



**TREBALL FI DE CARRERA
D'ENGINYERIA EN ELECTRÒNICA**

DEPARTAMENT ELECTRÒNICA

TÍTOL:

PROGRAMABILITAT DEL BUS CAN

AUTOR:

ORIOL RÚBIES

DIRIGIT PER:

DR. JOSEP SAMITIER

Barcelona, 29 de juny de 2001

BIBLIOTECA DE LA UNIVERSITAT DE BARCELONA



0700560385



4.3 HARDWARE

4.3.1 RACK

Tal com ja s’ha dit als objectius d’aquest projecte, ahora de realitzar l’entorn hardware de laboratori s’ha buscat altres finalitats de l’entorn. Així s’ha construït sis nodes CAN de forma que fossin reutilitzables per a altres aplicacions tot deixant connectors disponibles per entrades i sortides com es descriu en un altre apartat del projecte. Amb la mateixa filosofia de permetre l’ús dels dispositius per altres finalitats s’ha buscat un encapsulat el màxim d’estàndard. Això s’ha trobat gràcies als sistemes modulars segons la norma DIN 41494. Això permet de forma senzilla posar i treure plaques d’un bastidor, connectar i desconectar dispositius i en definitiva s’ha creat un entorn de proves flexible, modular, intercanviable i ampliable.

El sistema subrack triat és un bastidor de 19” que segueix les normes DIN 41494, part 5a i IEC 60297, part 3ª que permet muntar-hi circuits normalitzats, elements modulars, connectors DIN 41612 i cartes mare (backplanes). Té flexibilitat en quant a profunditat i en quant a fixació frontal i posterior en salts de 20mm per profunditat i 5.08mm per la frontal o posterior (5.08mm equival a 1E. I el seu acabat facilita la protecció d’emissions i interferències (EMI /RFI). Com el seu nom indica un subrack pot ser apilat en una estructura molt més gran, el rack. Aquests poden ser molt diversos, tipus armaris de subracks. Això dóna facilitat d’integració en altres sistemes estàndards. Els racks permeten apilar un gran nombre de subracks i aquests ahora tenir nombrosos dispositius. Però per al número de nodes dels quals partim, treballar amb armaris rack hagués estat un despropòsit, si més no és un exemple de ampliació i estandardització al que estat sotmès tot el que en aquest projecte s’ha creat, com s’ha volgut en els objectius.

La seva amplada frontal és superior a les necessitats del projecte; però la seva tria no ve donada només per les pròpies necessitats sinó per l’ampliació que tindrà i altres usos que se’n vulguin fer, en els moments de redactar aquest projecte en aquest sistema subrack s’hi troben els següents elements inclosos els propis. La feina de triar, buscar i muntar aquest sistema subrack ha estat feina d’aquest projecte.

Font Alimentació	Node Profibus	Node Profibus	Node Profibus	Nodes CAN	Nodes CAN	Nodes CAN	Tauler de Control	Lliures	Lliures	Lliures

Per cada targeta PCB tenim dos mòduls CAN de tal manera que cada PCB ha de tenir dos connectors masclesDB9, dos leds d'activitat de cada node i un led d'alimentació. S'ha posat cada targeta PCB en un panell porta. Cada panell porta cartes porta en el seu frontal els dos leds d'activitat, el led d'alimentació i els dos connectors DB9 mascle, corresponent a cada un dels dos nodes que té cada PCB. A més a més, cada panell portacartes té un tirador frontal per facilitar la seva col·locació i la seva extracció del subrack. La targeta s'ha fet segons les mides estàndards de les targetes Eurocard de 160mm x 100mm. Cada targeta va fixada al panell portacartes a través d'una escuadra portacartes del tipus integral.

La font d'alimentació s'ha muntat en mòdul especial que permet el muntatge de plaques voluminoses i pesades, o com el nostre cas peces no adaptables als portacartes. A més a més aquest mòdul facilita la seva connexió amb connectors normalitzats DIN 41612 a la reixeta de 2,54mm de subrack de forma fiable. Els acabats també són en alumini i porta un tirador frontal per facilitar l'extracció i la seva col·locació al subrack.

Material necessari per construir el sistema modular, tot el material és de la Serie 200 de Retex sobre sistema subrack, caixa instrument de 19".

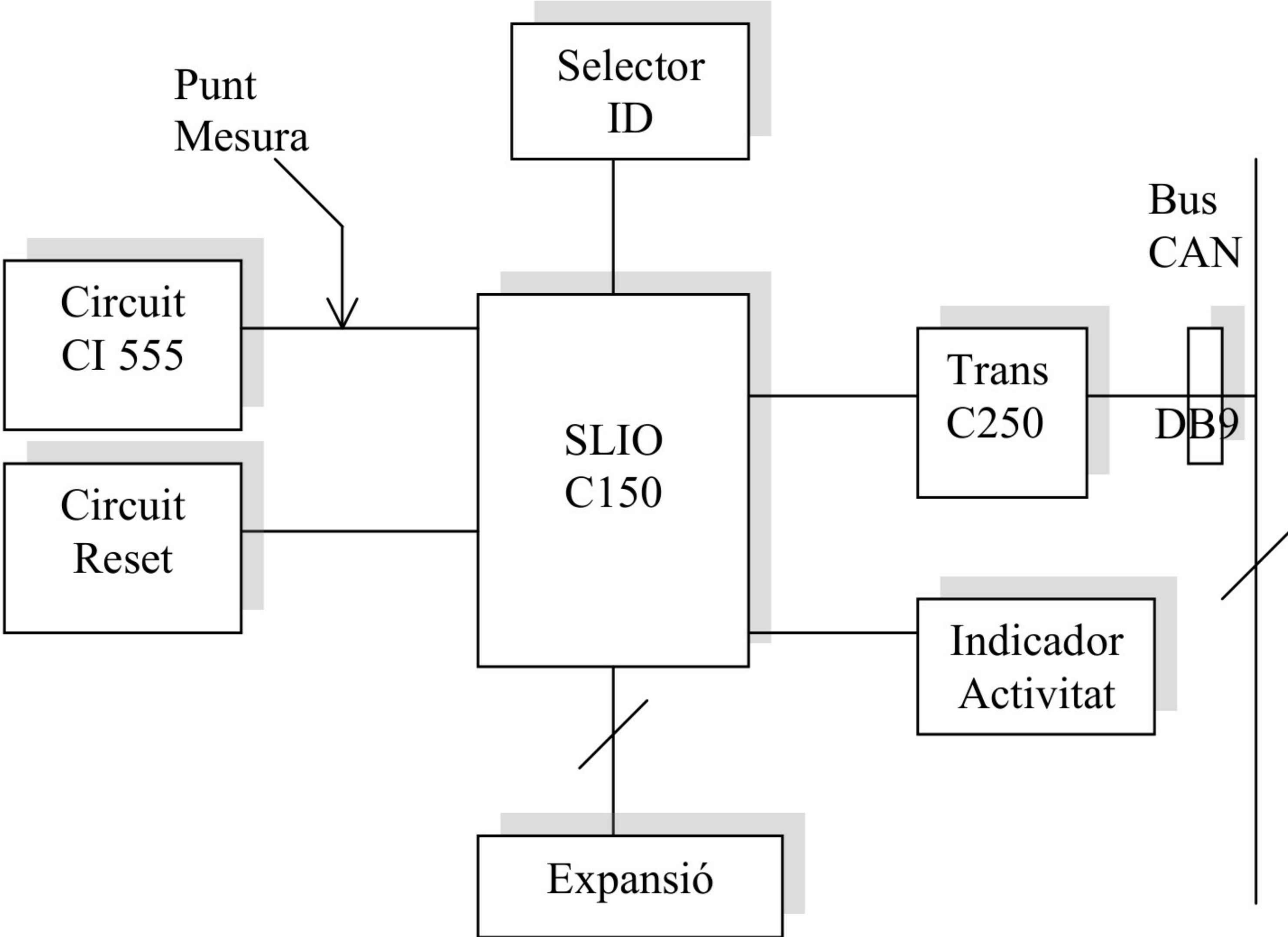
1	Subrack 84E 3U F160 IND Compost de: 2 perfils frontals. 2 perfils posteriors 2 laterals 3U/HE 2 esquadres de fixació rack 2 regletes roscades frontals Cargols de muntatge.	32201312
1	24 Guiacartes de 160 de PC injectat.	32203160
2	Tapa inferior / superior de planxa d'alumini de 1mm amb ventilació estàndard. (EMI/RFI)	32201716
1	2 Nanses d'alumini de 3U/HE	32203003
8	Panels portacartes de planxa d'alumini amb tirador en ABS de 3U, 6E. Compost de: 1 panel 1 escuadra portacartes 1 tirador Cargols de muntatge	32204306
2	Panels portacartes " de 3U, 4E	32204304
1	Panels portacartes " de 3U,	32204314

	14E	
1	Mòdul voluminós 3U, 14E	32205114
2	Perfils Z conversió DIN 41612	22203004

4.3.2 NODES CAN (AMB SLIO)

Per tal d’aconseguir nodes CAN senzill s’ha fet servir el SLIO. Cada node CAN disposa d’un d’aquests integrats més:

- un transceiver per al bus
- un circuit temporitzador
- un punt de mesura de la freqüència
- un circuit de reset
- una sortida d’indicació d’activitat
- una sortida d’expansió
- una connexió al bus
- un selector de l’identificador



Després s’han integrat dos nodes per targeta PCB de mida EuroCard, per tal de minimitzar l’espai a ocupar dintre del rack.

4.3.2.1 SLIO

L'integrat P82C150 de la Philips és el que hem descrit abans com a mòdul SLIO. Aquest és un dispositiu d'entrada i sortida amb un controlador que permet connectar-se a un bus amb protocol CAN. Permet configurar 16 pins com entrades i/o sortides, digitals i analògiques. Tot i que les especificacions del CAN parlen de velocitat de fins a 1Mbit/s el SLIO només treballa amb velocitats de 20Kbit/s a 125Kbit/s; tanmateix es capaç d'autocalibrar-se amb l'oscil·lador intern que incorpora. En general les tasques les realitza a través d'ordres d'un CANcontroller que rep a través del bus, tot i que pot enviar missatges de forma autònoma si es programa per captura esdeveniments, com canvis d'estat de pins d'entrada.

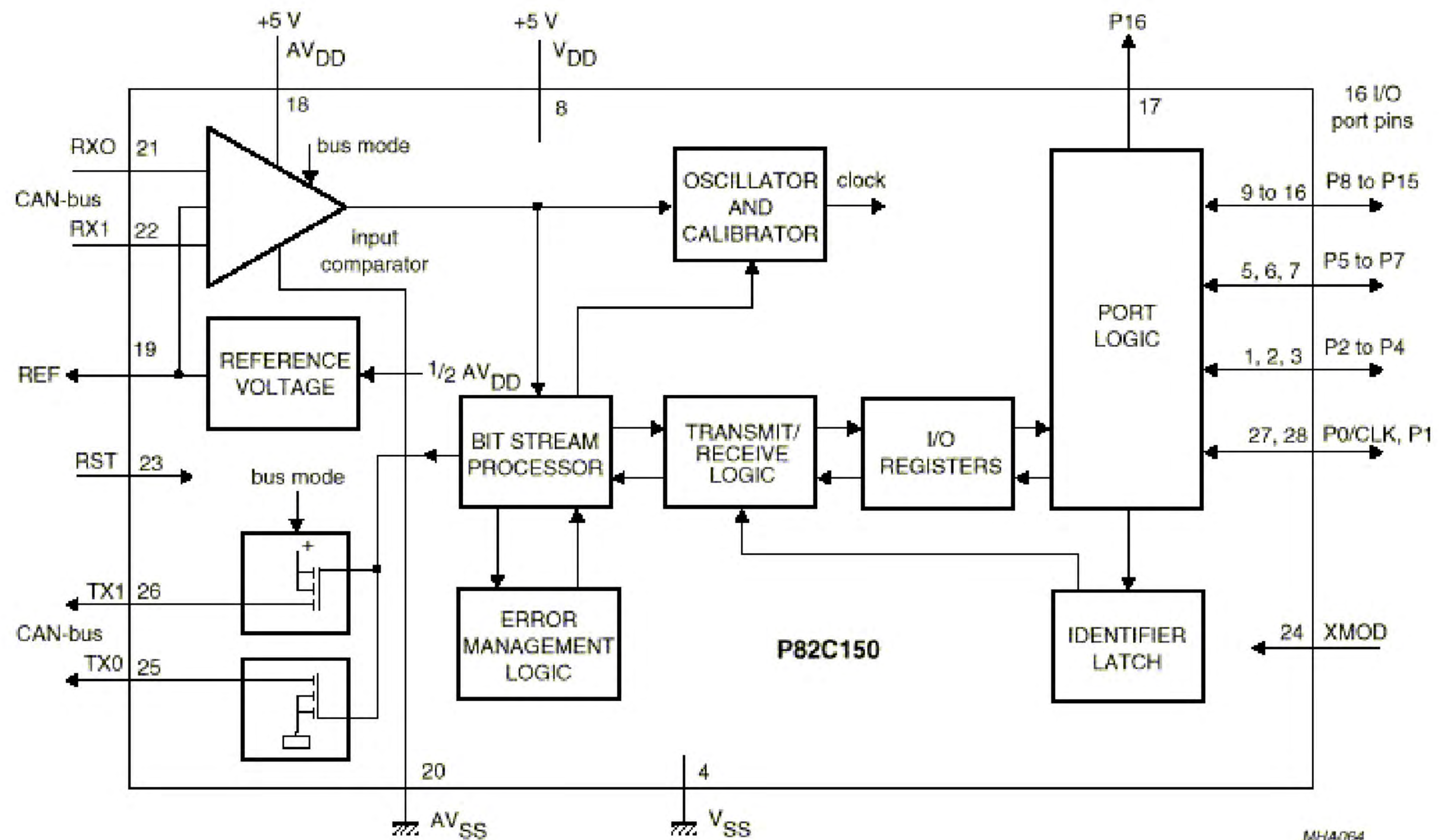
Aquest integrat és una bona solució de baix cost per incrementar la capacitat d'entrades i sortides d'un microcontrolador, basat en un bus CAN que dóna una gran fiabilitat, una baixa complexitat de cablejat i una taxa de bits bona. És per aquest motiu que és utilitzat al món de l'automòbil i la indústria com a sensor/actuador.

Característiques principals:

- En tan sols un dispositiu, entrades/sortides i controlador de protocol CAN.
- Treballa amb les especificacions CAN 2.0A, CAN 2.0B en mode passiu, però amb limitació de velocitat a 125Kbit/s.
- Incorpora un oscil·lador intern, també permet treballar amb un rellotge extern.
- 16 pins configurables com entrades i sortides digitals i analògiques.
- Tots els pins són configurables individualment i a través del bus CAN.
- Fins a 16 pins configurats com a captadors de canvis i que autoenvia els canvis pels bus.
- Fins a 16 pins amb sortida 3-state.
- Dos sortides quasi analògiques amb una precisió de 10 bits.
- Sis canals multiplexats de conversió analògica-digital amb precisió de 10 bits.
- Dos comparadors de propòsit general.
- Velocitats de 20-125Kbit/s
- Autodetecció i autocalibració de velocitat de bit.
- Fins a setze nodes SLIO en un sol bus.
- Quatre bits de l'identificador programables.
- Té sleep-mode desactivable via bus CAN.

Aquest dispositiu ve encapsulat amb un SO28 (plastic small outline package), el qual ens ha obligat a realitzar una petita placa PCB només per poder integrar-lo amb un connector DIP de 28 pins, més fàcil de treure i reutilitzar en cas d'errors en el prototipus; molt important degut al fet que aquest dispositiu ja no es troba amb facilitat al mercat.

A la següent figura podem veure com és el diagrama de blocs del SLIO:



Sense entrar en detalls sobre totes les característiques i maneres de funcionar del SLIO, s'explica com s'ha configurat i com s'ha fet funcionar com a node CAN de l'entorn de laboratori. Deixant en el CD que acompanya aquest projecte les fulles d'especificacions. El funcionament del SLIO és molt senzill només requereix de quatre missatges de calibració consecutius enviats al bus per a que es pugui autoiniciar i calibrar a la velocitat del bus. Si el node SLIO està ben connectat al bus, funciona bé i rep aquest quatre missatges, el primer que fa és enviar un missatge al bus on es dona a conèixer i diu que està ben calibrat, aquest missatge envia 3 bytes amb informació de l'estat actual del node amb un missatge especial de sign-on. Un cop el node intel·ligent del bus rep aquest missatge del SLIO ja pot enviar el missatge o missatges de configuració a l'integrat. Els missatges per configurar el SLIO han de tenir el identificador que s'ha assignat a través dels Pins P0, P1, P2 i P3. El missatge de sign-on retornat per l'SLIO té el mateix identificador més el bit de trama de direcció de resposta (típic en els SLIO per diferenciar missatges cap al bus o missatges del bus cap al SLIO).

La manera de configurar el bus és enviar un missatge al bus amb el id del SLIO que es vol configurar amb 3 bytes en el camp de dades. El primer byte és interpretat per el SLIO com una adreça de la memòria i els altres dos bytes són interpretats com els valors que s'han d'escriure en aquesta adreça de memòria del SLIO. Aquesta és la part més importat del SLIO, que cal conèixer bé per poder-lo configurar.

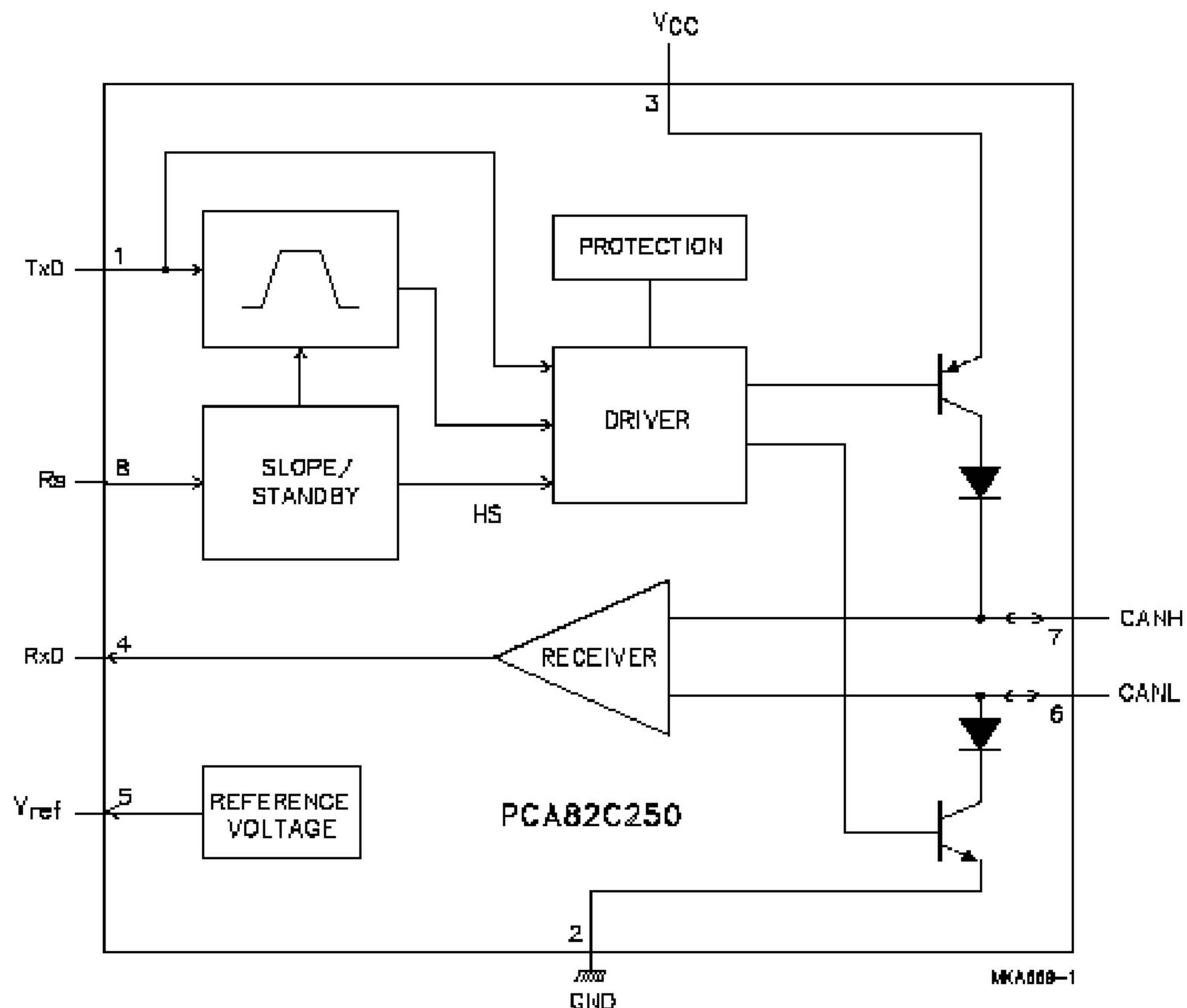
En la taula, a la següent pàgina, es pot veure les adreces de memòria del SLIO i per què serveixen cada una:

ADREÇA	FUNCIÓ (2BYTES)
0 ENTRADA DADES	Demandar al SLIO que indiqui els valors d'entrada de cada un dels pins (P0-P15) i enviar el seu valor en un missatge al bus.
1 POSITIVE EDGE	Indicar quins pins poden generar una interrupció al SLIO per a que aquest enviï al bus un missatge amb els valors dels pins. La interrupció ha de ser deguda a un canvi de senyal de negatiu a positiu en un d'aquest pins.
2 NEGATIVE EDGE	Indicar quins pins poden generar una interrupció al SLIO per a que aquest enviï al bus un missatge amb els valors dels pins. La interrupció ha de ser deguda a un canvi de senyal de positiu a negatiu en un d'aquest pins.
3 SORTIDA DADES	S'indica quins valors posar en cada un dels pins de sortida del SLIO.
4 HABILITAR SORTIDA	Només es posar el valor de sortida, indicat a l'adreça anterior, als pins que aquí s'indiquin prèviament com habilitats de sortida.
5 CONFIGURACIÓ ANALÒGICA	Especifiquem dels pins P0 a P15 quins funcionaran en mode analògic, tan DPM com ADC.
6 MODULACIÓ DPM1	Configuració per genera en els pins específics de DPM1 una modulació distribuïda amb polsos. Amb ajuda capacitats exteriors permet crear fàcilment un senyal analògic.
7 MODULACIÓ DPM2	Configuració per genera en els pins específics de DPM2 una modulació distribuïda amb polsos. Amb ajuda capacitats exteriors permet crear fàcilment un senyal analògic.
8 CONVERSOS ANALÒGIC A DIGITAL	Configuració del pins concrets per a realitzar ,mitjançant comparador analògic, una conversió analògica a digital.

Com es pot veure a l'annex, en el programa CAPL, per a la finalitat del projecte només s'han fet servir les adreces de zero a quatre. L'adreça zero per indicar quin pin serà l'entrada del circuit temporitzador. L'adreça u i dos per indicar que aquest pin generi una interrupció que enviï un missatge al bus, és la base per crear missatges periòdics de cada SLIO al bus. I finalment l'adreça tres i quatre per indicar activat al node, habilitant el pin que alimenta el LED i enviant missatges per canviar el nivell d'aquest pin.

4.3.2.2 TRANSCEIVER

A les pràctiques s'ha utilitzat l'integrat PCA82C250 de Philips per a realitzar aquesta funció. L'ús de transceivers serveix per donar capacitat de transmissió diferencial al bus per part d'un controlador CAN; així com a recepció des del bus. En concret el transceiver utilitzat és plenament compatible amb l'estàndard CAN, poden treballar a velocitat de fins a 1Mbit/s. Proporciona millores d'aïllament tèrmic i per reduir interferències del tipus RFI (Radio Frequency Interference) i EMI (Electromagnetic Interference). Suportant grans tensions en mode comú. Permet connectar fins a 110 nodes a un sol bus. A la figura següent podem veure el diagrama de blocs d'aquest integrat.



