capacitor (Radial)	CAPPR1.5-4X5	100nF
C4	Polarized Capacitor (Radial)	CAPPR1.5-4X5	100nF
C5	Polarized Capacitor (Radial)	CAPPR1.5-4X5	100nF
DB9	Receptacle Assembly, 9 Position, Right Angle	DSUB1.385-2H9	
R1	Resistor	DSO-F2/D6.1	4K7
R10	Resistor	DSO-F2/D6.1	2K2
R11	Resistor	DSO-F2/D6.1	100K
R12	Resistor	DSO-F2/D6.1	100

R13	Resistor	DSO-F2/D6.1	1K6
R14	Resistor	DSO-F2/D6.1	100K
R15	Resistor	DSO-F2/D6.1	1K6
R16	Resistor	DSO-F2/D6.1	100K
R17	Resistor	DSO-F2/D6.1	100K
R18	Resistor	DSO-F2/D6.1	10K
R19	Resistor	DSO-F2/D6.1	2K2
R2	Resistor	DSO-F2/D6.1	4K7
R3	Resistor	DSO-F2/D6.1	120
R4	Resistor	DSO-F2/D6.1	100K
R5	Resistor	DSO-F2/D6.1	100K
R6	Resistor	DSO-F2/D6.1	1K6
R7	Resistor	DSO-F2/D6.1	1K6
R8	Resistor	DSO-F2/D6.1	1K6
R9	Resistor	DSO-F2/D6.1	1K6
20MHZ	Surface Mount Quartz Crystal	TIRA2PINES	
RES	Switch	swich	
SW1	Switch	swich	
SW2	Switch	swich	
SW3	Switch	swich	

Taula 23: Llistat de materials

Capítol 7:

RESULTATS

7. Resultats

Per fer el projecta hem anat assolint uns passos, per tal de complir els nostres objectius.

Del primer objectiu plantejat, treballar amb la placa 3D Starter Kit R8C/11:

- Hem après sobre la casa Renesas i la família R8C.
- Això ens ha servit per entrar al mon dels microcontroladors i la programació amb el llenguatge C.
- També hem tret de positiu l'aprendré a treballar amb els Data Sheets i saber consultar els registres per programar un microcontrolador.
- Treballar amb la comunicació sèrie satisfactòriament.
- Llegir dades de la comunicació fent servir l'HyperTerminal i el LabView.

El segon objectiu basat en el propòsit final del projecte ha estat treballar amb la placa Starter Kit R8C/23:

- Hem reafirmat els coneixements apresos sobre la casa Renesas i la família R8C, la programació en llenguatge C.
- Hem treballat amb la comunicació CAN satisfactòriament, tant per enviar com per rebre.
- Programa amb LabView per llegir i enviar dades.

El tercer objectiu, la Centraleta CAN.

- Comprovació de rebre i enviar dades des del LabView.
- Interactuar amb la centraleta enviant els missatges per la placa Starter Kit R8C/23

Una vegada complert els nostres objectius del projecte, el següent pas ha estat la unificació en una única placa creada a partir dels coneixements de les anteriors.

- Comprovació d'alimentació correcte.
- Led d'alimentació correcte.
- Connexió amb l'ordinador establerta pel connector E8.
- Circuit de reset correcte.
- Polsadors en funcionament.
- Cristall oscil·lador extern no funciona.
- Realitzades les probes anteriors comprovem que la placa no funciona correctament pel moment. Cal comprovar el funcionament de l'oscil·lador intern per tal de confirmar, si el problema ve del cristall extern o del microcontrolador.

Capítol 8:

CONCLUSIONS I PERSPECTIVES

8. Conclusions i Perspectives

El treball desenvolupat en el projecte ens ha permès treballar amb dos sistemes de comunicació diferents, comunicació SERIE i comunicació CAN. Per tant tenint en compte l'experiència adquirida, podem afirmar quin dels dos es d'ideal per desenvolupar la comunicació entre el microcontrolador i el PC.

Tenint en compte que el submarí ha de tenir un marge ampli per agafar les dades, el sistema de comunicació ideal seria el bus CAN, ja que aquest pot aconseguir distancies relativament grans (100m) a velocitats de transmissió altes (500Kbs), en canvi la comunicació sèrie sols pot assolir amb garanties distancies de 15m.

Una altre de les característiques en que es recolza la viabilitat del bus CAN respecte el bus SERIE, es el sistema de transmissió de dades. Aprofitant que CAN es un sistema enfocat als missatges podem enviar la informació de multitud de sensors al mateix temps, utilitzant les propietats del missatge. En la comunicació SERIE la informació dels sensors l'hem d'enviar una rere l'altre.

Per aquets motius i altres exposats en la memòria, concloem que el bus CAN es el sistema de comunicació apropiat per realitzar l'objectiu d'aquest projecte.

El projecte realitzat esta preparat per enviar les dades d'un sensor , però el microcontrolador te 12 canals per convertir dades d'analògic a digital, en conseqüència podríem crear una ret de 12 sensors i aprofitant les propietats de bus CAN, llegir-les en temps real al PC.

Amb la experiència obtinguda en programació LabView podríem crear un Instrument virtual on es podrien visualitzar totes les dades dels sensors en temps real mitjançant gràfiques. Aprofitant també per adaptar els dos tipos de comunicació en un mateix VI.

També seria interessant aprofitar millor les capacitats del microcontrolador, i mes concretament el sistema de comunicació LIN, ja que es mes barat que el CAN i ens podria ser d'utilitat per algunes aplicacions.

Fer funcionar completament el disseny creat de PCB.

Capítol 9:

ÍNDEX DE TAULES I FIGURES

9. Índex de taules i figures

9.1. Il·lustracions

9.2. Taules

Capítol 10:

BIBLIOGRAFIA

10. Bibliografia

Llibres

[1] Ellis, Margaret A., “C++ manual de referencia con anotaciones”, 1994

[2] Michell Waite, Stephen Prata, Donald, “Programación en C introducción y conceptos avanzados”, 1990

[3] Manuel Lázaro, Antonio, “LabVIEW programación gráfica para el control de instrumentación”, 1997

[4] Manuel Torres Portero, ”Diseño e ingeniería electrónica asistida por ordenadoren protel”, 1999

PDF’s

[5] REJ09B0062-0093Z, “Harware Manual R8C/11 Group”

[6] 3-DKUman_R8C, “R8C 3-D starter kit Instruction manual”

[7] um16c, “M16C IAR Embedded Workbench™ IDE User Guide”

[8] REJ09B0251-0100, “Harware Manual R8C/22 Group, R8C/23 Group”

[9] rej10j1245_rskr8c23_usermanual, “Renesas Starter Kit for R8C/23 User’s Manual”

[10] ehewum51, “High-performance Embedded Workshop User’s Manual”

[11] 370289k_CAN, “NI-CANTM Hardware and Software Manual”

[12] MAX232, “dual eia-232 drivers/receivers max232, max232i”

[13] PCA82C250, “CAN controller interface”

[14] st232c, “rs-232 drivers and receivers”

[15] 74HCT74PW, “Dual D-type flip-flop with set and reset; positive-edge trigger”

[16] protomat, “Manual de usuario de LPKF”

Pagines Web

[17] www.renesas.com

[18] www.ni.com

[19] http://www.semiconductors.bosch.de/en/20/can/index.asp

[20] http://www.can-cia.org

[21] http://www.can232.com

[22] http://www.canbus.galeon.com/electronica/canbus.htm

[23] http://www.iespana.es/mecanicavirtual/canbus.htm

[24] www.protel.com

[25] www.iar.com