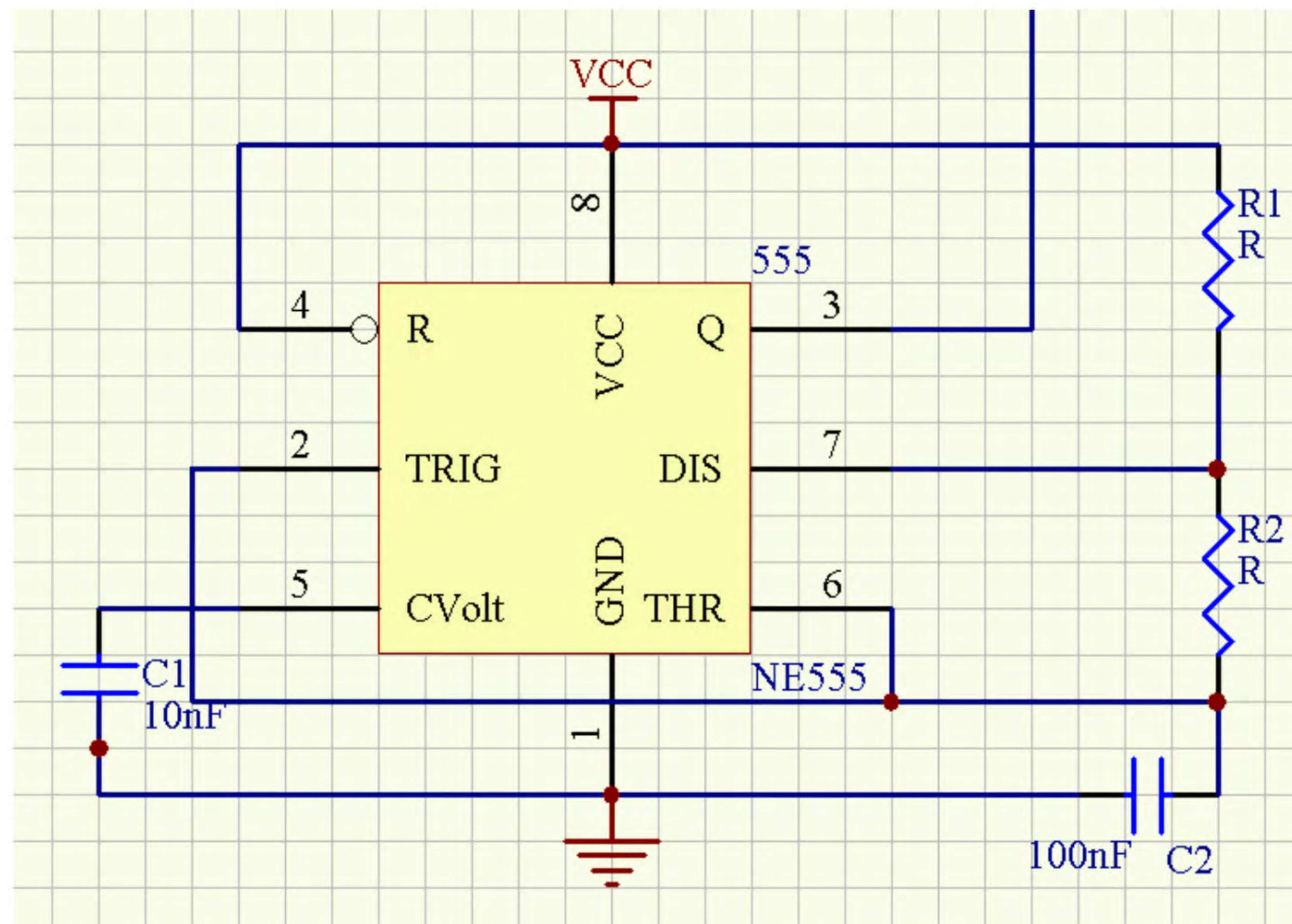
4.3.2.3 CIRCUIT TEMPORITZADOR

S'ha fet servir un senzill circuit basat en l'integrat 555 per tal de donar a cada node CAN una base de temps pròpia. Aquesta base de temps és la encarregada de generar les interrupcions que fan que el node SLIO envii un missatge al bus. Així aquest circuit ha de poder ser flexible alhora de canviar la freqüència i també a de ser senzill perquè s'ha d'implementar en molts nodes. L'opció triada és aquest 555 amb una configuració on la freqüència depèn de la relació de dues resistències integrades en un sol potenciòmetre

(R1 i R2). El valor de la freqüència que ens donarà la sortida (Q pin 3) del 555 serà segons la següent formula:

$$f = \frac{1.44}{(R_1 + 2R_2)C_2}$$

A la figura inferior es pot veure el circuit fet amb el 555 com a temporitzador per a cada node.



4.4 CANALYZER

El CANalyzer ha estat una eina de gran utilitat per a l'elaboració del projecte. En aquest capítol es descriu aquest entorn de desenvolupament i com s'ha configurat de forma concreta per a la finalitat d'aquest projecte.

4.4.1 DESCRIPCIÓ



CANalyzer és una eina estàndard de desenvolupament de sistemes basats en el bus de camp CAN. Amb aquesta eina podem observar, analitzar, tractar, simular i generar senyals, missatges CAN, sobre un bus.

S'ha treballat amb una targeta PC-ISA i el software que s'anomena CANalyzer, en la seva versió 2.2. En el mateix paquet que es troba aquest software i aquest hardware utilitzat trobem el CANoe. Però que en aquest projecte no s'ha fet servir. El CANopen és un perfil de comunicació, basat en la capa d'aplicació del model OSI i en llenguatge CAPL.

El CANalyzer per si mateix només és una eina d'anàlisi passiva de bussos CAN. Pot enviar missatges només sota demanda. Es poden generar missatges amb la creació de blocs que generen missatges, blocs de programes CAPL, blocs de resposta a certs missatges o amb blocs que generen missatges a través d'interactuació de l'usuari per interfície.

La targeta PC utilitzada és una CAN-AC2 ISA-card. I en conjunt requereix d'un PC tipus Intel o compatible, processador Pentium o superior, de almenys 32MBytes de RAM i equipat amb sistema operatiu Windows de 32 bits.

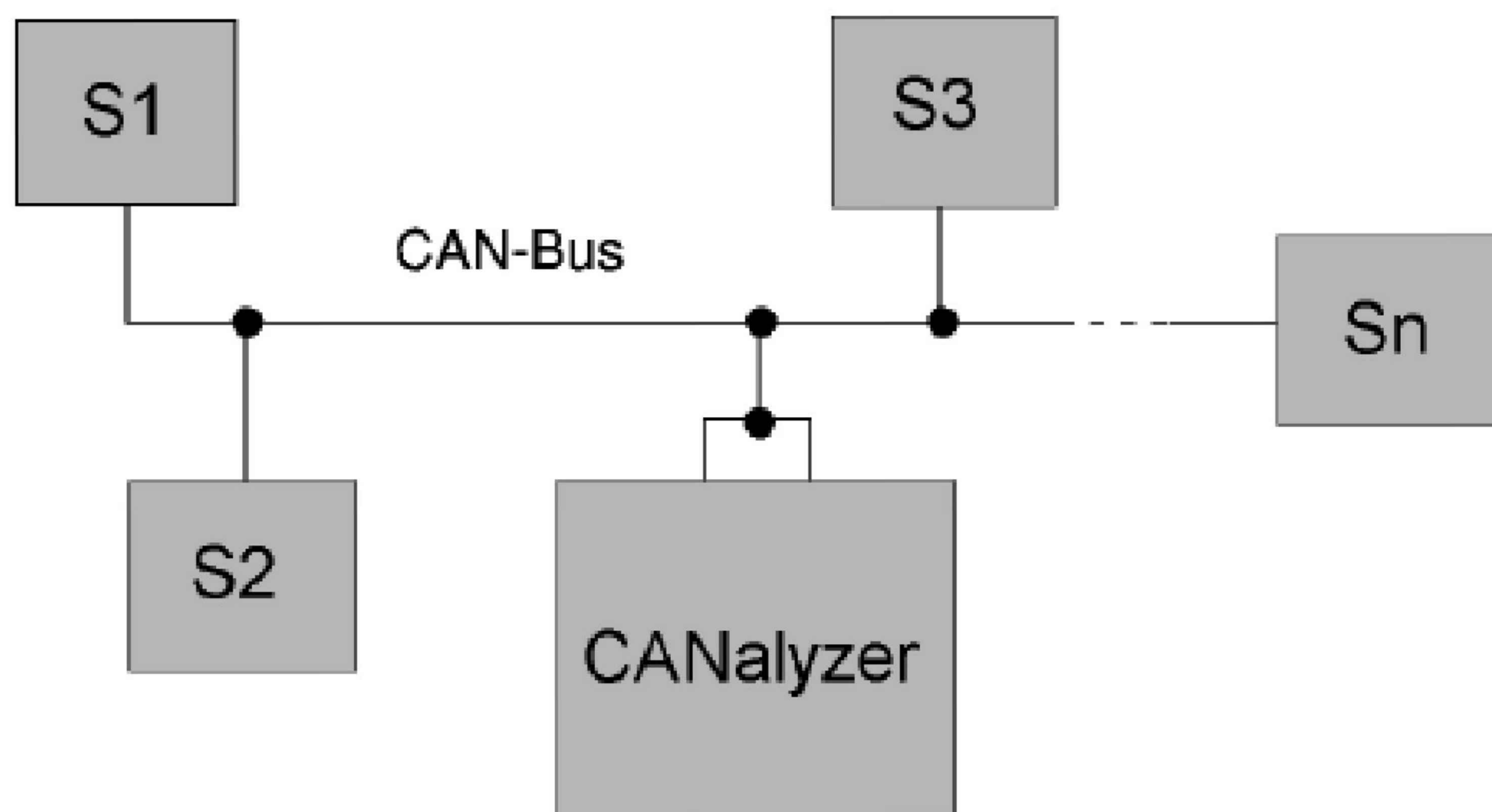
Amb el CANalyzer es pot treballar tan a nivell de control total de totes de dades, com es necessita en aquest projecte, o a nivell d'una representació més lògica de les dades. Això últim es pot aconseguir gràcies a la base de dades adjunta, CANdb, que té el CANalyzer. La programació és senzilla amb el CAPL, però si es vol és pot fer servir els blocs de generació de missatges que encara són més senzills i no requereixen cap coneixement de programació. A més a més, tots els aspectes bàsics també es poden fer amb certa facilitat sense necessitat de programar, com ara: Fer un traçat de tots els missatges, mostra segments concrets de dades de missatges específics, enviar o respondre missatges predefinits, avaluació estadística - càrrega i distribució del bus- o gravar tots els missatges per fer una avaluació en mode offline.

De totes maneres, si es vol estendre la funcionalitat del CANalyzer s'ha de fer servir el blocs de programa, que com els altres blocs, poden ser inserits en qualsevol punt del diagrama de flux. Aquest blocs estan programats en CAPL (CAN Access Programming Language. Amb la singularitat que tenen una programació orientada a esdeveniments, que permet als desenvolupadors trobar solucions a les seves necessitats d'una forma extremadament ràpida. El CANalyzer incorpora una utilitat, entorn de programació, per facilitar la tasca de generar aquest blocs en llenguatge CAPL.

En general, el CANalyzer pot servir al desenvolupador per treballar en una sèrie de configuracion, sistemes:

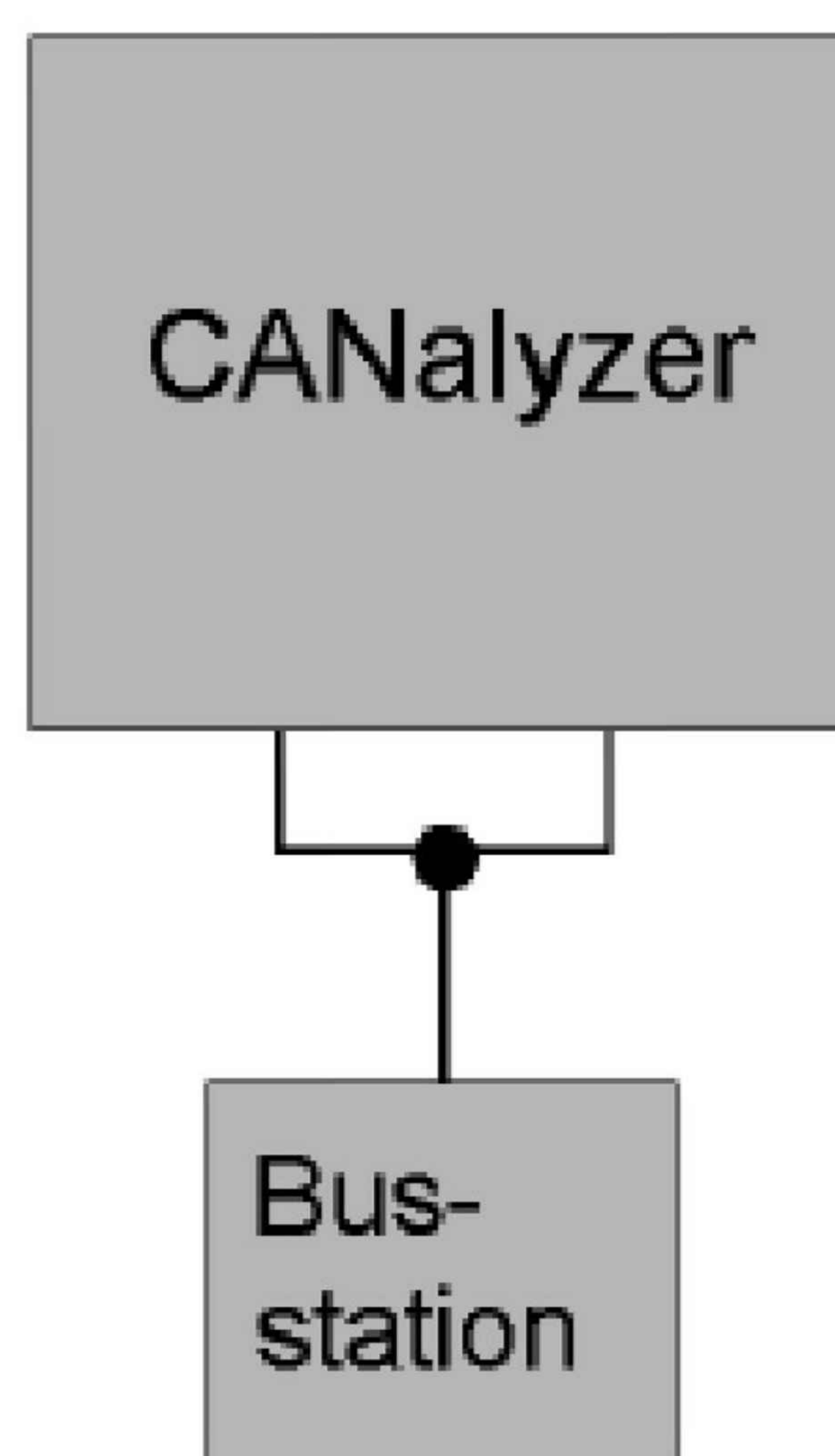
Emular un node CAN del bus:

CANalyzer pot emular tots aquells aspectes importants d'un node que són rellevants de cara al bus, com ara respondre a certs missatges o esdeveniments.



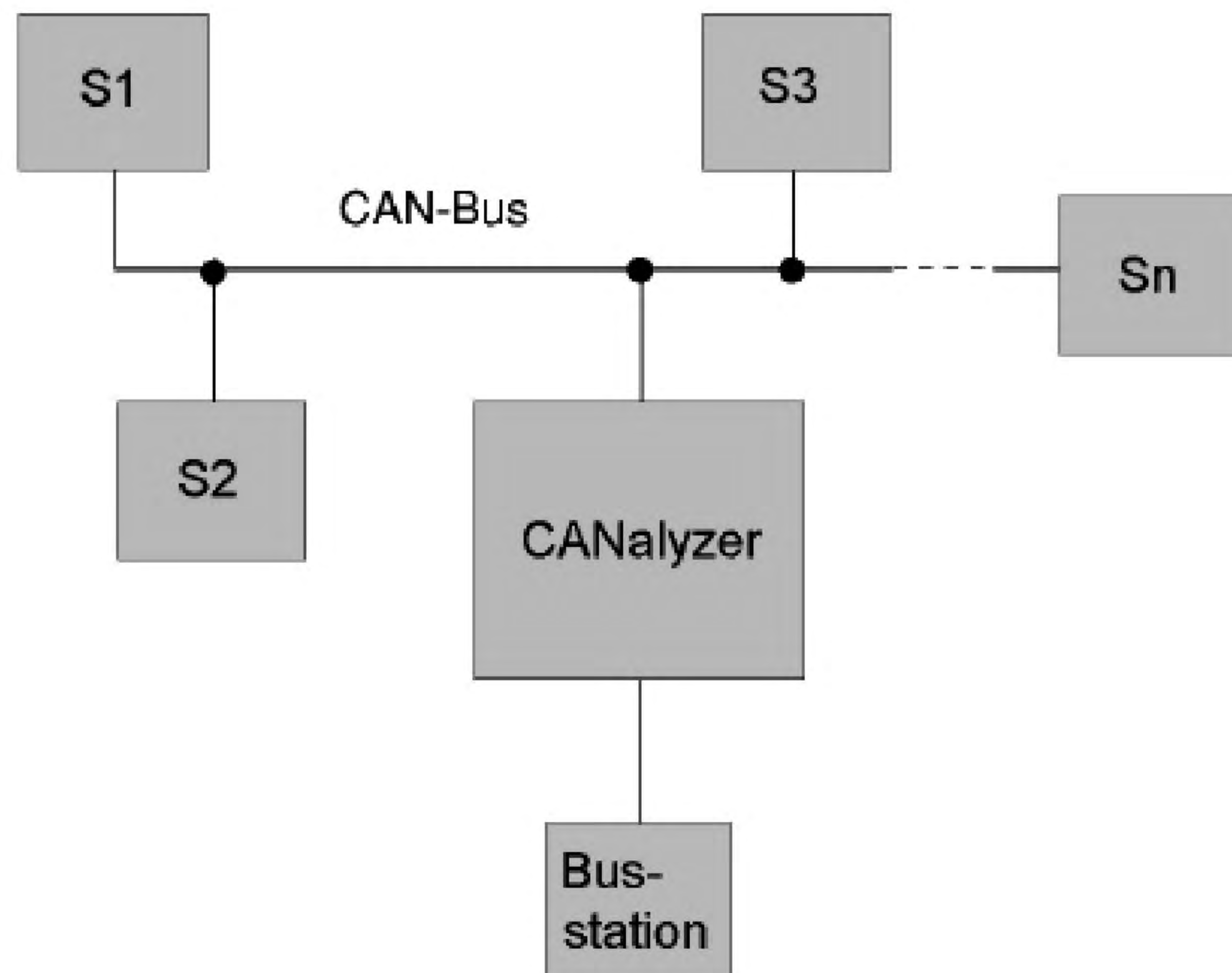
Emular un sistema sencer per testejar un node:

En tasques de desenvolupament podem utilitzar el CAN per simular tot un sistema i connectar-hi només un node només per poder provar-lo sense tenir tot el sistema encara construït. Tot el tràfic de tots els nodes que tindria el bus poden ser emulats pel CANalyzer.

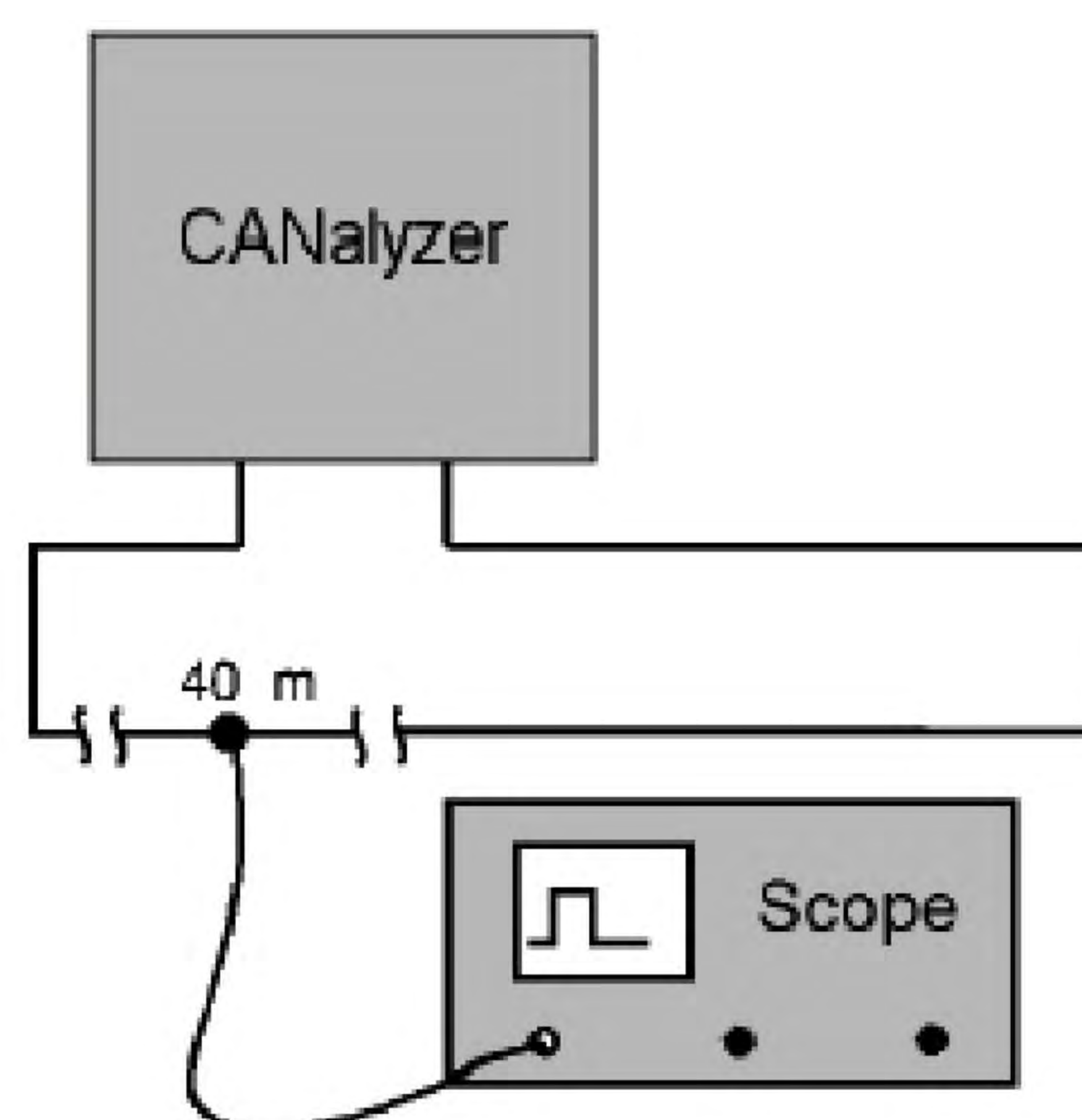


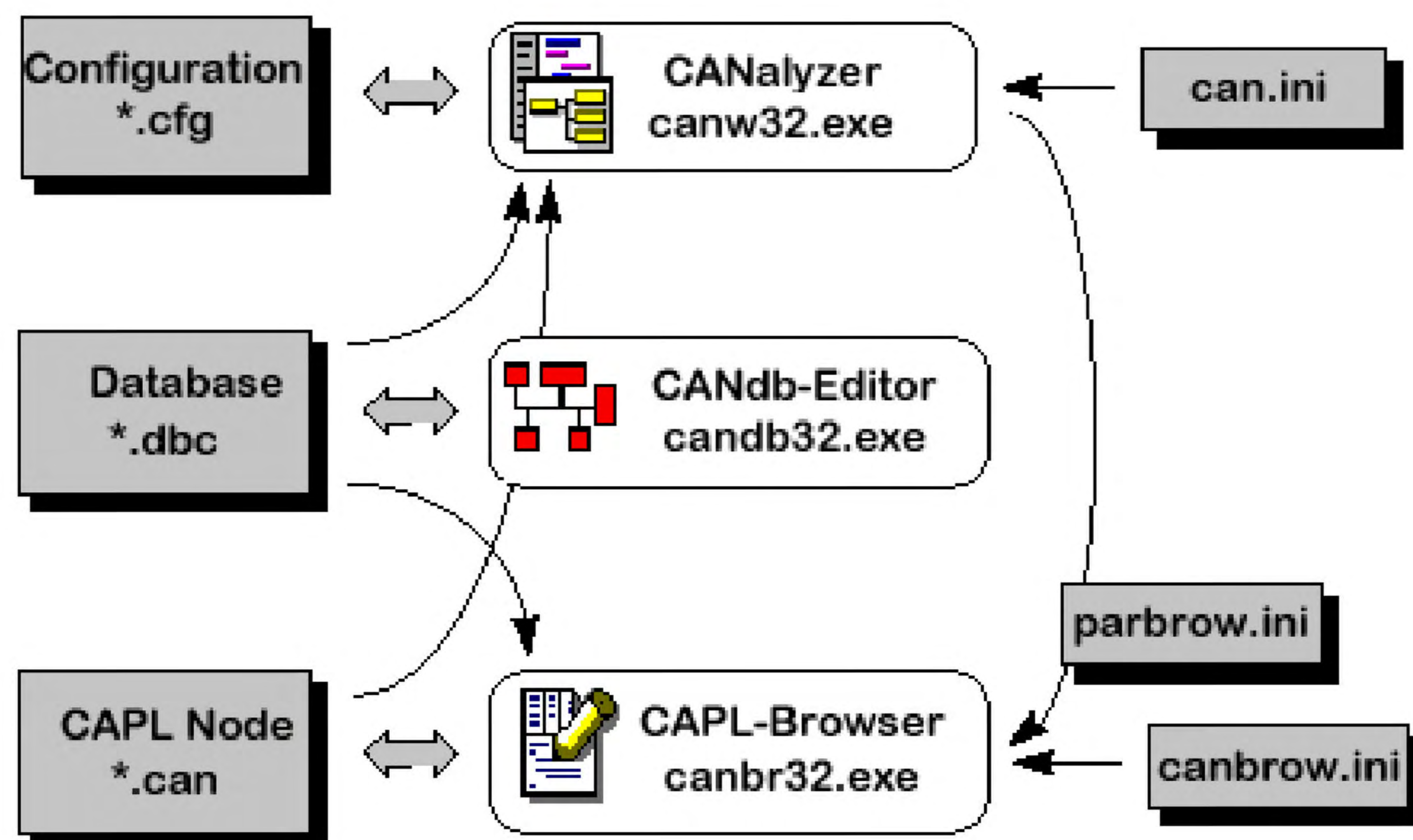
Enllaç entre dos bussos:

D'aquesta manera podrem intercanviar informació entre dos busos CAN, que fins i tot poden ser a diferents velocitats. En general una configuració com aquesta fa treballar al CANalyzer com a switch i permet actuar com a switch actiu o passiu.

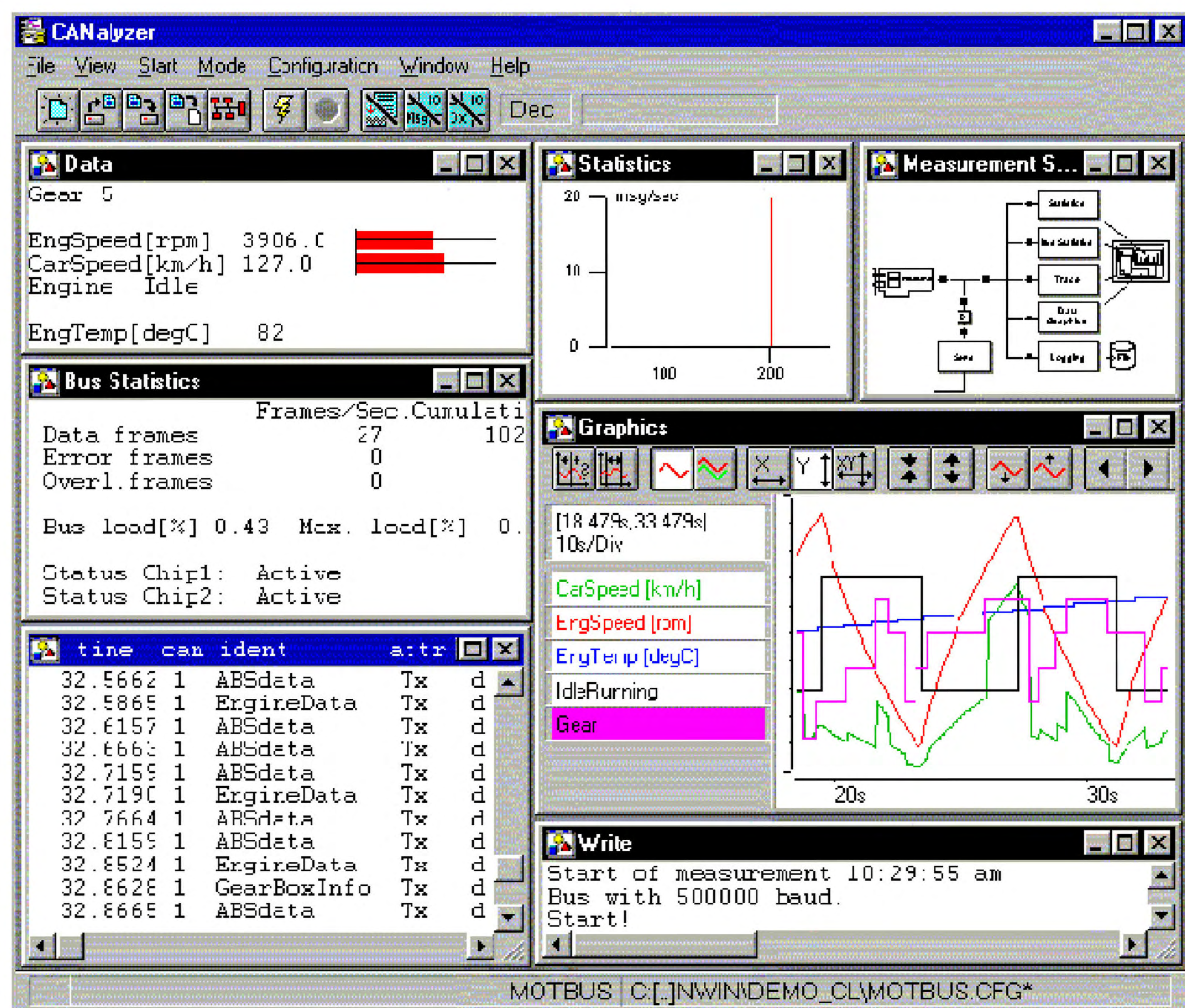
***Testejar la capa física:***

El CANalyzer permet enviar missatges entre els seus dos nodes a través d'un enllaç físic, bus exterior; de tal manera que és possible fer servir-lo per comprovar l'estat de l'enllaç físic utilitzat.



Relació altres programes de l'entorn CANalyzer:

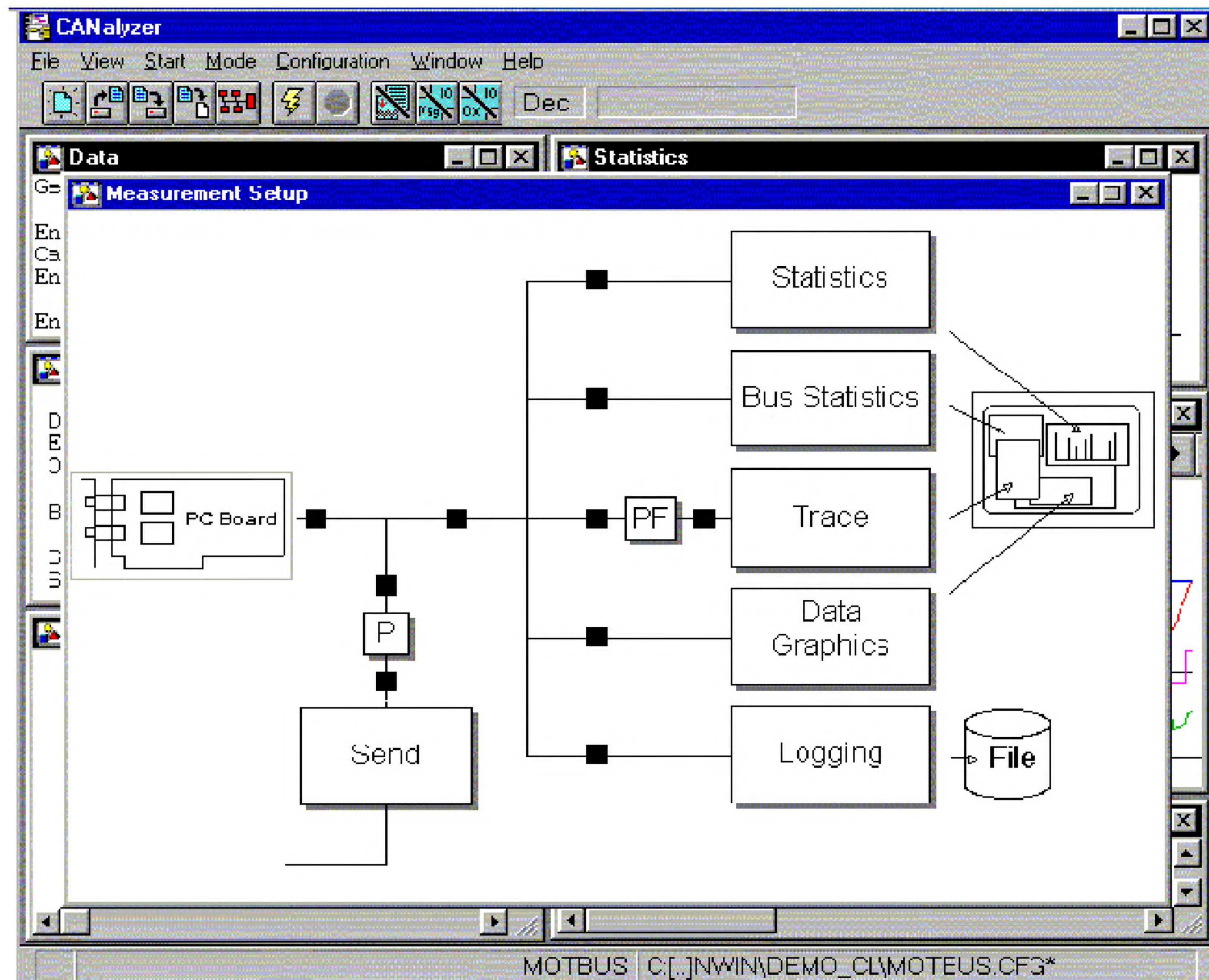
Sense entrar en massa detall veiem unes quantes aspectes més d'aquest programa. A la figura inferior podem veure l'aparença de les finestres que té el CANalyzer per poder monitoritzar el bus i representar les dades observades i els resultats estadístics, com l'ocupació mitja del bus.



S'inclou: llistat del tràfic de dades pel bus, presentació del contingut dels missatges, transmissió de missatges predefinits, transmissió de missatges gravats, estadístiques dels missatges i del bus i gravat de missatges en disc per a la seu posterior anàlisi.

La configuració del CANalyzer i de el seu mode de funcionament es fa mitjançant una interfície "user friendly".

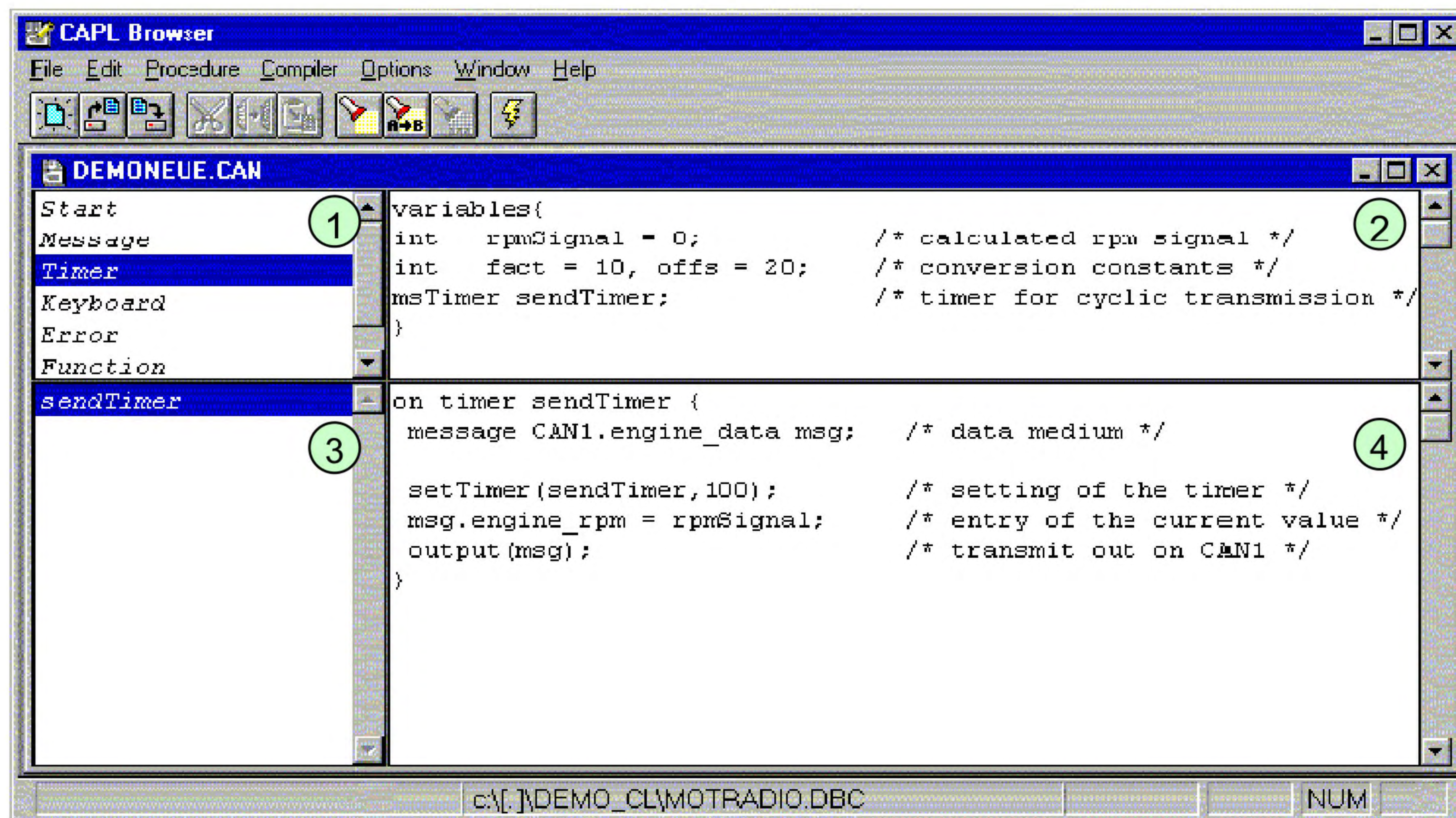
A la figura següent pàgina els blocs que el CANalyzer permet configurar, al diagrama de flux:



En aquest exemple s'ha inserit un bloc de programa [P] que permet actuar sobre el tràfic del bus. Estan activades totes les unitats de mesura, incloent un bloc de filtre [PF] abans de les mesures de Traçat (Trace).

4.4.1.1 CAN BROWSER

Aquest programa està dissenyat per editar y compilar els programes en CAPL. Així com permet associar bases de dades del CANdb al programa CAPL que estigui actiu. La interfície d'aquest programa està dividida en quatre finestres com es mostra a la figura següent:



- ① Presentació dels possibles esdeveniments que poden generar una certa resposta per part del programa.
- ② Variables globals del programa, accessibles des de qualsevol de les altres parts on hi ha codi..
- ③ Dintre dels tipus d'esdeveniment, seleccionat a la finestra número u, en aquesta finestra es selecciona quin esdeveniment en concret d'ells (a l'exemple veiem que quan el Timer conclou es crida sendTimer).
- ④ En aquesta finestra es descriu el procediment que se ha de executar si es produeix l'esdeveniment seleccionat.

4.4.1.2 CAPL

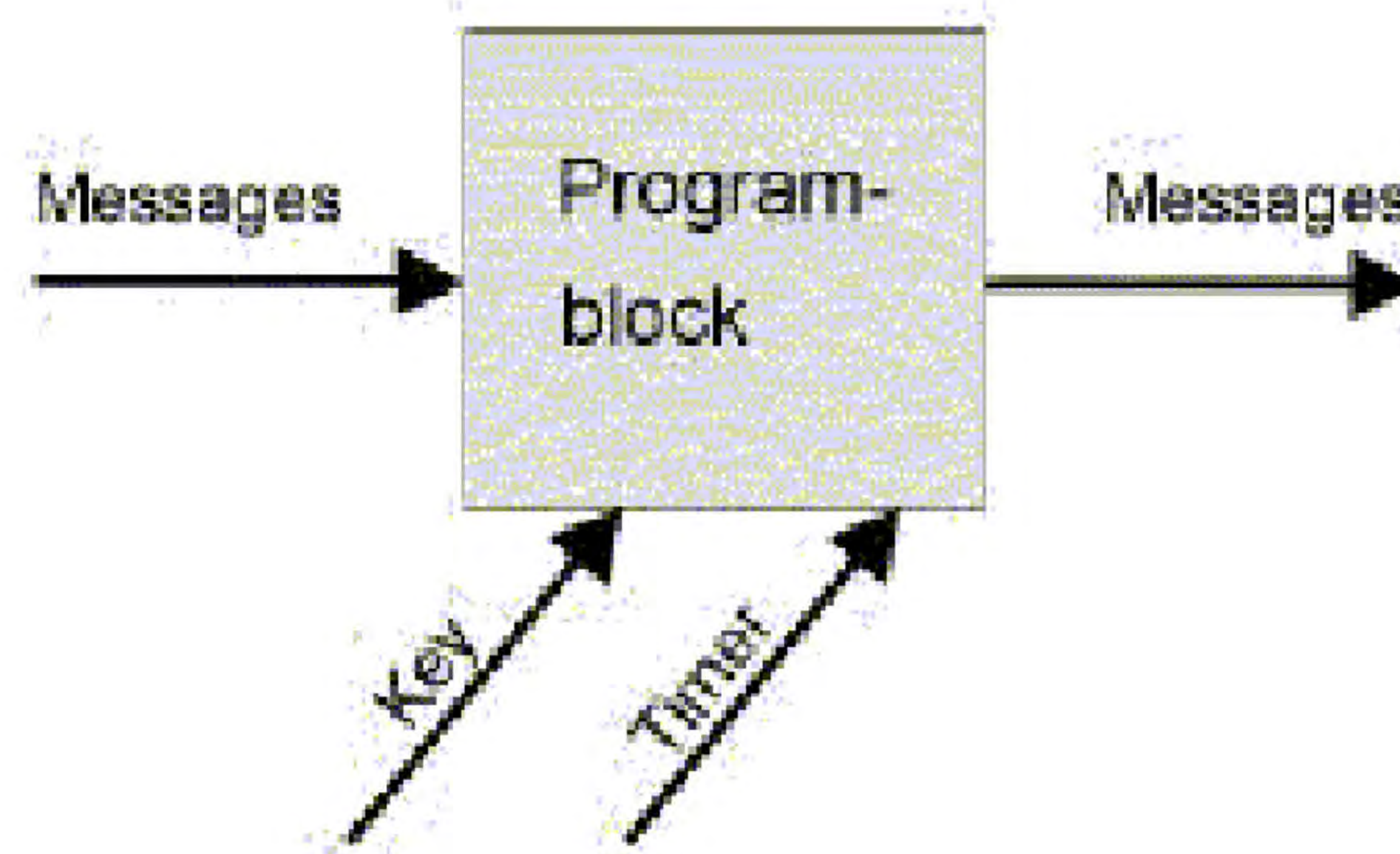
El llenguatge CAPL està sintàcticament basat en el llenguatge de programació C. Un programa CAPL consisteix en la declaració de variables globals (es pot no declarar cap) i en un conjunt de procediments guiats per esdeveniments.

La declaració de variables admet variables dels següents tipus:

- * Int, dword, long, word, byte, char com als llenguatge C
- * Float, double també com al C
- * MsTimer, Timer són variables que corresponen a un rellotge, la primera indica que aquest rellotge conta en ms, i la segons en s.
- * Message són els missatges CAN que han de ser enviats pel CAPL.

Las variables poden ser inicialitzades quan es declaren, les variables que no es declaren explícitament es consideren zero en CAPL. També és possible declarar matrius, vectors i taules de la mateixa manera que es fa en el llenguatge C.

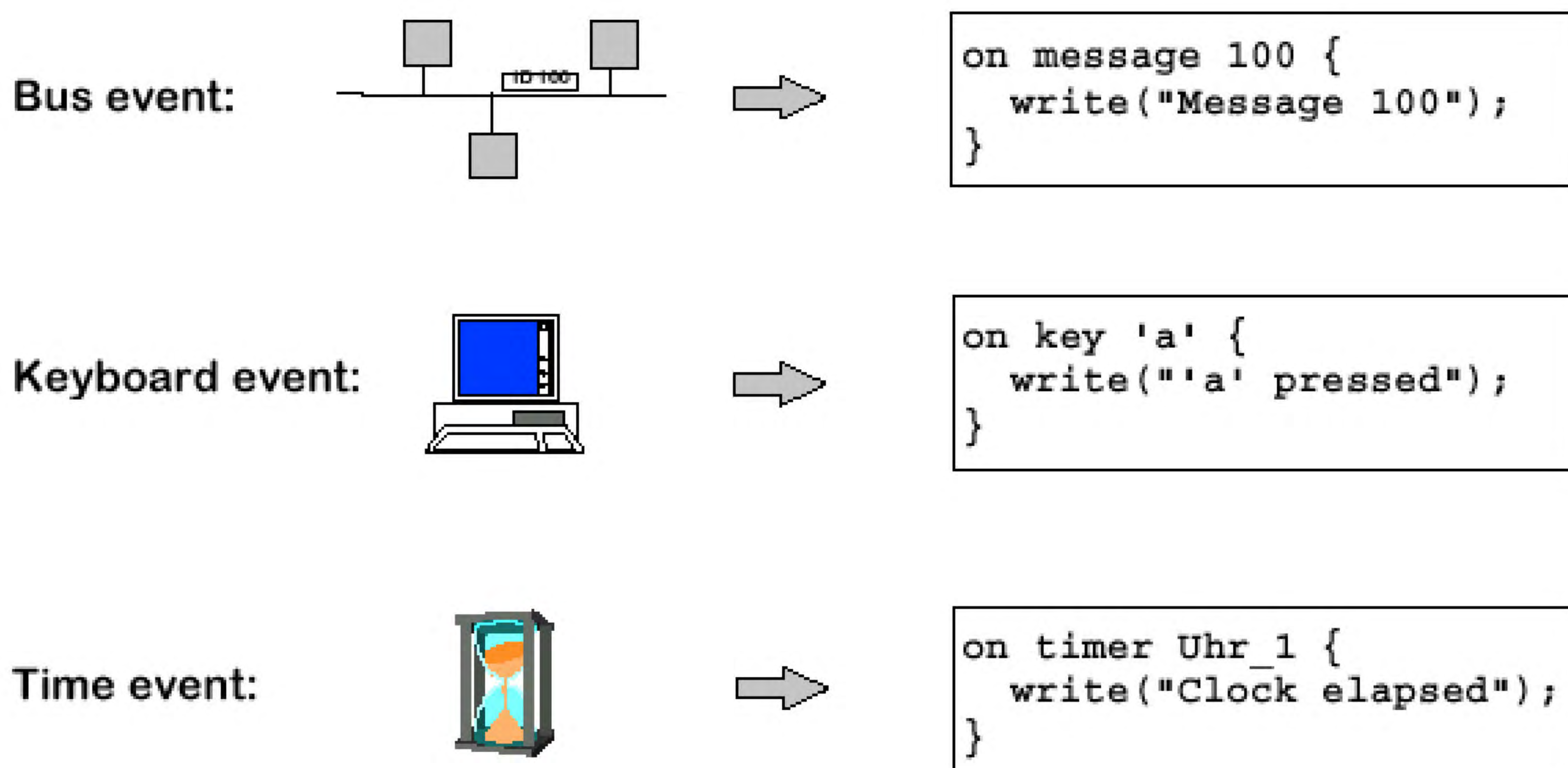
Els procediments que s'implementen poden anar guiats pels següents esdeveniments: inici del programa, detecció d'un missatge, venciment d'un timer, pulsada de una tecla del teclat o detecció d'un marc d'error. Aquests procediments inclouen una declaració de variables locals, que són considerades com a "static" per defecte (en contra de lo que fa el llenguatge C), i un cos del procediment que és la subrutina corresponent.



És possible també implementar subrutines en CAPL, totes les variables són definides com "static". Si no s'especifica un tipus per la subrutina el CAPL ho assumeix com a "void".

Així mateix, el CAPL proporciona una llista de funcions ja construïdes com `resetCan()` que resetja el controlador CAN del PC i moltes més.

A les figures inferiors podem veure exactament el codi que cal utilitzar per les rutines, que van associades a aquests esdeveniments:



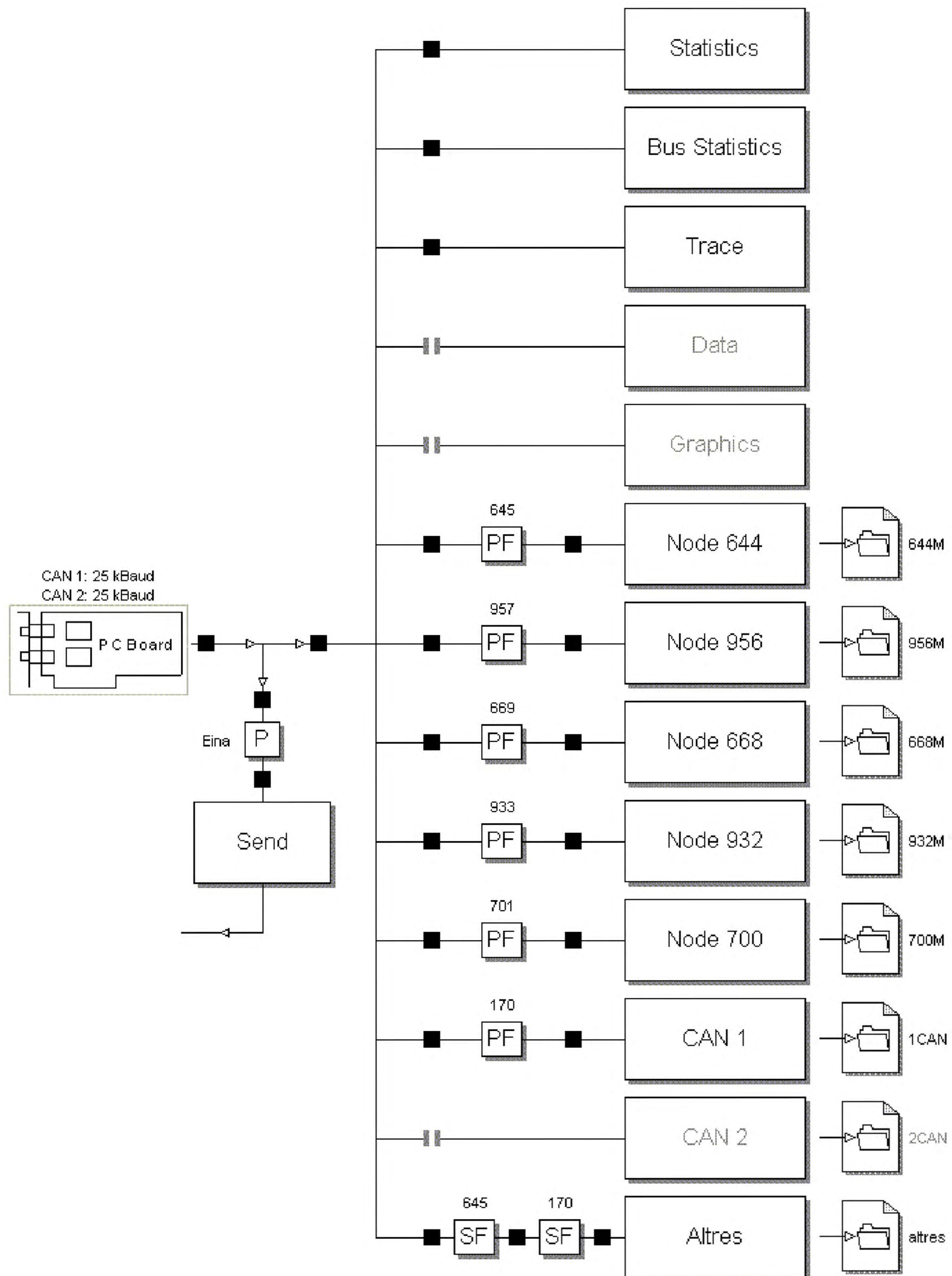
4.4.2 CONFIGURACIÓ

Durant l'etapa de mesures de laboratori del projecte s'han utilitzat moltes configuracions del CANalyzer per tal d'obtenir el màxim de dades i veure les diverses maneres d'obtenir resultats comparables amb els generats amb el jCAN1. S'ha pogut constatar, com ja es coneixia a priori, que en certs aspectes el CANalyzer no és extremadament fiable en el tractament en temps real de les dades al nivell que es podia haver requerit en aquest projecte; per això s'ha optat per treballar amb el CANalyzer com a simple monitoritzador del bus i fer el tractament de les dades a posteriori amb altres eines. D'aquesta manera es treu feina al CANalyzer i al PC i es pot assegurar que les dades són més confinades. A la figura de la pàgina següent es pot veure, a nivell de diagrama de flux, com s'ha configurat el CANalyzer.

Per a totes les mesures dutes a terme s'han realitzat amb aquesta configuració bàsica del CANalyzer. Els canvis s'han fet a tres nivells:

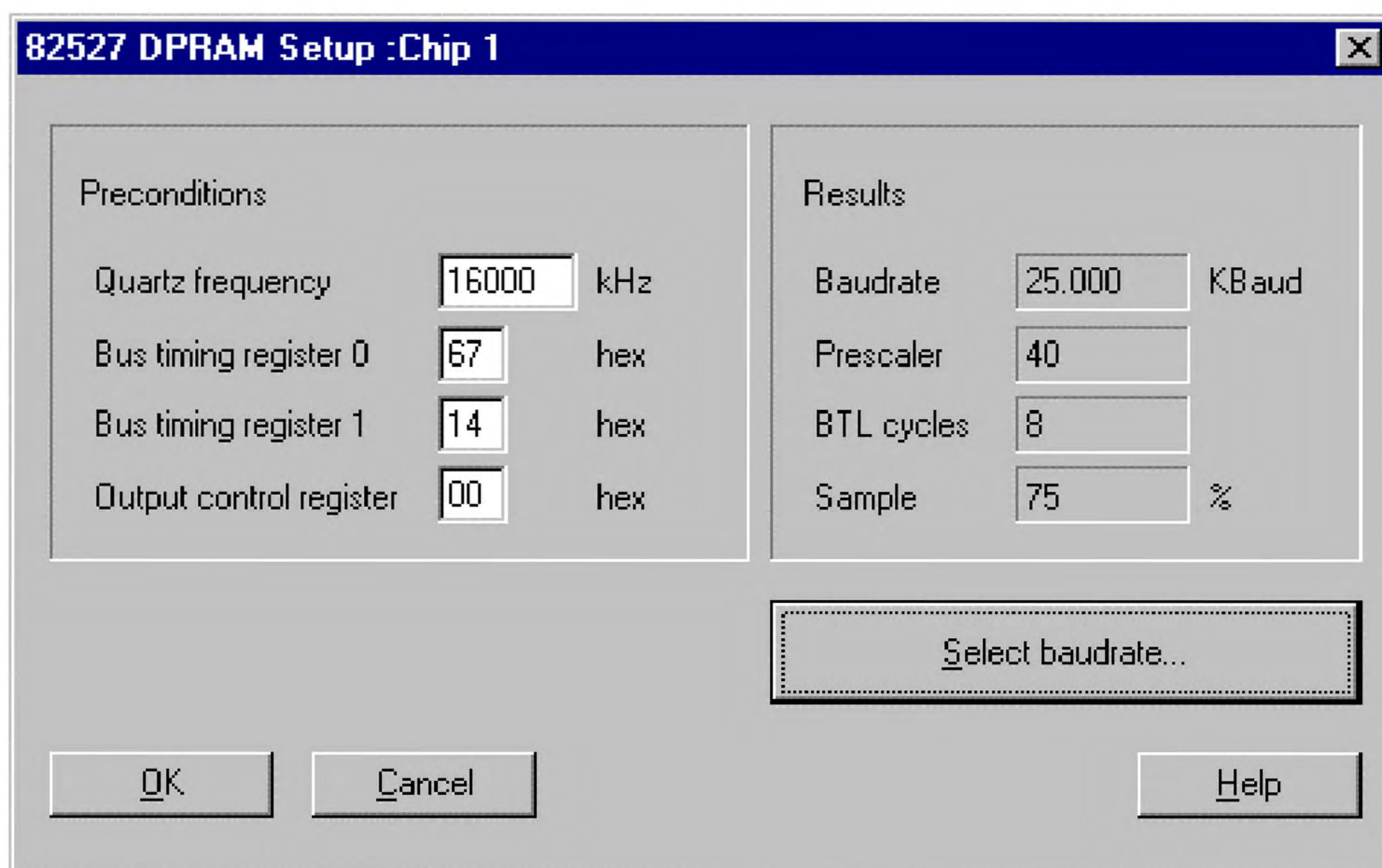
- Canviar la velocitat de bus
- Canviar la configuració dels dispositius SLIO a través del bloc de programa, Eina.
- Desactivar els modes d'estadístiques, gràfiques, traçat, estadístiques de bus i dades per tal d'alleugerir de feina el CANalyzer i al PC.

Diagrama de Flux de la configuració utilitzada del CANalyzer. El bloc inserit, és el programa Eina, en CAPL [P]:

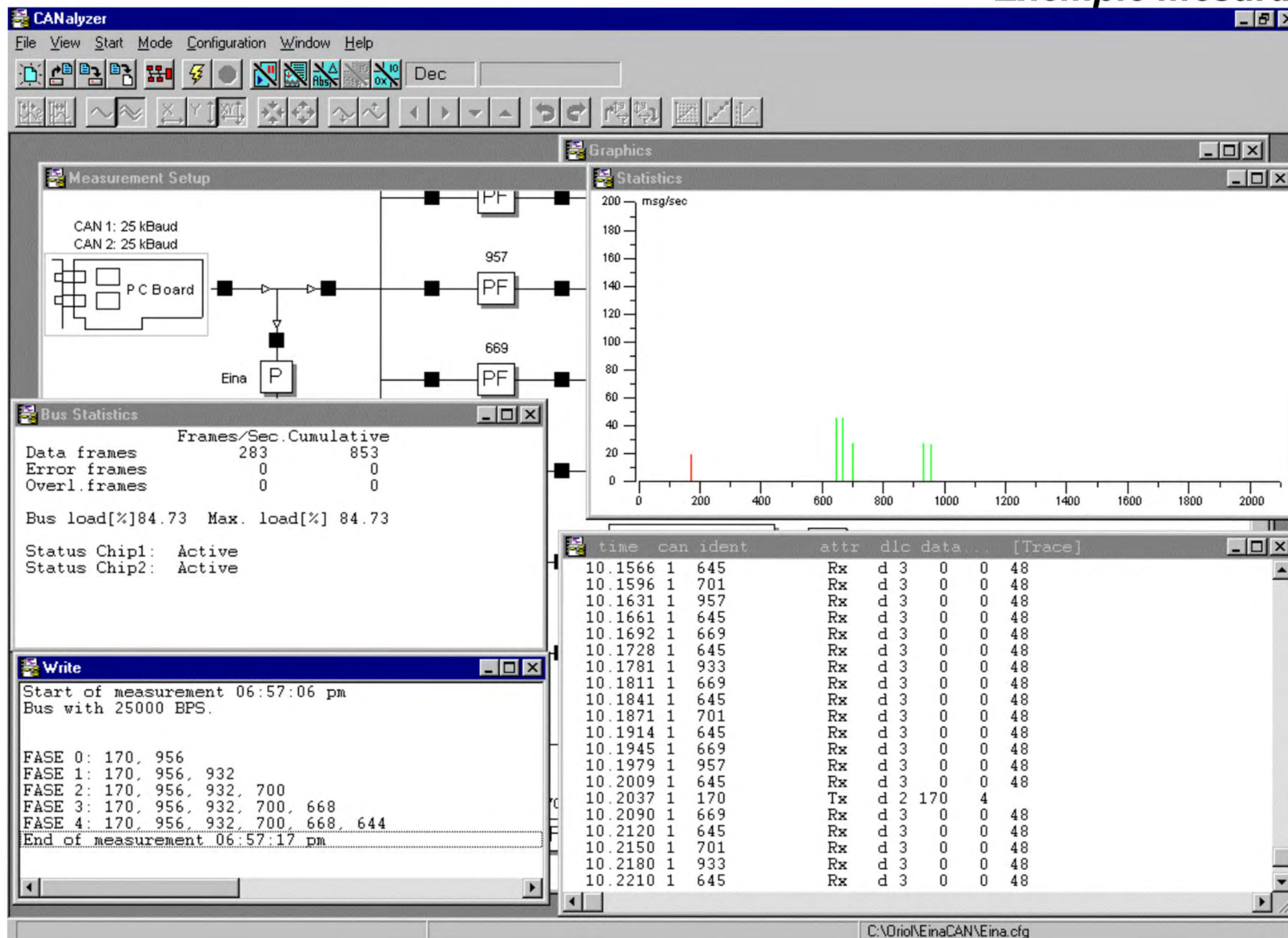


Canviar la velocitat:

Com es veu a l'apartat de mesures s'ha treballat constantment canviant la freqüència del bus; això s'ha fet des del CANalyzer. La freqüència dels nodes SLIO automàticament s'ha adapta a aquesta velocitat; ja que agafen la velocitat del bus des dels missatges que el CANalyzer envia a l'inici, els nodes SLIO venen equipats amb rellotges auto-configurables. La finestra de canvi de velocitat del bus inclou, quantes mostres s'agafen per període de bit:



Un fet els canvis de rellotge en un dels dos nodes CAN que té sota control el CANalyzer també es procedeix a fer el mateix a l'altre node CAN.

Exemple mesura

4.4.3 PROGRAMA EN CAPL

Amb anterioritat a aquest apartat ja s’ha descrit les característiques bàsiques del llenguatge CAPL i del CAPL Browser que és l’eina que proporciona el CANalyzer per desenvolupar programes en aquest llenguatge. Aquí només és vol descriure breument quines tasques es duen a terme amb el programa EinaCAN utilitzat en l’obtenció de dades de laboratori en la configuració del CANalyzer explicat al capítol anterior. El codi font d’aquest programa es troba a l’annex II d’aquest projecte.

El programa EinaCAN s’encarrega de les següents tasques:

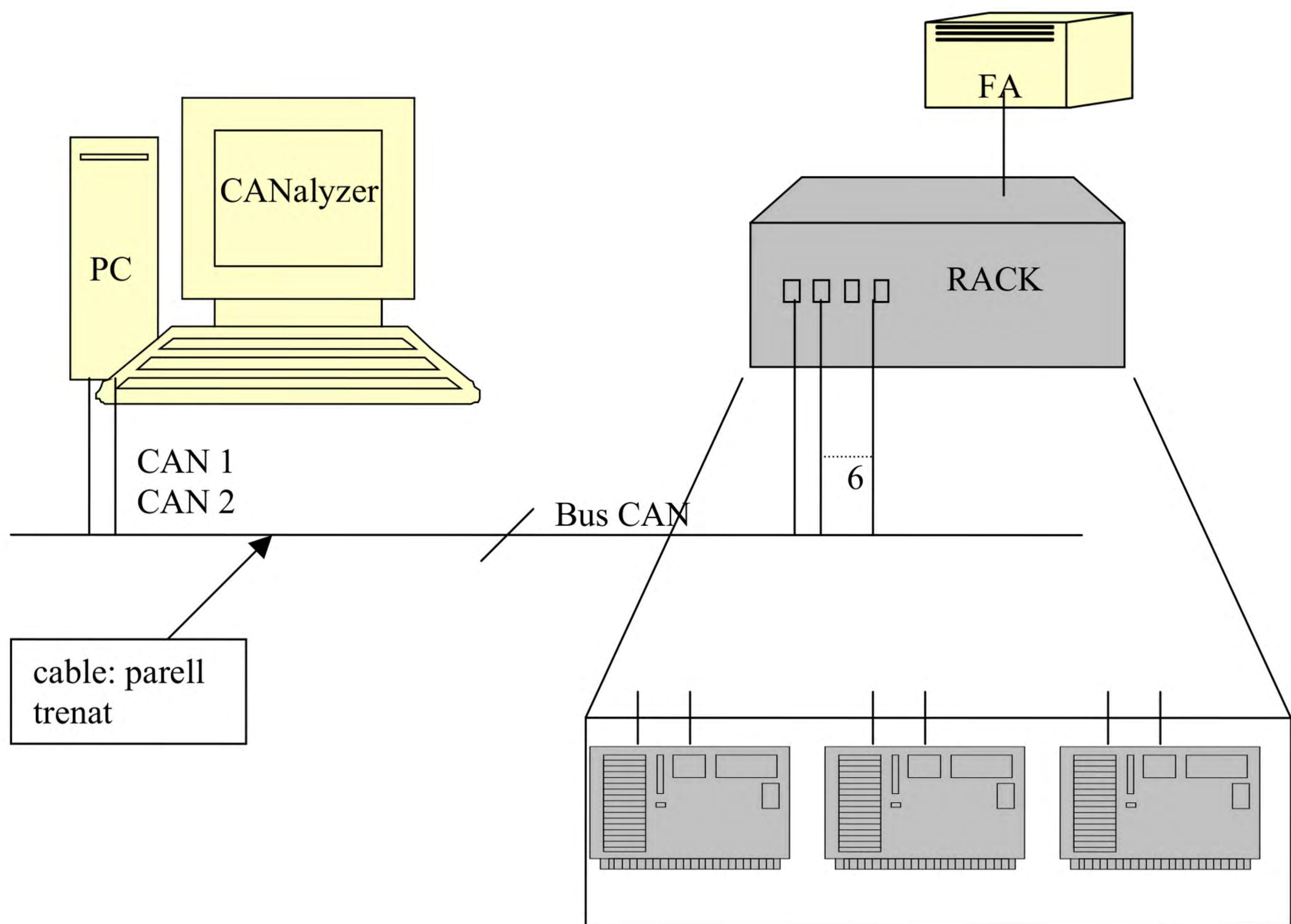
- Inicialitzar el bus CAN i inicialitzar els nodes SLIO, proporcionant-los els missatges necessari perquè pugui calibrar-se i agafar la velocitat amb la qual treballarà el bus.
- Enviar un dels missatge que és enviat de forma periòdica al bus i que també és utilitza pels nodes SLIO per continuar autoajustant-se a la velocitat del bus.
- Enviar els missatges necessària a cada mòdul SLIO per configurar-los i què aquests iniciïn l’enviament de missatges periòdics al bus.
- Enviar missatges per ordenar als mòduls SLIO que indiquin la seva activitat per mitja dels LEDs que disposen cada un.
- Iniciar el procés d’activitat per a les mesures.

Amb el llenguatge CAPL és poden fer moltes més coses; però com més endavant es veurà, per tal de reduir la càrrega de la CPU s’ha simplificat al màxim el programa per tal de focalitzar tot el procés a fer una adquisició de dades el màxim de fiables possible. El programa ha de ser modificat cada cop per canviar la freqüència amb la qual ha de treballar en nodes CAN1 que també entra a l’estudi del bus. A l’iniciar el programa disposem de diverses opcions, triables amb lletres del teclat. Per aturar el programa i les mesures només cal pitjar al botó de STOP del CANalyzer.

Tecles	Acció
‘1’, ‘2’, ‘3’, ‘4’ i ‘5’	Iniciar l’activitat dels nodes amb el número respectiu.
‘q’, ‘w’, ‘e’, ‘r’ i ‘t’	Indicar amb el LED respectiu que el node (1 a 5) està actiu i despert al bus.
‘z’ i ‘x’	Encendre i apagar tots els LEDs
‘c’	Encendre i apagar indicadors d’activitat.
‘j’	Iniciar procés de mesures.

4.5 ENTORN LABORATORI

Un cop ja es tenen tots els dispositius que formen l'entorn de laboratori aquests s'interconnecten com mostra la figura inferior. En el moment de finalitzar aquest projecte encara es desconeixien les necessitats d'alimentació de la resta de dispositius que han de treballar al bus, i que forma part d'un projecte més gran del departament d'electrònica, així no s'ha pogut integrar encara una font d'alimentació al rack i s'ha fet servir una font d'alimentació externa del laboratori.



4.6 PROCESSAT RESULTATS

Amb tot el sistema muntat ja és viable obtenir resultats. L'objectiu ara és comparar resultats obtinguts en aquest entorn de laboratori amb resultats teòrics obtinguts amb el programa.

Per això s'han realitzat diverses mesures. Cada mesura representa un sistema a simular amb diversos valors, tot i que de cada sistema se'n han fet diverses mesures per verificar-ne la validesa.

És important recordar que el model teòric que tenim ens serveix per fer una predicció de programabilitat en el pitjor cas. Per això els resultats obtinguts del laboratori no poden ser directament comparats. Ha calgut l'ús de l'estadística per obtenir barems de comparació.

A més a més, s'ha trobat diverses maneres diferents de plantejar els sistemes problema per tal de treure'n conclusions. En aquest capítol del projecte s'explica com s'han processat les dades amb uns quants exemples, que són mostres de totes les mesures dutes a terme.

4.6.1 DESVIACIÓ EN FUNCIO DE NODES

Amb aquest mètode d'obtenció de dades es purament observacional i sense la participació dels nodes CAN del PC en mode actiu. És el mètode més pur. És tracta de fer servir el CANalyzer com a espia, a part de la inicialització i configuració dels nodes SLIO. Es fa treballar els nodes SLIO amb diversos períodes, i se'ls assigna la prioritat pel mètode DM (Deadline Monotonic. Segons el possibilitats del CANalyzer es fan mesures de certs períodes continus. D'aquesta manera és pot veure com es comporten els missatges segons la càrrega del bus i quina són els temps de latència que pateixen el nodes.

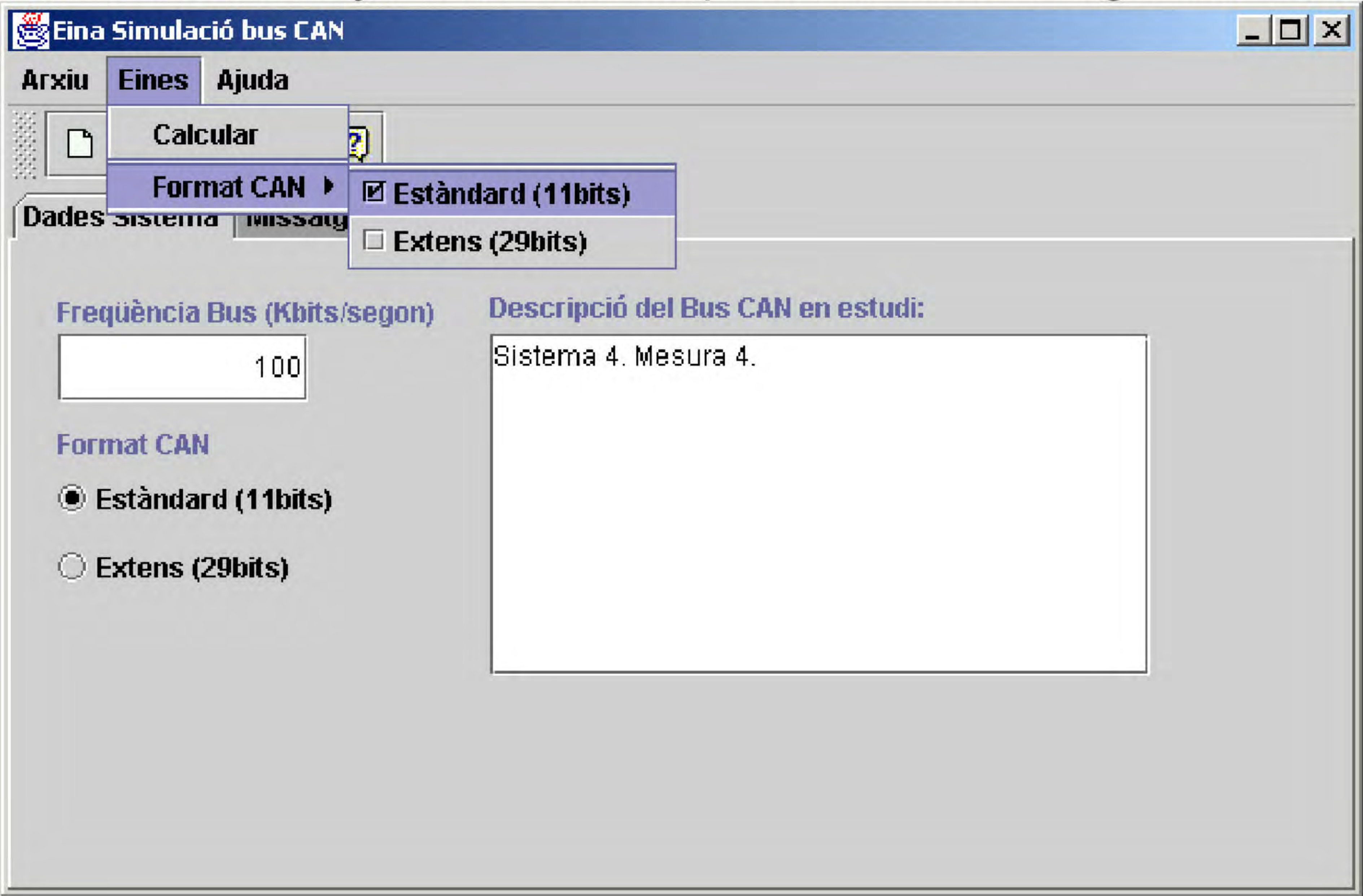
Per explicar aquest mètode amb detall es descriuen dos sistemes problema sotmesos a estudi, un amb càrrega normal i un altre amb càrrega elevada de missatges al bus.

4.6.2 EXEMPLE I

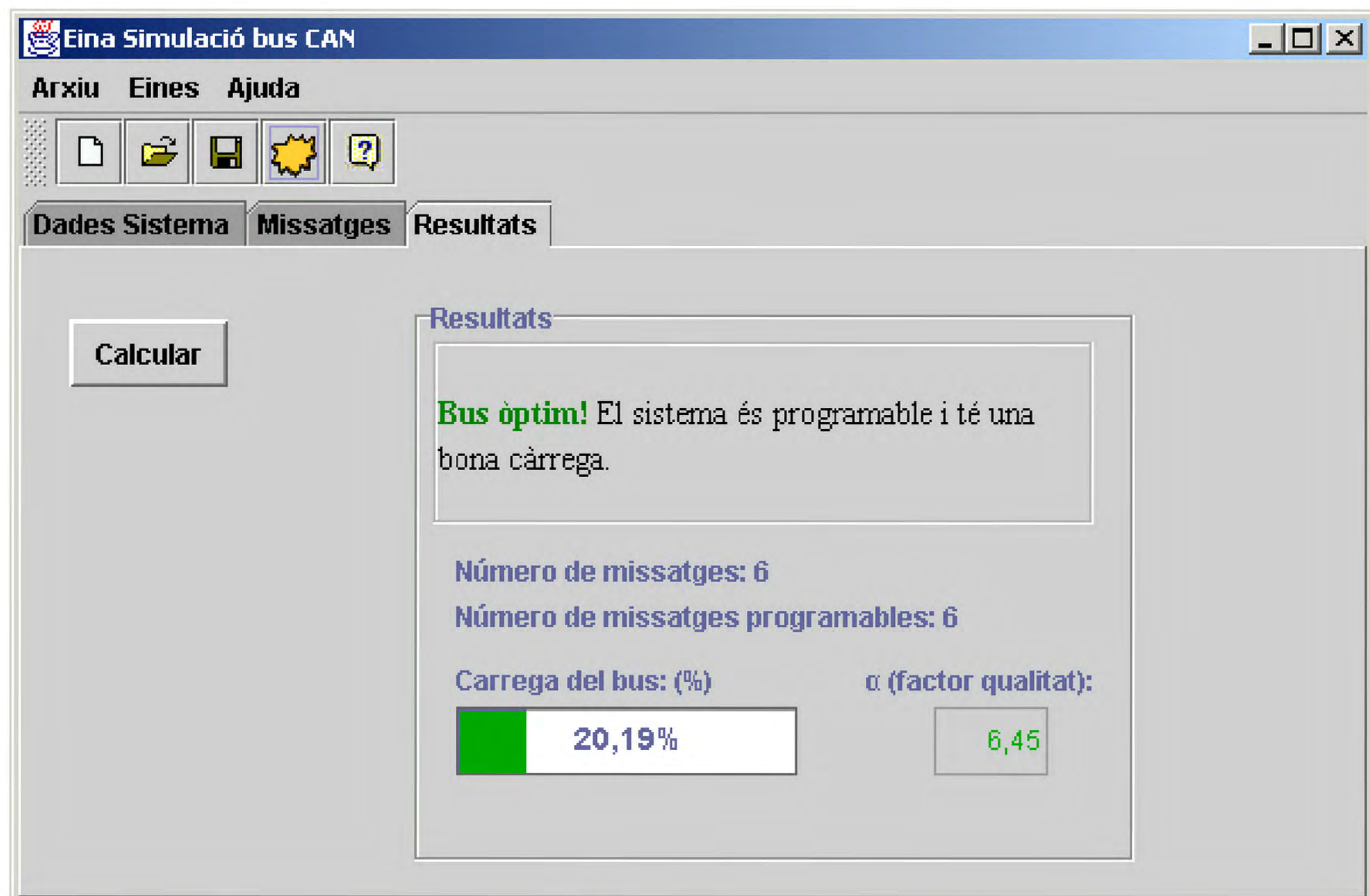
El primer que s'ha fet es configurar els circuits temporitzadors de cada node amb un seguit de freqüències. Els valors dels períodes no tenen cap importància concreta, l'important és que estan una mica espaiats per fer més visible la idea de que els missatges amb més prioritat són aquells amb freqüència més gran. També el conjunt de freqüències triades respecta a la freqüència d'us del bus permet que el sistema treballi sota unes condicions de càrrega baixa. No és el millor sistema per poder observar els problemes de l'alta càrrega, objectiu del projecte; però és útil per explicar el mètode mesura utilitzat i veure el comportament diferent amb l'altre exemple on la càrrega és més elevada. Les dades per aquest problema són les següents:

Node	ID	Freqüència(Hz)	Període (ms)	Deadline (ms)
CAN1	170	20.1	49.7	10
SLIO-1	645	107.5	9.3	15
SLIO-2	668	33.3	30.0	20
SLIO-3	700	42.2	23.7	25
SLIO-4	932	25.1	39.8	30
SLIO-5	956	20	49.98	35

Un cop posades aquestes dades a cada mòdul, introduïm les dades al programa de simulació jCAN1. Amb el qual obtenim els següents resultats:



Id	Nom	Jitter (us)	Període (...)	Deadline ...	Num. Bytes	Temps (us)	% d'us	Ok
170	cap	0	49690	10000	2	1550	1.46910...	✓
644	cap	0	9290	15000	3	2370	8.82669...	✓
668	cap	0	30000	20000	3	3190	2.73333...	✓
700	cap	0	23690	25000	3	4010	3.46137...	✓
932	cap	0	39800	30000	3	4830	2.06030...	✓
956	cap	0	49970	35000	3	5650	1.64098...	✓
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐
0	cap	0	0	00		0	0.00	☐

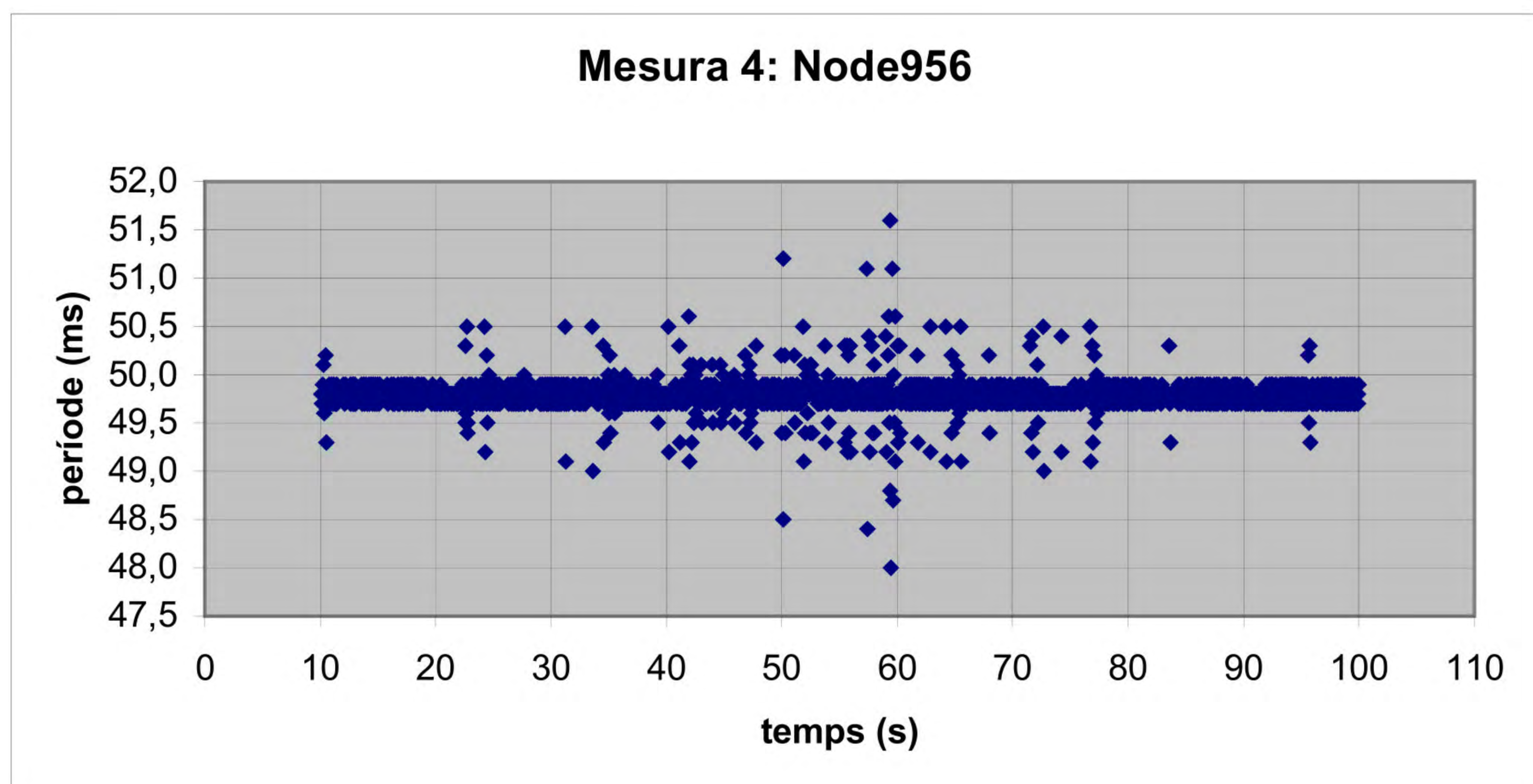


Ara cal comparar aquest resultat amb el obtinguts amb el CANalyzer. Però com que el programa està pensat per trobar situacions límit i pitjors casos amb aquest sistema serà difícil. El que queda clar és que el programa troba que el sistema serà programable, amb un bon factor de qualitat i una càrrega baixa. De totes maneres fem una cronologia de que hem mesurat amb el CANalyzer que ens servirà per veure la tendència i l'efecte sota estudi. Em dividit el sistema en onze etapes, totes elles de 10 segons

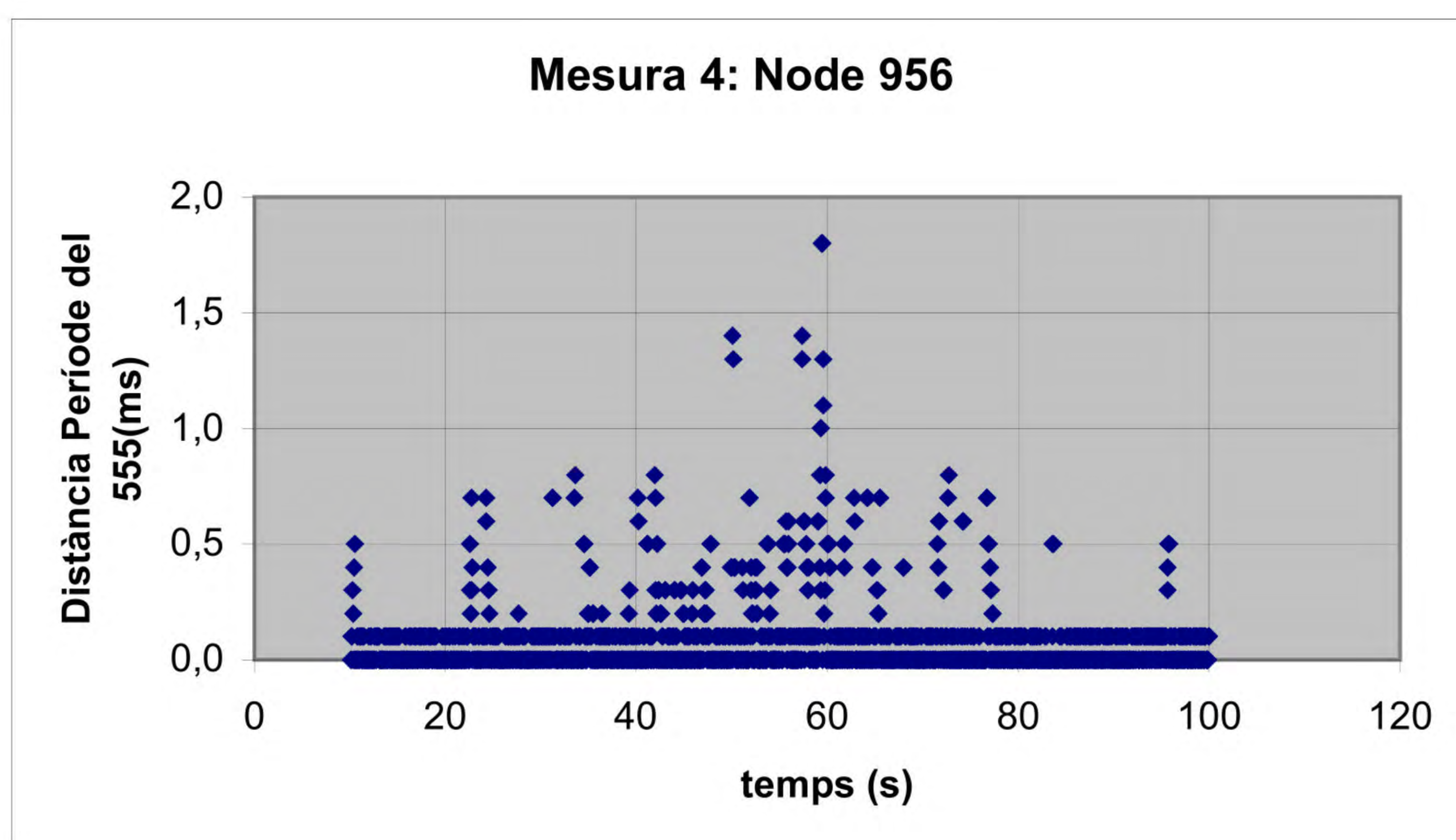
Etapa	Temps	Nodes actius
1a	0s-10s	170
2a	10s-20s	170, 956,
3a	20s-30s	170, 956, 932,
4a	30s-40s	170, 956, 932, 700
5a	40s-50s	170, 956, 932, 700, 668
6a	50s-60s	170, 956, 932, 700, 668, 644
7a	60s-70s	170, 956, 932, 700, 668
8a	70s-80s	170, 956, 932, 700
9a	80s-90s	170, 956, 932,
10a	90s-100s	170, 956
11a	100s-110s	170

Ara estudiem el comportament dels missatges enviats pel node 956 al bus, entre les etapes 2a i 10a. La següent gràfica mostra quin a estat el període entre missatges enviats al bus. És pot observar que la major part tenen una distància entre ells de entre 49,7 i 50 que correspon a la freqüència a la que ha estat programat el node 956, amb un petit marge d'error propi de les limitacions del CANAlayzer. Però observem, que hi ha de forma puntual, moments en què el període és dispara a valors no centrats.

Aquest valors sempre tenen un per sobre i un per sota. Aquest efecte és normal: En un moment donat el node 956 no pot enviar un missatge al bus perquè aquest està ocupat; quan el bus tornar a estar lliure el node 956 pot enviar el missatge. Però és clar que el proper cicle d'interrupció generat pel circuit temporitzador de node 956 ara serà més proper al darrer moment d'enviar un missatge i és aquest el temps inferior que observem. Per corregir aquest efecte hem generat una altra gràfica on només és té en compte la diferència respecte el període mig.



i amb el tractament adient podem trobar els retards (latència):



A simple vista és pot observar que en les etapes centrals hi ha més casos de missatge que s'han retardat. Això és obvi, en les etapes centrals hi ha més càrrega i per tan més possibilitats que un missatge trobi el bus ocupat. L'etapa de 50segons a 60segons és la representativa de la simulació duta a càrrec amb el jCAN1. Segons el JCAN1 en el pitjor dels casos el node 956 pot arribar a tenir un retard de fins a 5.65ms, i efectivament a la gràfica

com a molt s'observa un instant on el retard del missatge ha arribat a ser de 1.8ms i estem dintre de la teoria.

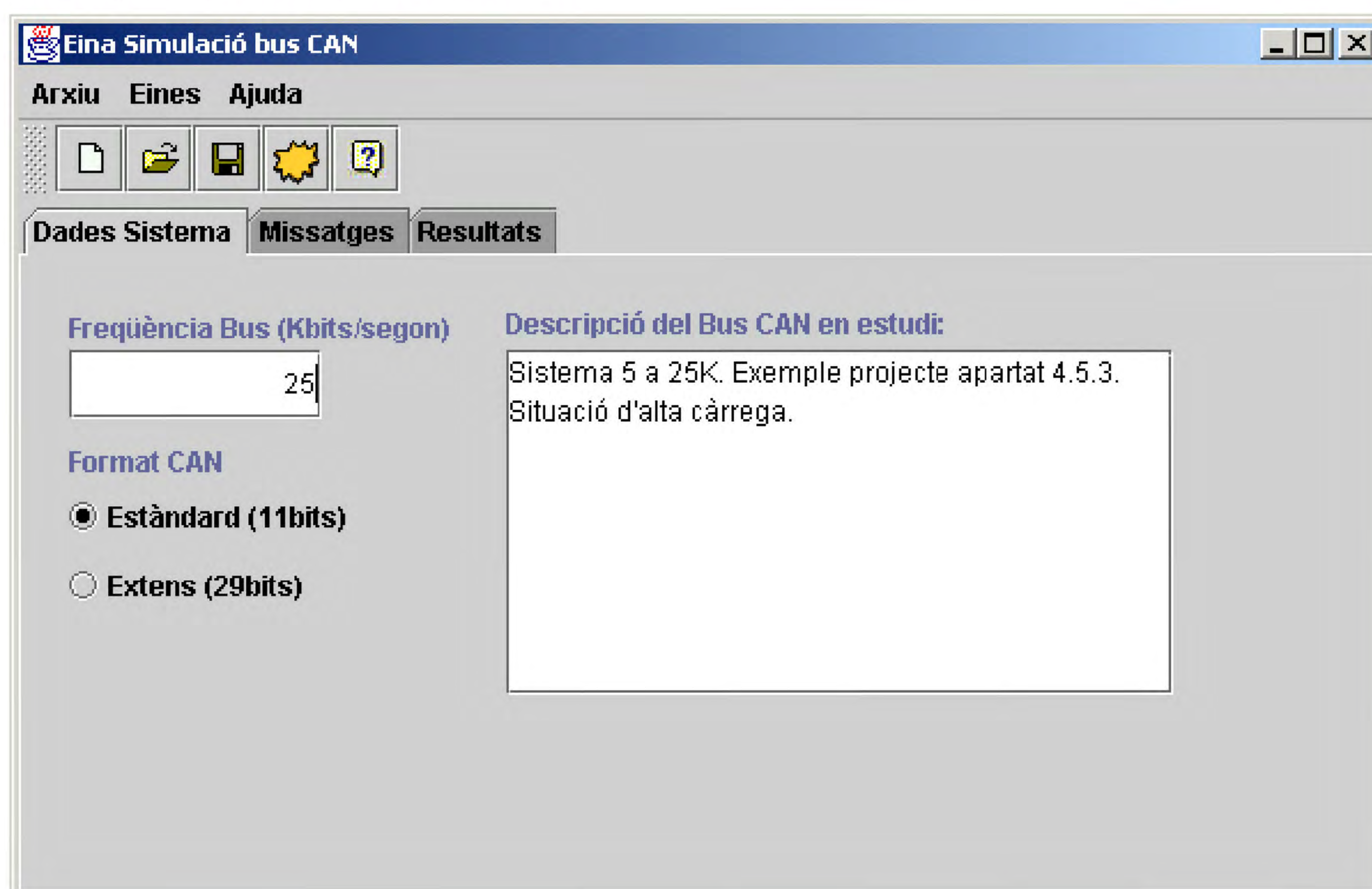
El procediment per treure més conclusions passa per fer moltes més mesures i fer molta estadística dels resultats obtinguts, part que es dur a terme al capítol de conclusions. Abans però veiem un nou exemple amb explicacions d'un sistema més crític.

4.6.3 EXEMPLE II

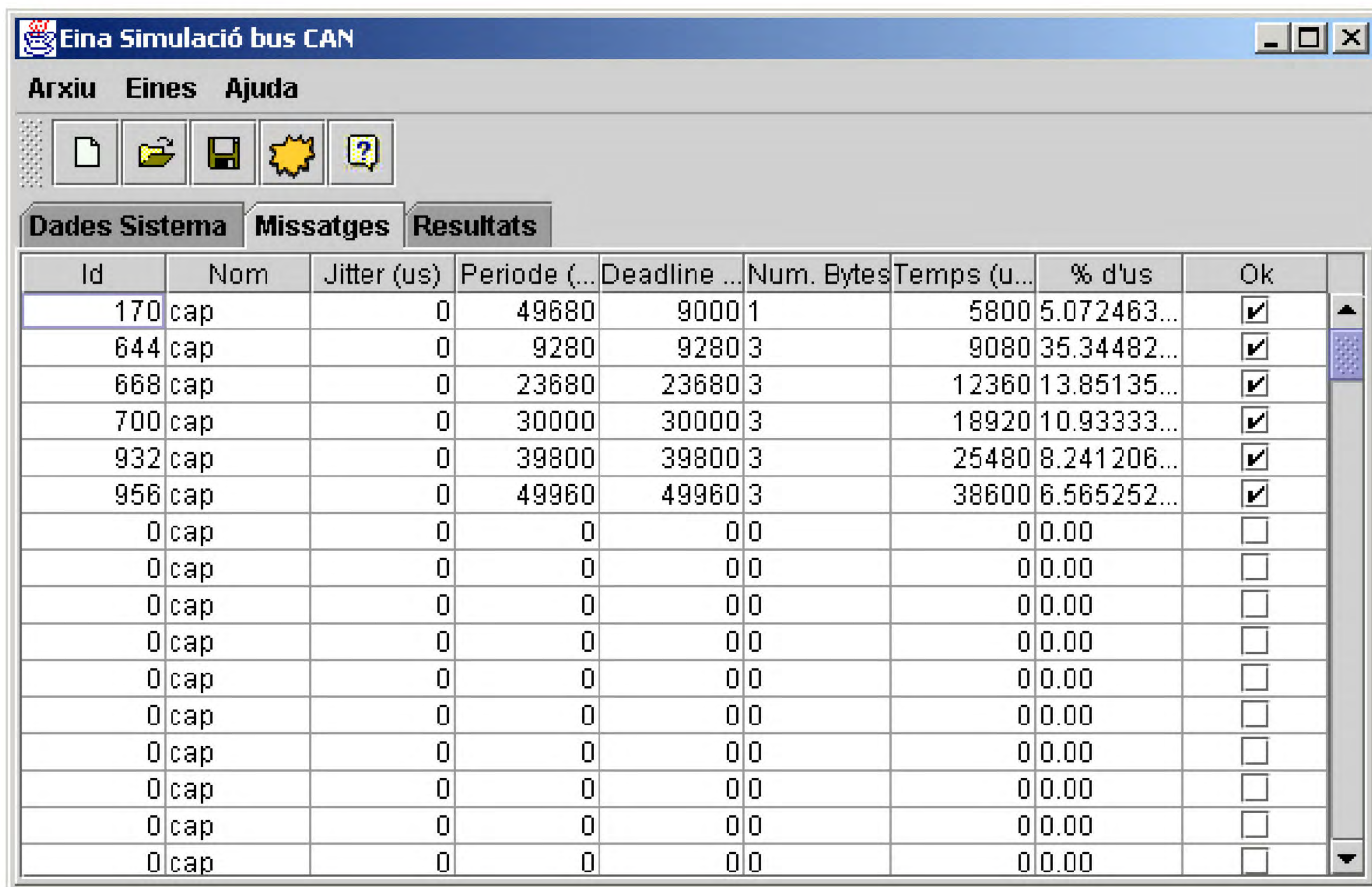
Com a l'exemple anterior, el primer que s'ha fet es configurar els circuits temporitzadors de cada node amb un seguit de freqüències, seguint els mateixos criteris d'abans (on els valors dels períodes no tenen cap importància concreta, l'important és que estan una mica espaiats per fer més visible la idea de que els missatges amb més prioritat són aquells amb freqüència més gran). Ara les freqüències triades respecta a la freqüència d'us del bus permet que el sistema treballi sota unes condicions de càrrega alta. Amb aquestes dades es d'esperar una situació límit al bus. Situació d'especial interès. Les dades per aquest problema són les següents:

Node	ID	Freqüència	Període	Deadline
CAN1	170		49.7	49.7
SLIO-1	645		9.3	9.3
SLIO-2	668		30.0	30.0
SLIO-3	700		23.7	23.7
SLIO-4	932		39.8	39.8
SLIO-5	956		49.98	49.98

Un cop posades aquest dades a cada mòdul, introduïm les dades al programa de simulació jCAN1. Primer posem les dades generals del sistema, freqüència del bus, que el bus és del tipus estàndard i una breu descripció del sistema en estudi:

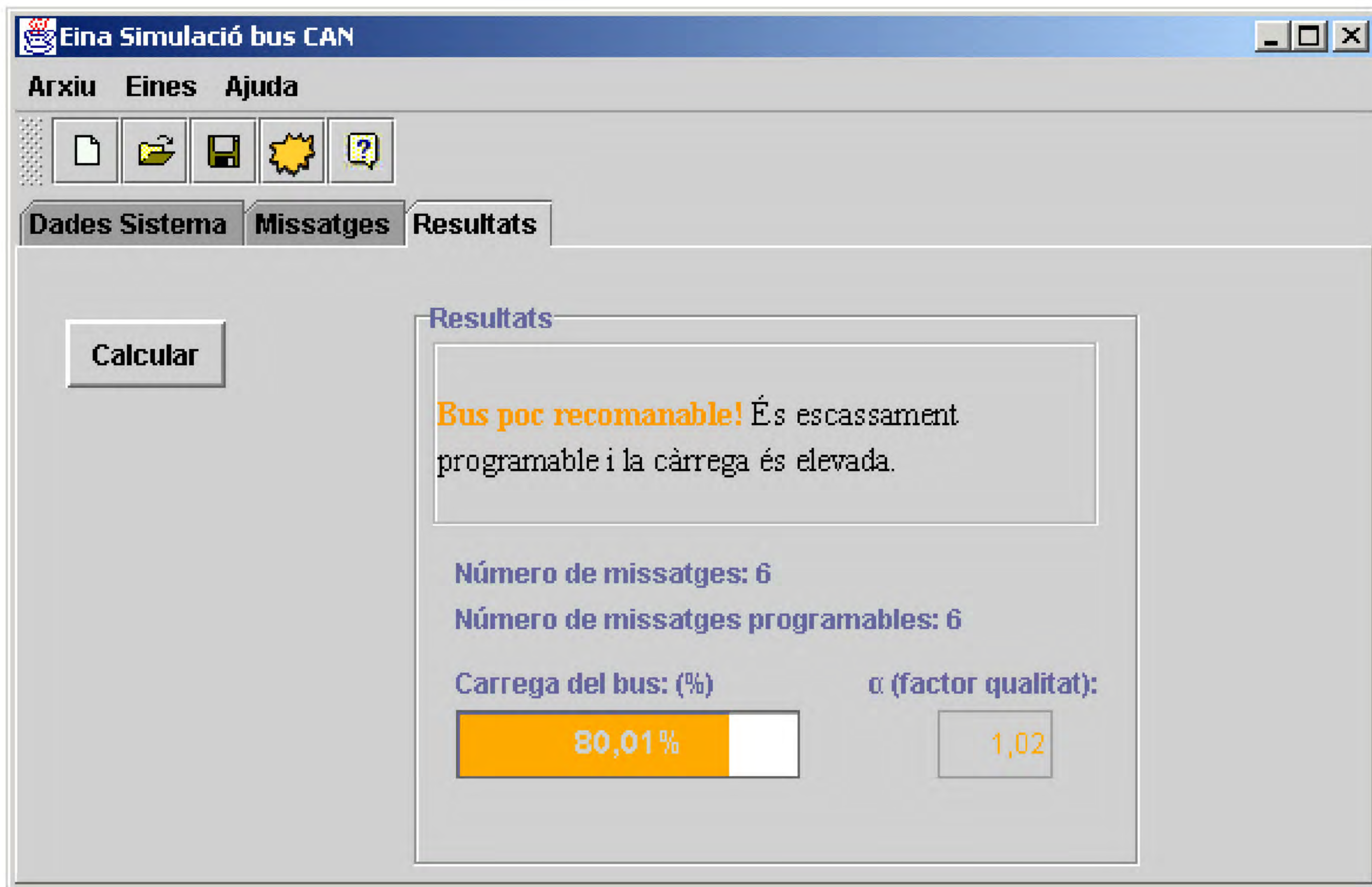


Després introduïm les dades dels nodes que treballaran en el bus i calculem:



Id	Nom	Jitter (us)	Periode (...)	Deadline ...	Num. Bytes	Temps (u...	% d'us	Ok
170	cap	0	49680	9000	1	5800	5.072463...	☒
644	cap	0	9280	9280	3	9080	35.34482...	☒
668	cap	0	23680	23680	3	12360	13.85135...	☒
700	cap	0	30000	30000	3	18920	10.93333...	☒
932	cap	0	39800	39800	3	25480	8.241206...	☒
956	cap	0	49960	49960	3	38600	6.565252...	☒
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐
0	cap	0	0	0	0	0	0.00	☐

Com a resultat obtenim que la càrrega del bus és molt alta, tal com volíem i que tots els missatges són programables; però un factor de qualitat irrisori de només $\alpha = 1.02$ al límit de ser un sistema no programable. En definitiva tenim el sistema perfecte per veure si la eina de simulació s'adiu amb el que passarà realment en el sistema de laboratori.



Resultats

Bus poc recomanable! És escassament programable i la càrrega és elevada.

Número de missatges: 6
 Número de missatges programables: 6

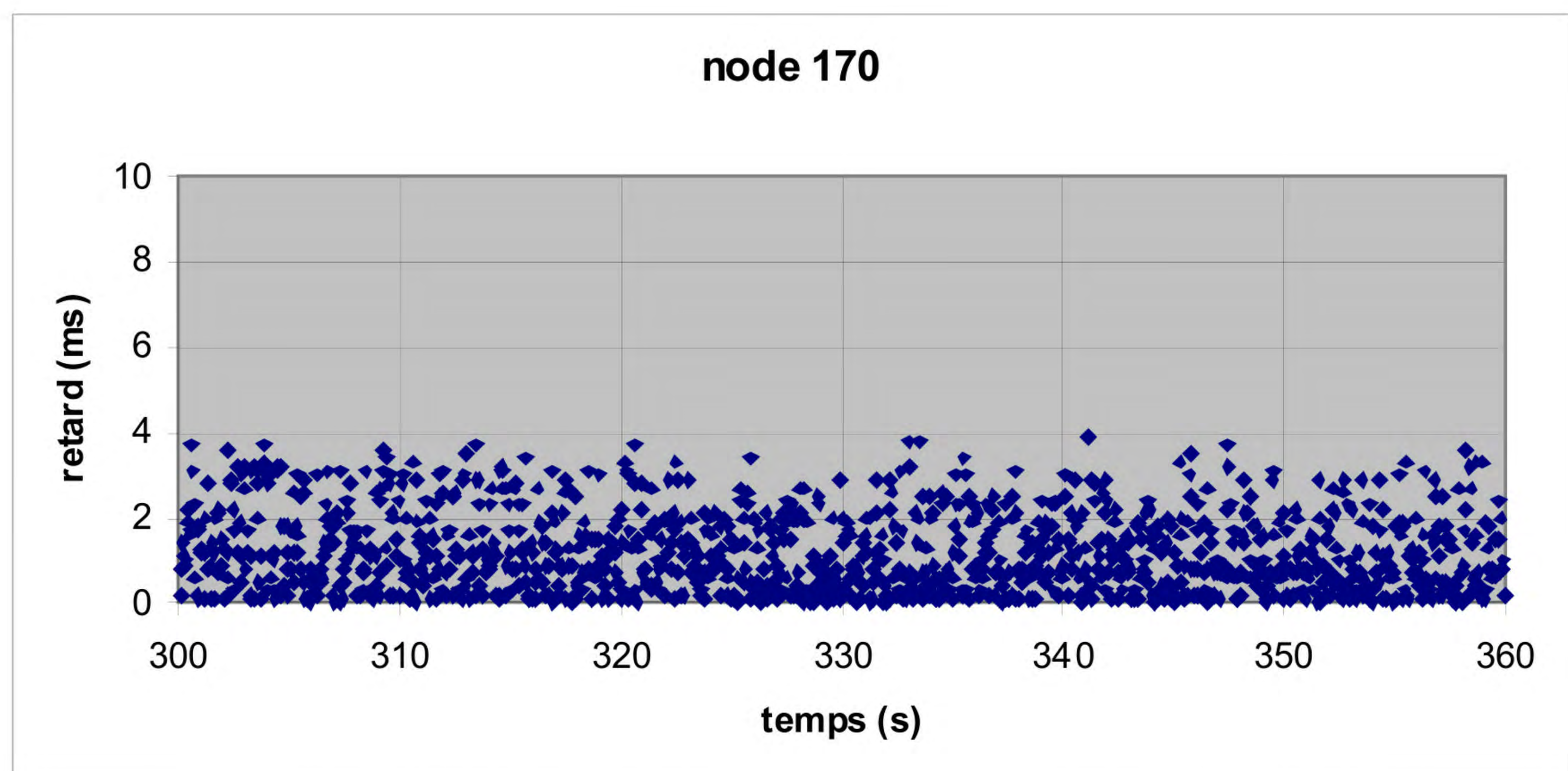
Carrega del bus: (%) **80,01 %** α (factor qualitat): **1,02**

Ara hem de posar aquestes mateixes dades al bus de laboratori a través del CANalyzer i el programa en CAPL que controla tot el sistema. Per fer la simulació s'han fet diverses proves fins trobar el sistema fidel d'obtenir dades del bus. Aquest sistema consisteix en dividir el temps d'estudi en sis períodes de 60 segons on el número de missatges actius és diferent; això ens permet veure en una sola mesura l'evolució de la programabilitat del bus en funció del número de nodes actius. Encara que el període més interessant és el de càrrega més elevada, i que ja hem introduït al jCAN1. Comencem comentant aquest període de 60 segons on tots els nodes estan treballant i a posteriori farem un anàlisi comparatiu del mateix bus amb càrregues més baixes.

En les següent pàgines podem observar el comportament node per node. L'adquisició i tractament de dades ja s'ha explicat amb anterioritat.

Node 170

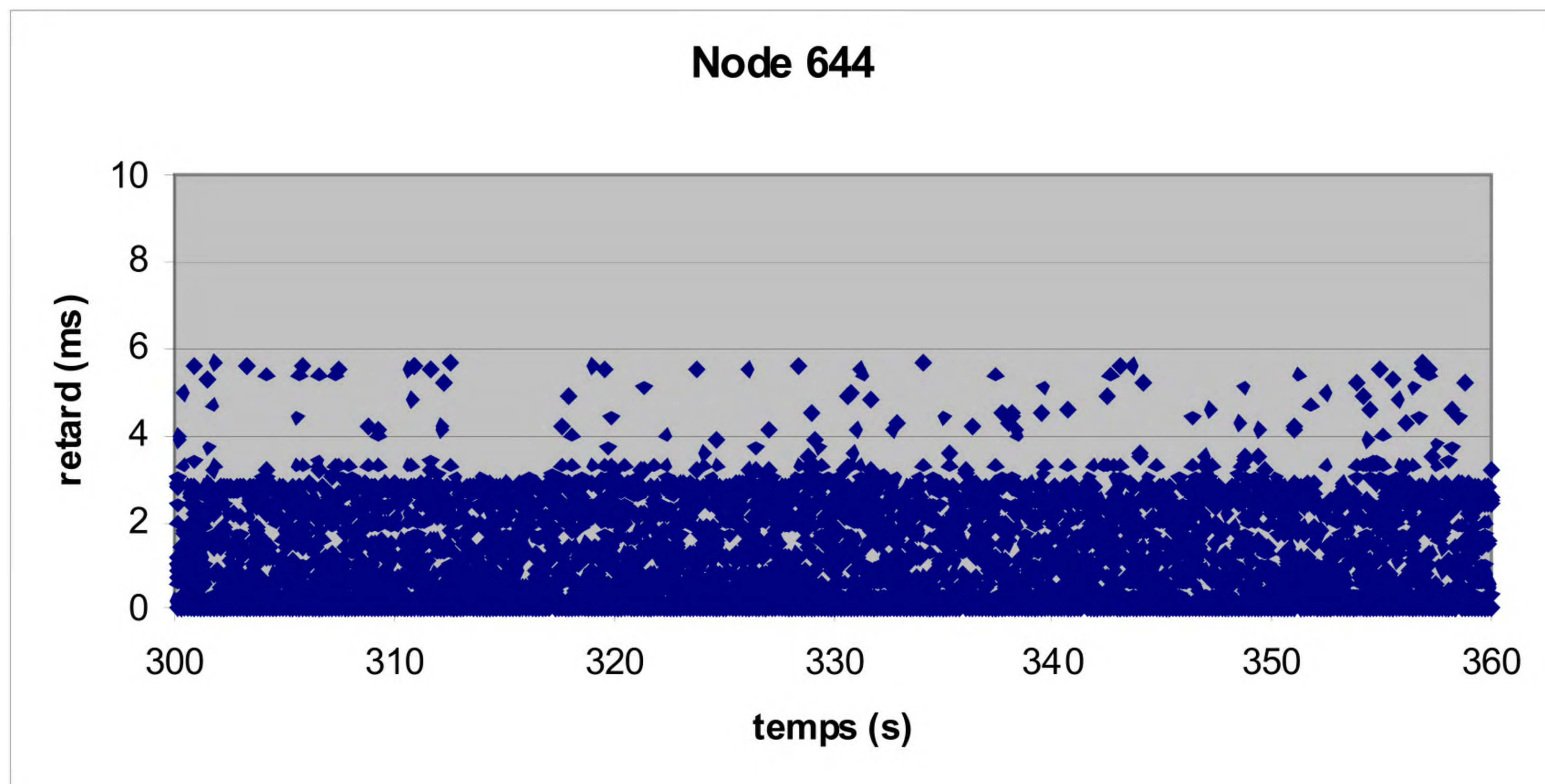
Aquest node és el de més prioritat al bus i segons les simulacions es d'esperar que mai tingui un retard superior a 5.8ms (bàsicament només pot ser degut al temps de bloqueig d'un missatge de més baixa prioritat). En les mesures dutes a terme observem que la teoria s'acompleix i com era d'esperar d'un bus de alta càrrega s'arriba sovint a la situació on el node 170 troba el bus ocupat per un missatge de més baixa prioritat. La següent gràfica mostra tots els retards observats en el període d'estudi.



A la gràfica podem veure com, en cap cas, el temps de retard mai sobrepassa els 5.8ms que el model i el jCAN1 han predit com a pitjor cas possible.

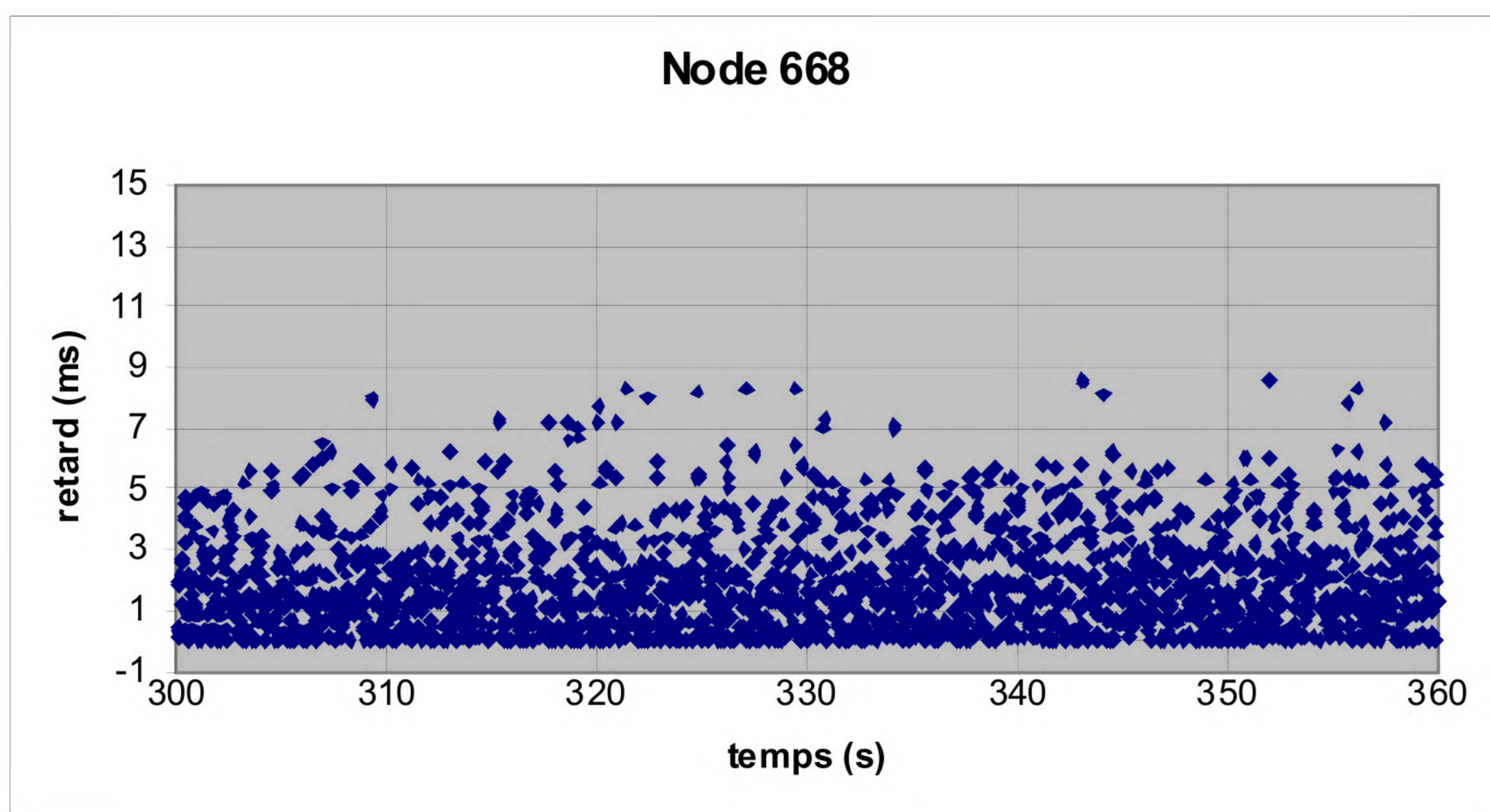
Node 644

Aquest node és el segon en prioritat i només tindrà retard quan un missatge de més baixa prioritat ja estigui al bus o quan el node 170 li guanyi el bus per tenir aquesta prioritat més alta. Segons el jCAN1 en els pitjors dels casos això podria arribar a provocar un retard màxim de 9.08ms.



Altres cops podem observar com és veritat que no es donen casos pitjors que els predits amb el jCAN1.

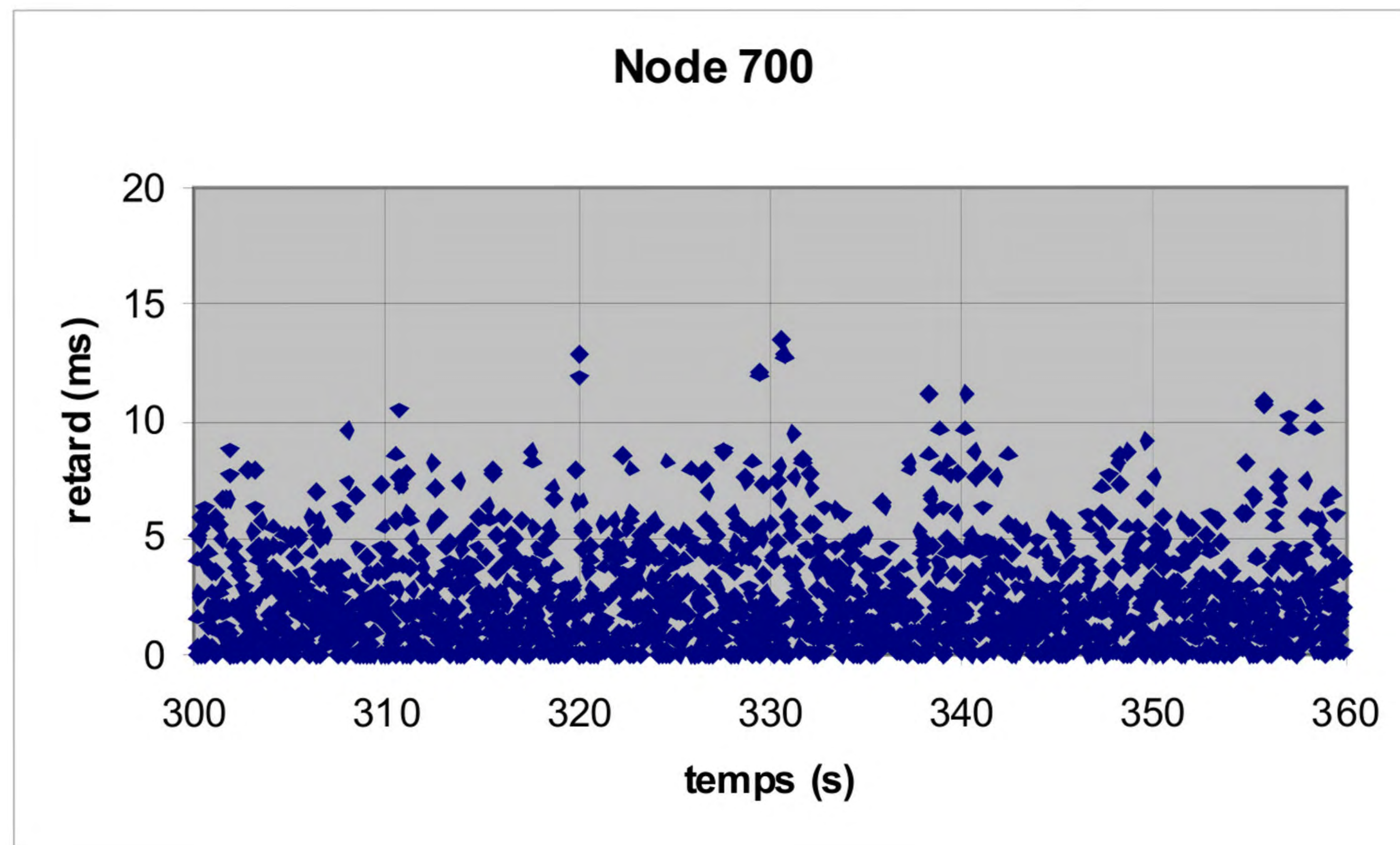
Node 668



Segons la simulació duta a càrrec amb el jCAN1 no hauríem de trobar en cap cas un retard superior als 12.36ms

Node 700

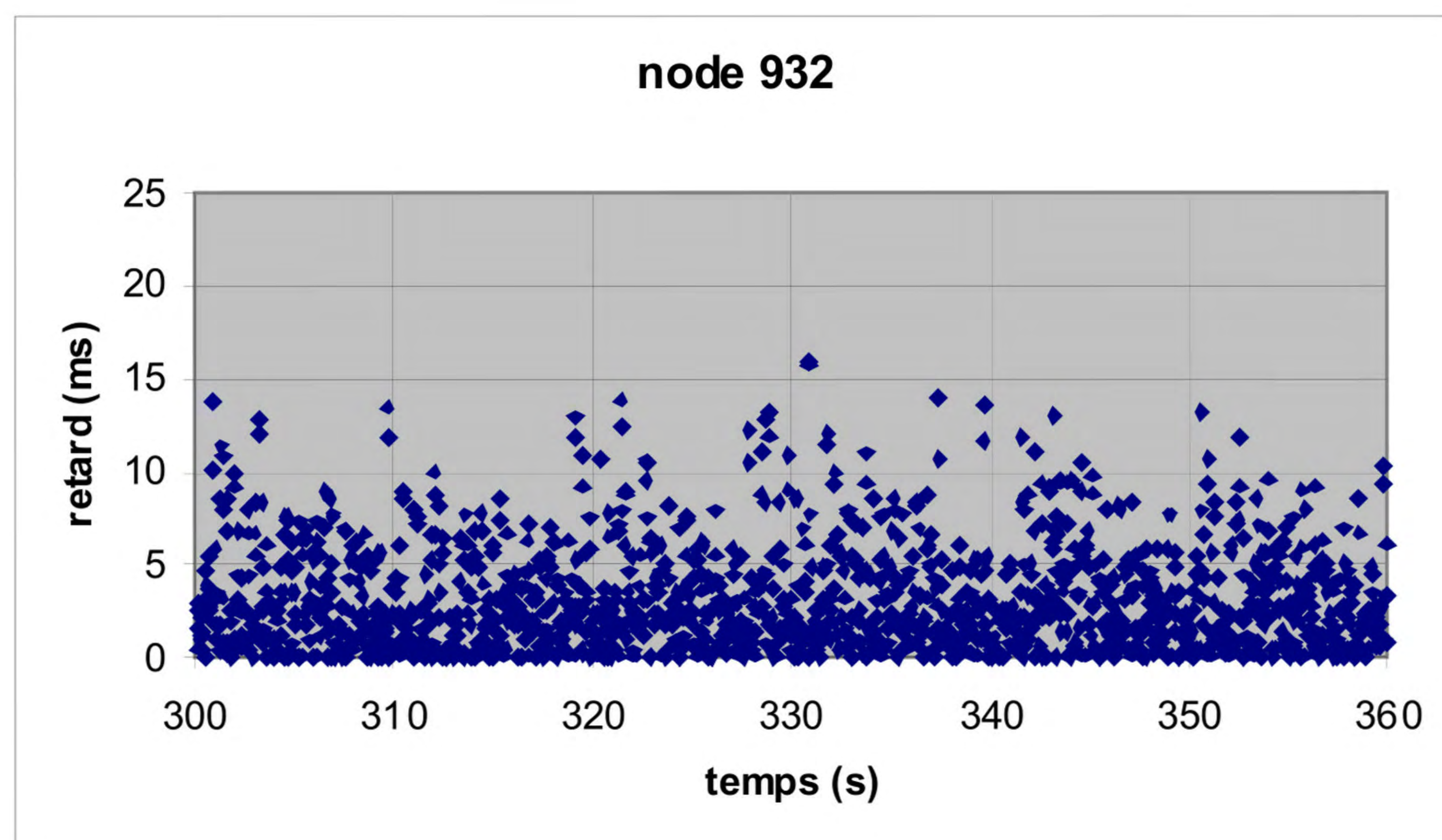
Aquí el temps màxim de retard és de 18.92ms.



El valor observat continua estant dintre del marge previst al jCAN1.

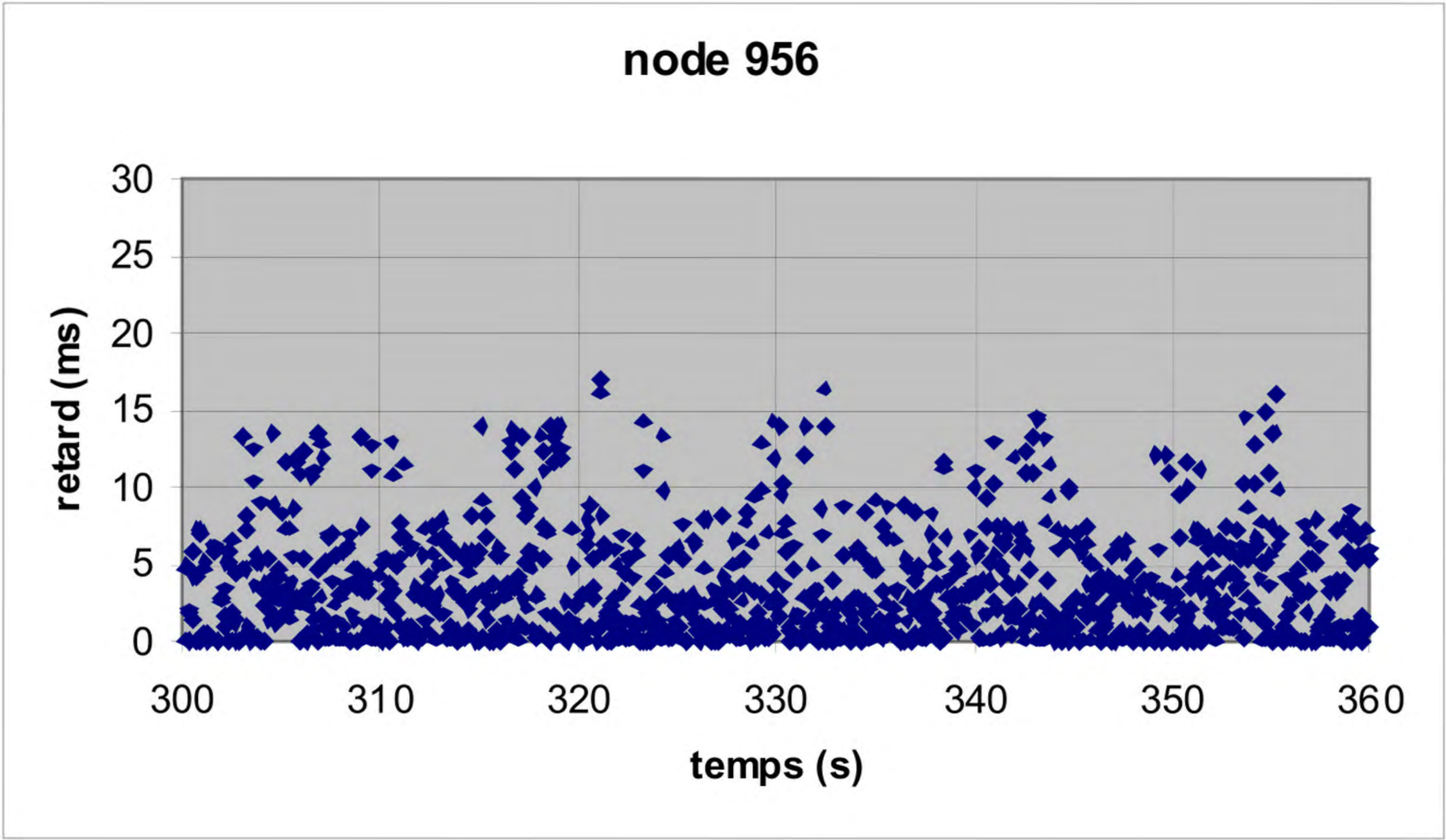
Node 932

Temps de 25.48ms



Node 956

Temps (previst pel jCAN1 com a màxim retard possible en aquest node) de 38.60ms

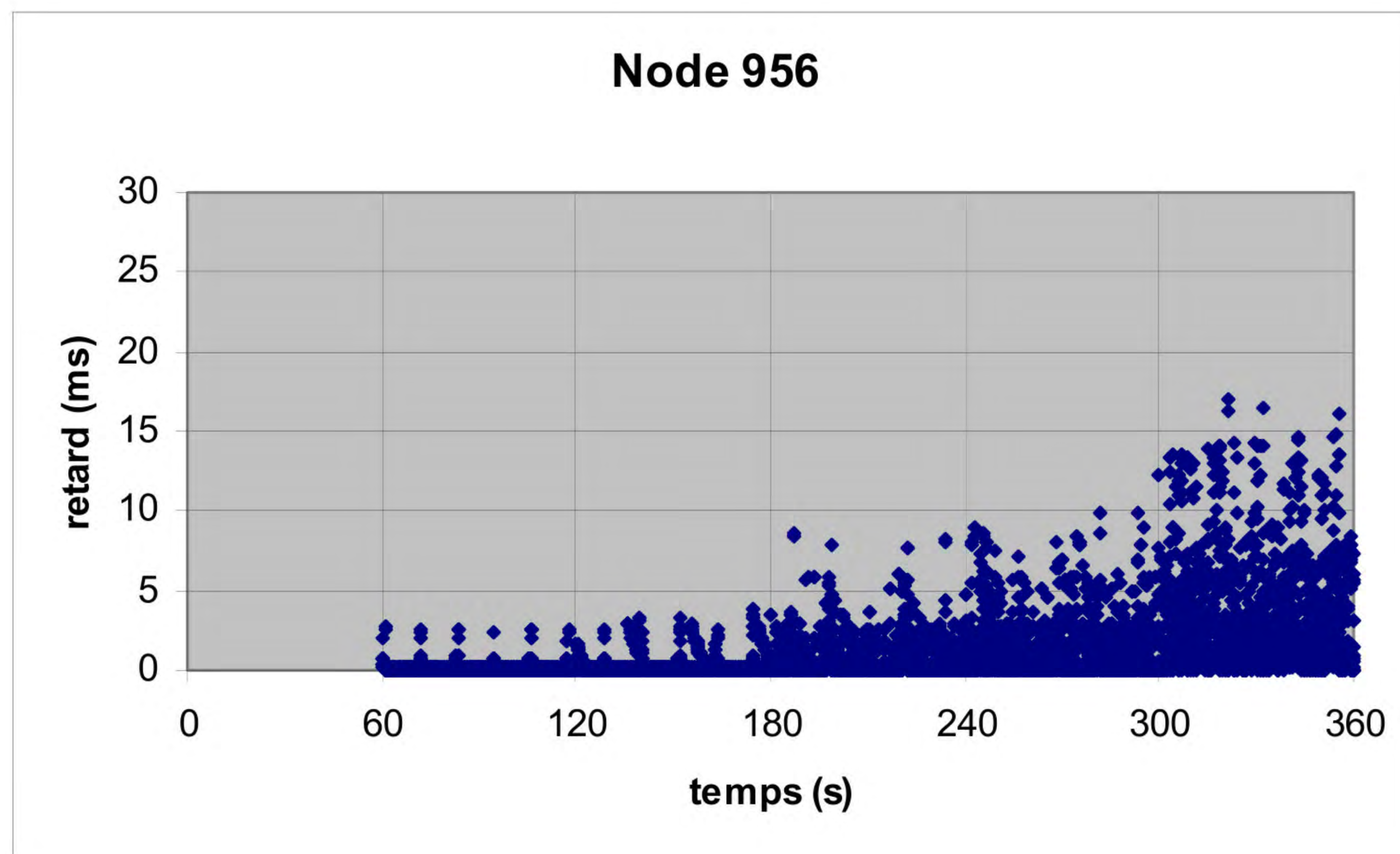


En aquest últim node, el que té menys prioritat, es veu que el cas de més retard és més petit que el màxim previst pel jCAN1. Aquesta distància és més gran que en altres casos.

Les mesures fins ara vistes estan compreses entre el segon 300 i el 360. Això és degut a que les mesures formen part d'un prova de laboratori més amplia. En aquesta prova es busca veure com els temps de retard realment tenen a veure amb la càrrega del bus i el número de missatges que aquest ha de suportar.

Etapa	Temps	Nodes actius
1a	0s-60s	170
2a	60s-120s	170, 956,
3a	120s-180s	170, 956, 932,
4a	180s-240s	170, 956, 932, 700
5a	240s-300s	170, 956, 932, 700, 668
6a	300s-360s	170, 956, 932, 700, 668, 644

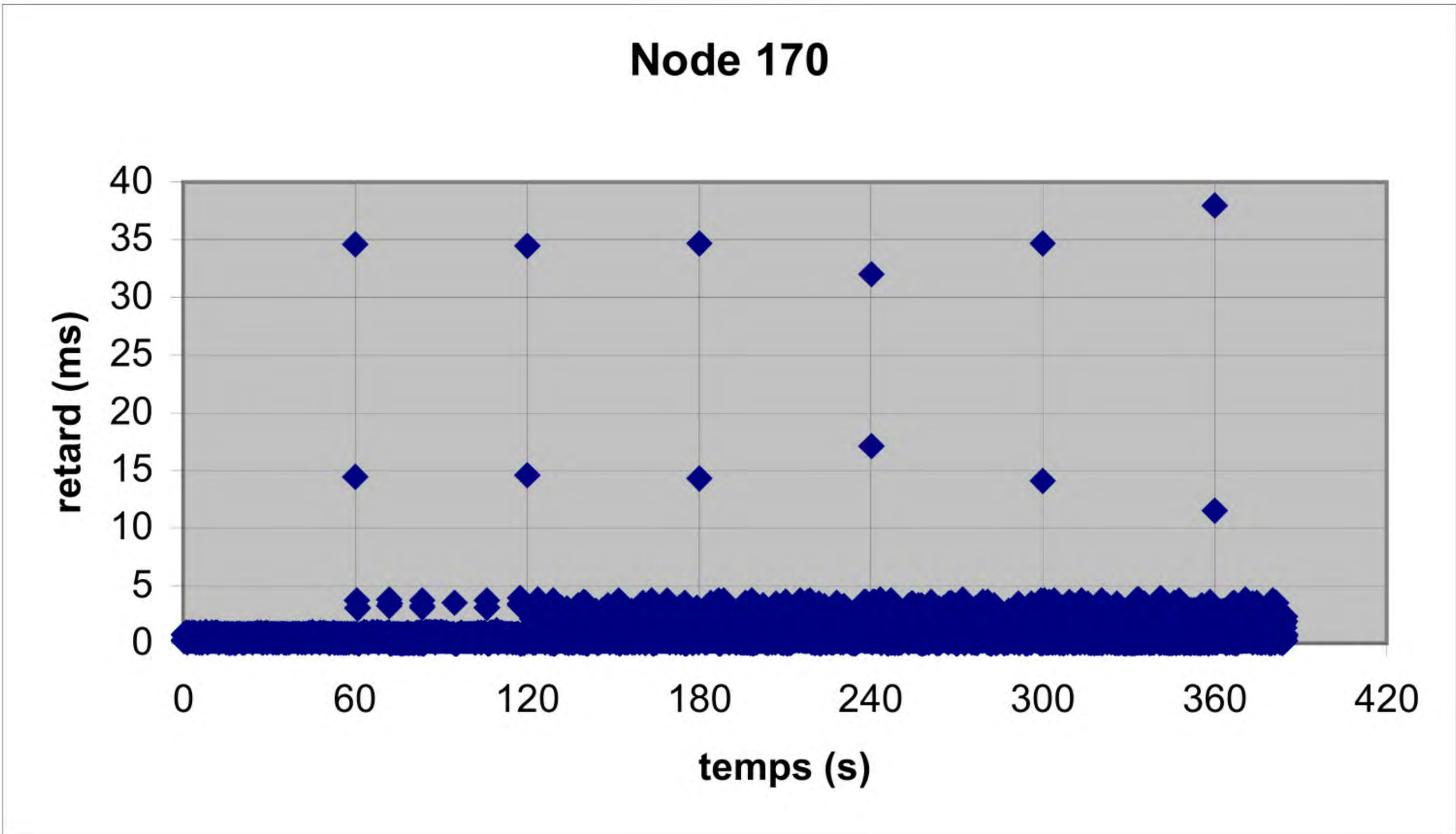
A la pàgina següent hi ha la gràfica on es poden veure les sis etapes sobre les que està basada la prova i on en cada etapa només hi ha actius uns certs nodes, tal com s'explica la taula anterior. És mostra el node amb menys prioritat, perquè és on millor s'observa l'efecte. D'aquesta manera es pot observar de forma clara com afecta el número de missatges que hi ha en cada moment als temps de retard que pateixen els missatges.



4.6.4 ALTRE ALTERNATIU

Un cop vist aquests exemples d'observació del bus, veiem que hem de conèixer i interpretar els resultats d'una forma estadística. El sistema CANalyzer ens permet programar dos nodes CAN i tenir força control sobre aquests nodes. Així un mètode proposat per fer la comprovació passa per fer servir l'altre node CAN per enviar un missatge d'una certa prioritat al bus i veure quins retards té en ser lliurats al bus, en diverses condicions de càrrega. En un cas concret el que s'hauria de fer és fer treballar aquest node amb una certa freqüència i observar-lo en diversos sistema de càrrega i de freqüència com hem fet en els exemple anteriors amb la avantatja que és el mateix CANalyzer que fa la tasca d'enviar el missatge i calcular el retard i en temps real podem processar les dades obtingudes i fer una estadística més immediata i ràpida que la duta a terme amb els exemples anteriors on les dades eren guardades en fitxers i processades a posteriori amb Excel havent de repetir masses vegades les mesures per destacar fonts d'errors com trames d'error i de sobre càrrega que amb el node del CANalyzer programat en CAPL seria més fàcil obtenir dades.

El problema que s'ha trobat és que el CANalyzer és molt poc exacte alhora de lliurar missatges el bus. Volent dir que té un jitter massa gros i diferent en cada cas que fa inviable tenir en compte els resultats obtinguts. Veiem un exemple d'un node programat amb el CANalyzer i que ha de lliurar un missatge de forma periòdica introduït en un sistema equivalent als anteriors:



En aquest exemple s'està treballant amb un bus a 25Kbits/s i amb els següents missatges de la taula següent:

Node	ID	Freqüència	Període	Deadline
CAN1	170	20.12	49.7	49.7
SLIO-1	645	107.53	9.3	9.3
SLIO-2	668	33.33	30.0	30.0
SLIO-3	700	42.19	23.7	23.7
SLIO-4	932	29.13	39.8	39.8
SLIO-5	956	20.01	49.98	49.98

A la gràfica anterior es pot observar com hi ha moments puntuals on el retard sobrepassa immensament els previstos amb el jCAN1. I al ser aquest node el de més prioritat del bus és inexplicable; ja que mai hauria d'estar més temps que el temps de bloqueig esperant ser lliurat al bus; i aquest temps de bloqueig no és de més de 5ms. I els punts aquests estan al voltant de 15ms. El fet més clar és que aquestes singularitats estan en posicions fixes de cada 60 segons. La conclusió que en traiem és que quan el CANalyzer té dos tasques afer en el mateix instant de temps pot arribar a retardar una de les tasques, sense tenir en compte que la tasca que pot arribar a retardar és de més prioritat. Això segurament es degut a la combinació de dos factors:

- Els Controladors CAN que incorpora la targeta PC-ISA del CANalyzer són els integrats Philips PC82C200 que ja de per si tenen aquest problema. Què al tenir diversos treballs en cua és incapaç de llegir-ne les prioritats de tots aquest treballs i lliurar-ne el que tingui més prioritat al bus. A l'apartat de descripció del model teòric ja s'ha explicat que aquest model no té en compte directament aquest fenomen que només passa en controladors CAN antics com aquest. La forma d'incloure aquest fenomen al model és posant aquest temps mort a la cua del controlador PC82C200 com a temps de jitter. Volent dir amb això que el model i el programa si que permeten simular

situacions on aquests dispositius existeixin però per fer les comparacions introdueix massa marge d'error i interessa fer l'estudi del model en situacions més límit.

- L'altre possible origen d'aquests retards pot estar en el fet que tots aquest esdeveniments venen donats per un programa en C (CAPL) que genera els esdeveniments. Queda pales que les pròpies limitacions del PC a realitzar dos tasques a l'hora introdueixen sempre uns certs retards que ara són visibles quan dos tasques s'activen en el mateix instant de temps. En un cas real d'utilització del jCAN1 això es veuria inclòs en el camp de jitter.

D'aquest dos problemes sorgeix el dubte de si el CANalyzer és capaç de monitoritzar i emmagatzemar correctament en el temps tot el que passa en el bus i que per nosaltres és vital.

A les diverses mesures dutes a terme s'ha observat que no té un error tan gran. Aquesta qüestió també ha estat preguntada al suport del CANalyzer de la empresa Vector-Informatik: Com de precís és el temps que ens dóna el CANalyzer per les dades d'entrada? I com d'acurat és el CANalyzer alhora d'enviar missatges al bus?

Com de precís és el temps que ens dóna el CANalyzer per les dades d'entrada?

Les marques de temps que escriu el CANalyzer als fitxer o a la finestra de traçat tenen una resolució de 10microsegons.

I com d'acurat és el CANalyzer alhora d'enviar missatges al bus?

Al parlar de precisió dels missatges enviats hem de parlar de dos casos:

Per l'enviament d'un missatge individual: En aquest cas la precisió és de +/- 2.0ms.

En el cas de més d'un missatge s'ha de dir que al final el període serà la mitja i serà tan acurat com el rellotge del PC, i cada missatge tindrà una desviació de +/- 2.0ms sobre la mitja.

De totes maneres aquesta segona resposta no acabar de ser del tot encertada, com ja s'ha vist en aquest projecte, es pot veure que aquest error pot ser superior per culpa de la manera de treballar del CANalyzer i el seu controlador (mirar explicacions al capítol 4.6.4)

El que si queda clar és que per una mesura d'uns temps de retard d'entre 5ms a 50ms un error de 10 microsegons és molt petit, cas dels nodes SLIO. I en el cas del node CAN1, que treballa entre 15ms i 50ms un error de 2.ms és superior al 10%. Per tan els resultats de latència en missatges de baixa prioritat (cas CAN1) encara són més acurats.

En el cas dels nodes CAN, també s'hauria de considerar un error proper al 10% de l'acumulació de l'error del circuit 555 i, sobretot, del SLIO i el Transceiver.

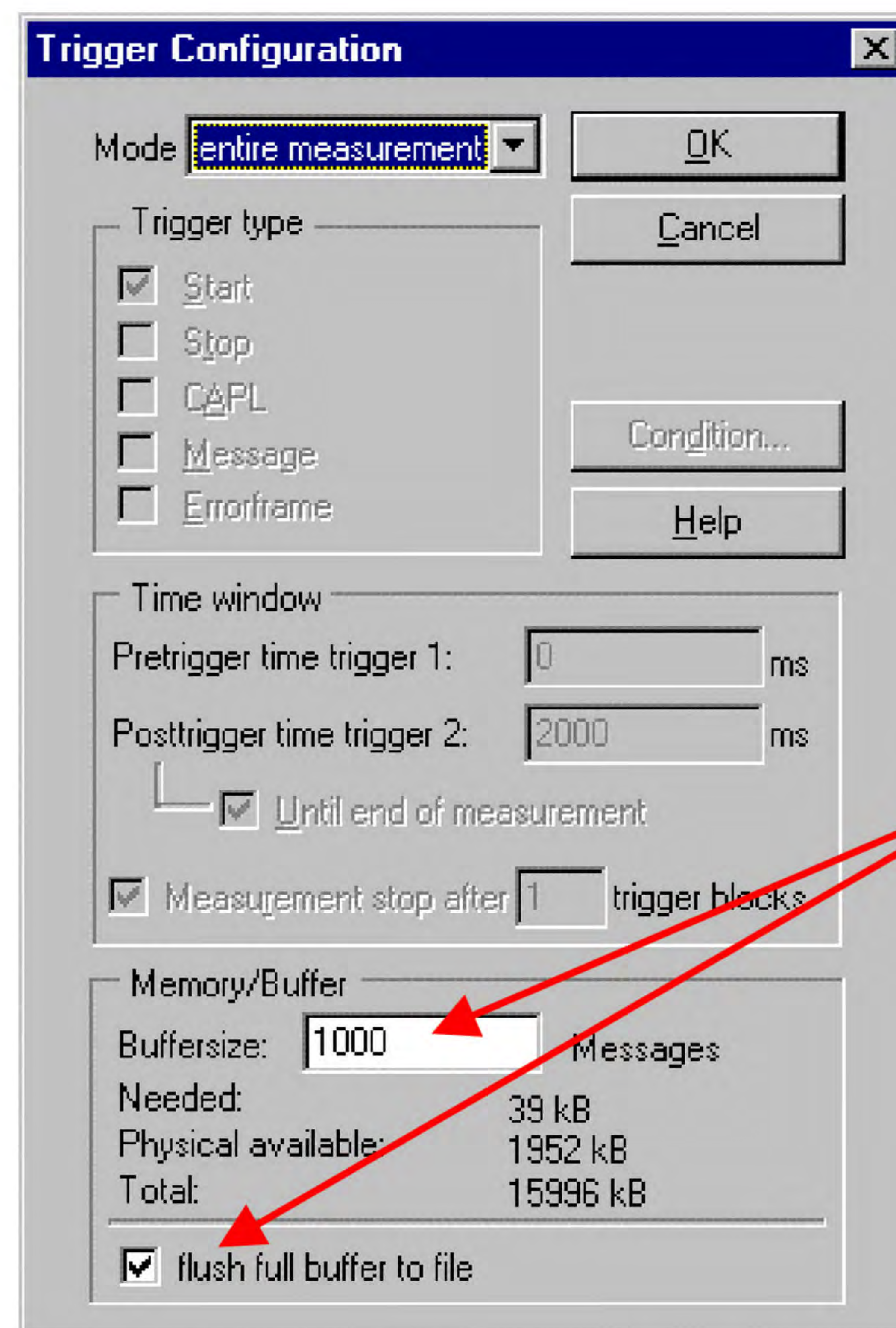
5 CONCLUSIONS

Un cop es té el model matemàtic, el programa de simulació, el entorn hardware i software de laboratori i es disposa de dades tractades ja es poden treure conclusions. Les conclusions es basen en els resultats obtinguts al laboratori respecte als resultats retornats pel algorisme de càlcul de la programabilitat. En les primeres mesures s'ha vist que hi ha una diferència apreciable entre els resultats teòrics i pràctics, sobretot pels missatges amb menys prioritat. Ara és el moment de quantificar aquesta diferència i mirar de trobar-ne alguna explicació. Per fer-ho cal assegurar-nos que les dades obtingudes són el màxim de fidedignes. Per això primer s'explicarà les millores dutes a terme en el sistema d'adquisició de dades.

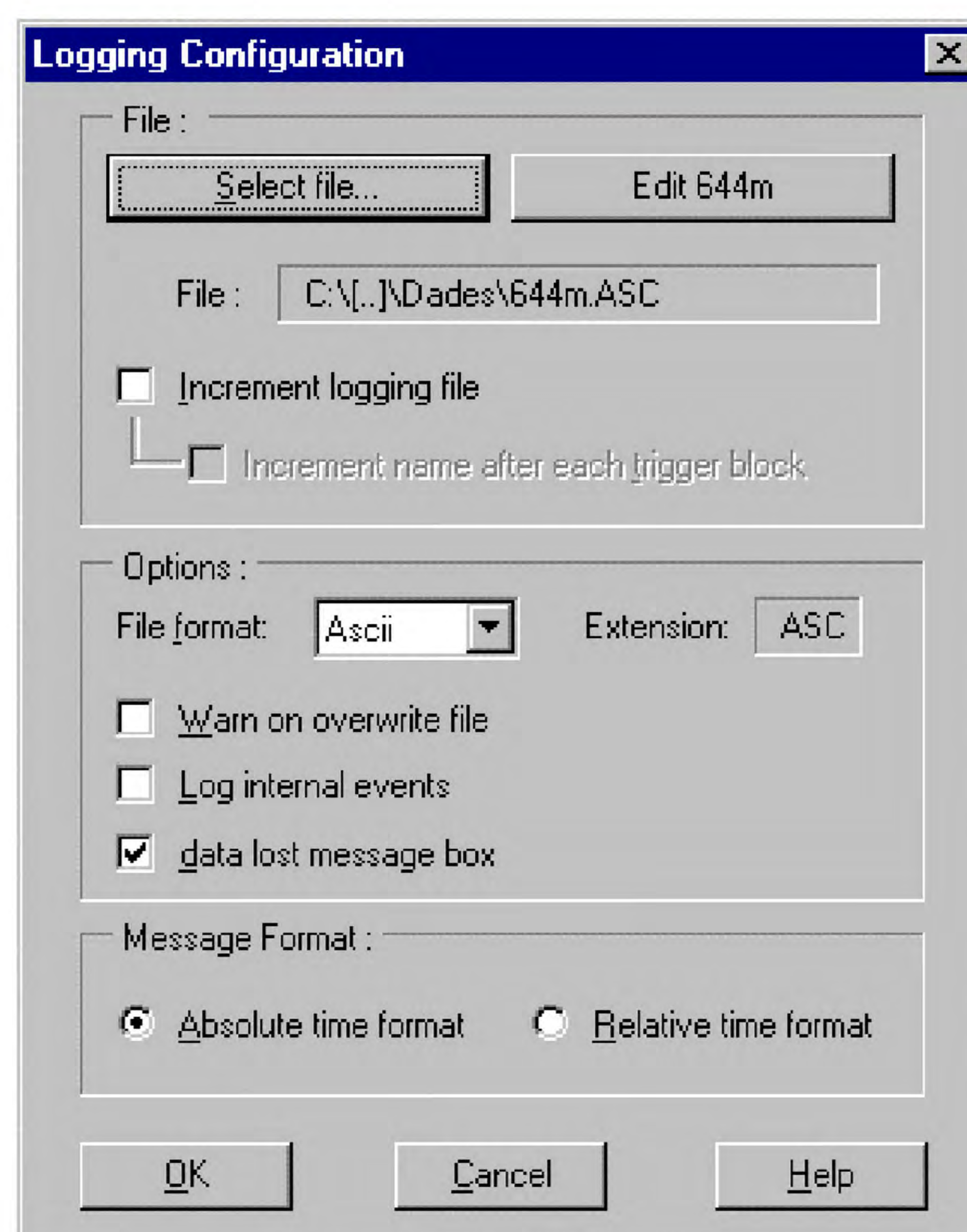
Millorar l'adquisició de dades

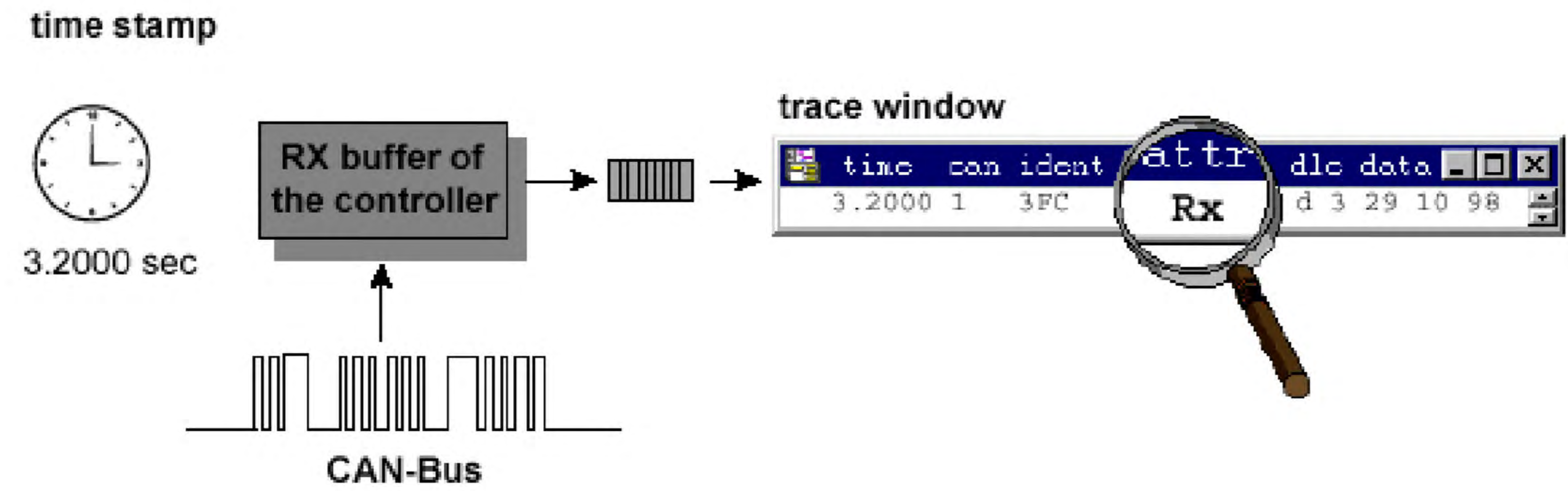
Alhora de fer les mesures s'ha de maximitzar la mostra en el temps per poder observar el màxim número de casos de latència. Per fer això es topa amb dues limitacions de l'eina d'adquisició de dades i de l'eina utilitzada pel tractament d'aquestes dades:

- El CANalyzer dona moltes errades en mesures llargues degut a l'ocupació dels buffers on s'escriu. Cal tenir present que s'envien molts missatges al bus i que cada missatge ocupa un 50bytes a la memòria de l'ordinador. Amb aquesta càrrega es donen casos on el programa i el PC no tenen capacitat d'emmagatzemar tots els missatges que es produeixen al bus i es perden de l'estudi, sent aquest mesures invàlides per al seu estudi. Per minimitzar aquest efecte s'han pres un seguit de mesures:
 - Configurar l'escriptura de missatges en acumulacions de 1000 missatges, s'ha trobat que és el valor més òptim, millor compromís entre carregar la memòria i reduir els accessos a disc dur. Amb aquesta opció s'escriu al disc dur en blocs de cada 1000 missatges.
 - Treballar amb bussos de molt baixa freqüència, alhora això dona molta més fiabilitat al bus.
 - Tancar totes les aplicacions de l'ordinador, i també aturar els processos en background.
 - Deshabilitar tots els blocs possibles de la configuració del CANalyzer, deixa només els d'escriptura al disc i amb filtratges.



- Evitar qualsevol interrupció al PC durant les mesures.
- Descartar totes les mesures que tinguin pèrdues de dades. Per mirar-ho això s'activa l'opció Data Lost Message Box:





Un cop millorades les mesures, es procedeix a quantificar la diferència entre el model matemàtic i els valors de laboratori. Això s’ha fet amb uns exemples finals que són representatius d’una mostra ampla de mesures (al CD que acompanya el projecte es troben aquestes mesures). S’han buscat els exemples més crítics, que són aquells on els resultats són més divergents com ja s’ha vist als capítols anteriors, en els exemples d’adquisició i tractaments de dades.

Seguint les indicacions d’adquisició de dades i fent servir el sistema d’anàlisi ja descrit, s’ha obtingut uns valors arrodonits de la diferència entre el jCAN1 i l’entorn de laboratori. És important dir que s’han fet múltiples mesures de llargades mitges de 1.000 segons per tal de trobar el màxim de punts crítics de latència. Però sempre es difícil assignar un valor absolut amb la màxima precisió, perquè, com ja s’ha dit, hi ha l’error de precisió del CANalyzer (que s’ha pogut observar en les sèries de mesures realitzades). És per això que a la taula inferior és donen els valors mitjos arrodonits. Amb tot obtenim per al problema estudiat (que és el millor sistema per a l’adquisició de dades i observació d’alta càrrega) una taula que ens dona una diferència del voltant del 30%.

Node	Id prioritat	Latència (ms)		Diferència
		jCAN1	Laboratori	
CAN 1	170	5800	≈4000	≈31%
Node 1	644	9080	≈6000	≈33%
Node 2	668	12360	≈9000	≈27%
Node 3	700	18920	≈14000	≈26%
Node 4	932	25480	≈17000	≈33%
Node 5	956	38600	≈22000	≈43%
Node 6	---	---	---	---

Al llarg del projecte ja s’ha comentat que aquest gap era previsible en el fet que el model és molt estricte alhora de buscar el pitjor cas, ja que té en compte situacions molt poc provables. Però, tot i així, aquestes situacions poc provables són possibles i cal tenir-les en compte. Una opció de futur fóra fer petites correccions de probabilitat al model, però cal tenir present que aquest gap pot servir com una distància de seguretat. L’important és saber-ne l’origen de la diferència per saber si aquest gap és problema del model matemàtic o de l’entorn de laboratori creat.

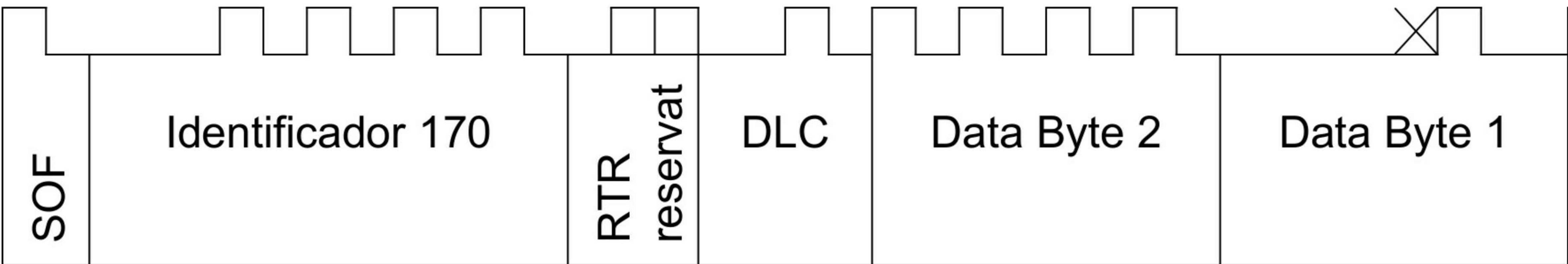
Un dels principals motius que s’ha trobat és degut a que el càlcul del model es fa tenint en compte els Pitjors casos per stuff bits.

Quan es fa el càlcul del temps que un missatge ocupa el bus es fa servir la següent formula:

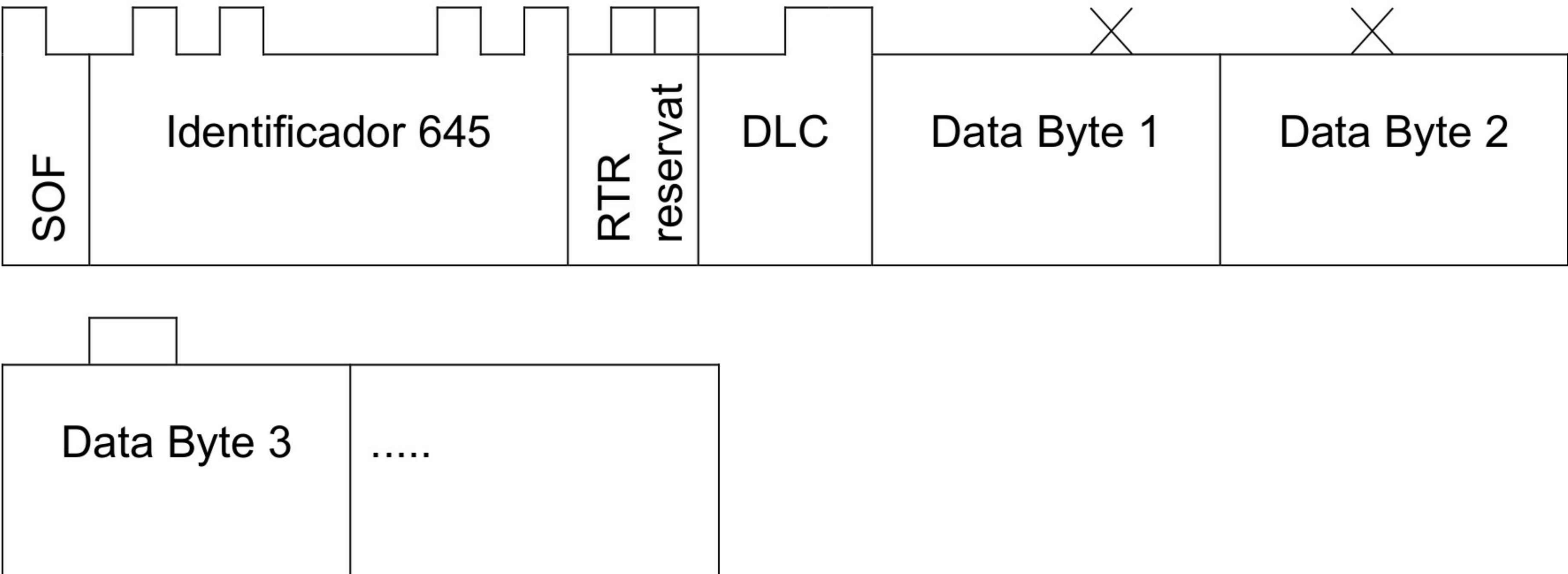
$$C_m = (stuff\ bits + bits\ fixes + bits\ dades) \cdot \tau_{bit} = \left(\left\lceil \frac{34 + 8S_m}{5} \right\rceil + 47 + 8S_m \right) \cdot \tau_{bit}$$

Com es pot veure aquesta fórmula inclou el càlcul dels bits de stuff, aquests bits ja s’han explicat en la descripció del bus CAN. En aquest càlcul es té en compte el pitjor cas, aquest cas correspon a una trama plena de bits d’igual nivell, si més no cada 6 bits seguits (dominants o recessius). Però, en el nostre laboratori aquest cas no s’ha donat. El primer fet és que justament el missatge més prioritari està format per una combinació de bits feta a propòsit per tenir el màxim d’estona una alternança de bit dominant i de recessiu.

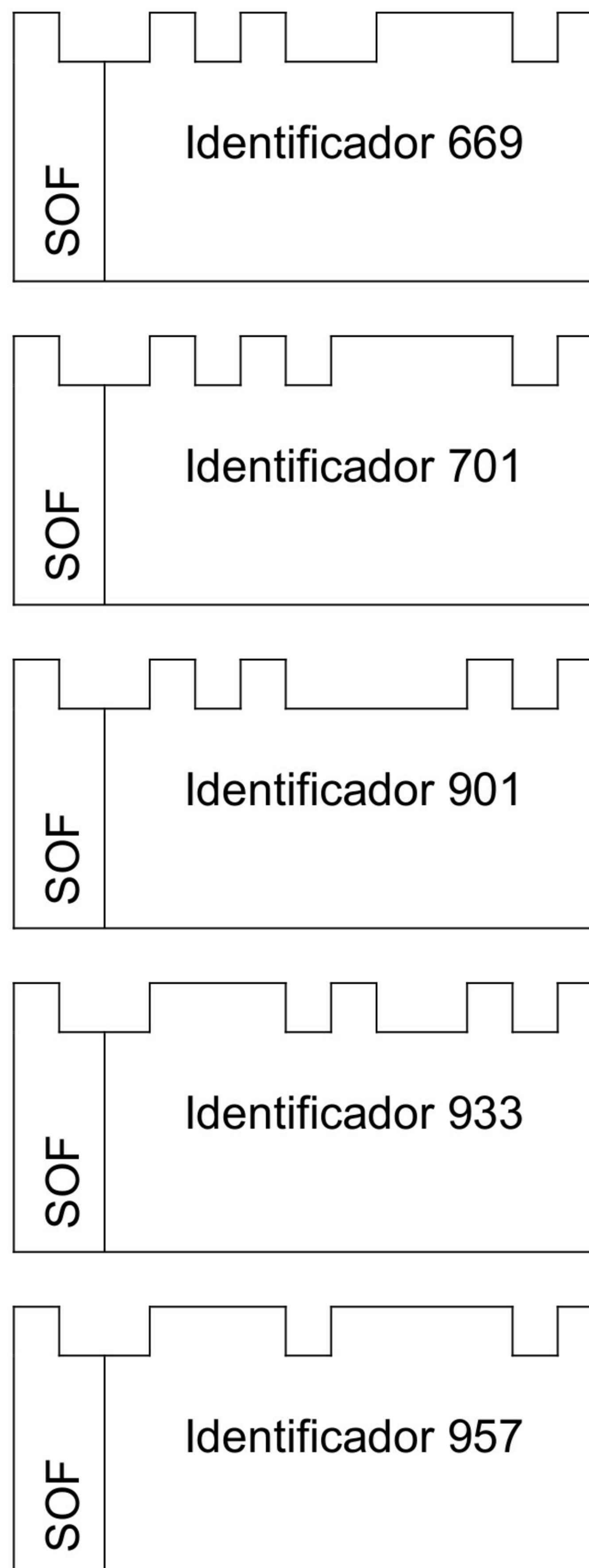
A la figura inferior s’observa que la trama del missatge de calibració només consta d’un sol bit de stuff. Depenent dels valors dels bits de CRC i dels reservats es podria arribar al cas de tenir un total de quatre bits de stuff. (Amb X es marca el punt on hauria d’anar un bit de stuff)



En les trames dels missatges dels nodes SLIO només trobem dos bits de stuff, que poden arribar a ser quatre o cinc amb el còmput dels bits de stuff de la trama de CRC. La figura següent mostra la trama de bits del node SLIO 1 (id=645):



Per la resta de nodes ens trobaríem que la única diferència està en el cap d’identificador de 11 bits i que cap d’ells sobrepassa mai els 5 bits seguits del mateix nivell i no estan sotmesos a cap bit de stuff:



El fet de tenir una trama de calibració amb molts canvis de nivell, de recessiu a dominat i viceversa, és degut a que aquesta trama està pensada perquè els nodes SLIO agafin la velocitat del bus d'aquest missatge, i és lògic que sigui el màxim de canviant. També ens passa una cosa semblant en els missatges enviats pels nodes SLIO, on degut al disseny fet, i als identificadors que poden agafar, tampoc es dona cap cas de tants bits dominats seguits com perquè succeeixi una inserció de bit per la tècnica de stuff. En total tenim que el jCAN1 té en compte molts més bits de propagació dels que realment tenim a l'entorn de laboratori. Això no és un problema de del jCAN1, que sempre busca el pitjor cas; però si que ho hem de tenir en compte de cara a comparar resultats.

En l'algorisme s'han comptat sempre 12 bits de stuff, tal com mostra la següent fórmula, per missatges de 3 bytes de longitud:

$$3\text{bytes} \Rightarrow \frac{(34 + 8 \times 3)}{5} = 12\text{bits}$$

i a la pràctica, al nostre entorn de laboratori, es donen uns quatre bits per missatge. Això vol dir que l'algorisme sempre ens conta 8 bits de més en cada un dels missatges que es produeixen al bus. I això representa respecte

a les mides reals de les trames un error de més del 10%. Amb aquest 10%, el resultat són molt més propers entre el pitjor cas de latència observat al laboratori i els que ens retorna el programa jCAN1.

De totes maneres encara resten diferències, segurament degudes al mètode de calcular la latència màxima que té l'algorisme utilitzat amb el jCAN1.

La valoració final és que el model matemàtic utilitzat per trobar els temps de latència dels missatges no és massa precisi, però és útil, i l'eina jCAN1 pot ser de gran ajuda.

Es diu que no és precís perquè els temps de latència màxima retornats pel model són superiors als observats realment. D'altra banda també es diu que és de gran utilitat perquè permet als dissenyadors de sistemes CAN saber per endavant si el seu bus serà programable. Si bé és cert que hi ha un marge d'un 20%, aquest marge és sempre per sobre i mai portarà a creure que un sistema és programable i té una bona càrrega i que alhora d'implementar físicament el sistema ens trobem amb que no és programable o que el dimensionat ha estat erroni. A més a més, aquest marge observat del voltant del 20% pot ser utilitzat com un marge de seguretat. I en aquest sentit cal prestar molta atenció a la dada facilitada per la SAE (norma SAEJ1939), que demana que els bussos CAN es dissenyin amb una càrrega màxima del 30%. Però amb l'ús de l'algorisme que s'ha fet servir, es pot treballar amb tota seguretat amb càrregues del 60%, s'ha triat aquest valor per buscar un compromís entre programabilitat i càrrega; a la pràctica la càrrega màxima recomanada si fem servir el jCAN1 serà variable i vindrà marcada pel factor de qualitat (alfa). Així per valors d'alfa grans es pot forçar al màxim la càrrega del bus.

Un cop fet la valoració del model triat i vistes les seves deficiències i virtuts es passa a valor el projecte en general. Al capítol d'objectius s'ha parlat de dividir el projecte en quatre objectius específics, que ara es passen a valorar per separat:

1. Trobar model matemàtic

El primer objectiu s'ha assolit segons l'esperat. S'ha trobat i desenvolupat un model matemàtic per simular i trobar resposta als problemes de programabilitat del bus CAN; però a la vista dels resultats queda la possibilitat de continuar treballant en la millor d'aquest algorisme utilitzat; ja que amb els resultats ja es veuen possibles modificacions a fer, així com tenir en compte casos obviats per simplificar la modelització

2. Crear programa informàtic

Aquest objectiu s'ha assolit de forma satisfactòria. Un dels resultats visibles d'aquest projecte fi de carrera és l'aplicació jCAN1, que ha demostrat sé fiable i útil en els càlculs del model triat. Si es produeixen millores al model, també, s'hauran de fer de forma paral·lela en jCAN1. Gràcies a la

documentació generada en javadoc, el programa és fàcilment ampliable, i per tercers.

3. Crear entorn hardware de laboratori

L'entorn de laboratori s'ha realitzat segons tots els previstos i ha funcionat perfectament. Només cal destacar que un dels vuit nodes utilitzats ha fallat en la seva fase física de construcció. L'entorn creat pot ser útil per altres finalitats del departament, gràcies a les connexions de expansió que s'han deixat i a la comoditat i estandardització del subrack construït. Tot l'entorn ha funcionat bé amb el CANalyzer. Amb els resultats obtinguts, però, s'ha vist que per un estudi en més detall caldria treballar també amb altres eines, a més a més de l'entorn creat, per tal d'obtenir més resultats per verificar la validesa del jCAN1.

4. Fer simulacions i comparar resultats

Finalment dir que les fases de simulació i comparació de resultats han estat fetes segons el previst tot obtenint resultats esperables, però no prou acurats com s'hagués desitjat. Deixant la possibilitat de continuar innovant en els mètodes d'obtenció i comparació de dades de l'entorn de laboratori.

6 BIBLIOGRAFIA

Articles

K. TINDELL, A. BURNS, "Guaranteeing Message Latencies on Control Area Network (CAN)", YCS 229, Department of Computer Science, University of York (Juny 1994)

S. SAAD-BOUZEFRANE, F. COTTET, "A Performance Analysis of Distributed Hard-Real Time Applications", Computer Science Laboratory, Futuroscope (1997)

K. M. ZUBERI, K. G. SHIN, "Scheduling Messages on Controller Area Network for Real-Time CIM Applications", Department of Electrical Engineering and Computer Science, The University of Michigan (1997)

K. TINDELL, A. BURNS, "Analysis of Hard Real-Time Communications", YCS 229, Department of Computer Science, University of York (Gener 1994)

F. SIMONOT-LION, Y.Q. SONG, J. RAYMOND, "Validating Real-Time Applications distributed over CAN: an interoperability verification", 4th International CAN conference (Novembre 1997)

G. CENA, C. DEMARTINI, A. VALENZANO, "On the Performance of Two Popular Fieldbuses", Dipartimento di Automatica e Informatica, Politecnico di Torino (1997)

K. UWE, J. DIRK, S. SCHNEIDER, "Performance analysis of a distributed automotive real-time system", 4th International CAN conference, November 1997

K. TINDELL, A. BURNS, A. WELLINGS, "Calculating Controller Area Network (CAN) Message Response Times", Control Engineering Practice, vol.3 no.8, pp. 1163-1169 (1995)

K. M. ZUBERI, K. G. SHIN, "Non-Preemptive Scheduling of Messages on Controller Area Network for Real-Time Control Applications", Department of Electrical Engineering and Computer Science, The University of Michigan (1995)

Data Sheets

* Controller 82527 de Intel

* Controller 82C200 de Philips

- * SLIO 82C150 de Philpis
- * CAN controller Interface PCA82C250

Documents Tècnics

CAN Specification Versió 2.0, Robert Bosch GmbH, 1991.

Webs

- * www.nrtt.demon.co.uk/can.htm ➔ Web de la NTRR, Ken Tindell
- * www.hitex.com/can/ ➔ Hitex'Home page on CAN
- * www.uni-magdeburg.de/ ➔ IPE CAN Home page
- * www.kvaser.com/can/ ➔ Kvaser CAN page
- * www.omegas.co.uk/CAN/ ➔ Mike Schofield's CAN pages

- * www.vector-informatik.de ➔ Empresa (CANalyzer,...)
- * www.can-cia.de ➔ Web de la Cia
- * www.warwick.ac.uk/devicenet/indus.htm
- * www.docs.uu.se/~ken/ ➔ Ken Tindell page
- * [ftp.cs.york.ac.uk/pub/realtime/papers/](ftp://cs.york.ac.uk/pub/realtime/papers/)
➔ Articles públics de U. of York
- * www.eecs.umich.edu/~zuberi/research/
➔ Més articles públics
- * www.fokus.gmd.de/step/simulations/
➔ Eines de simulació

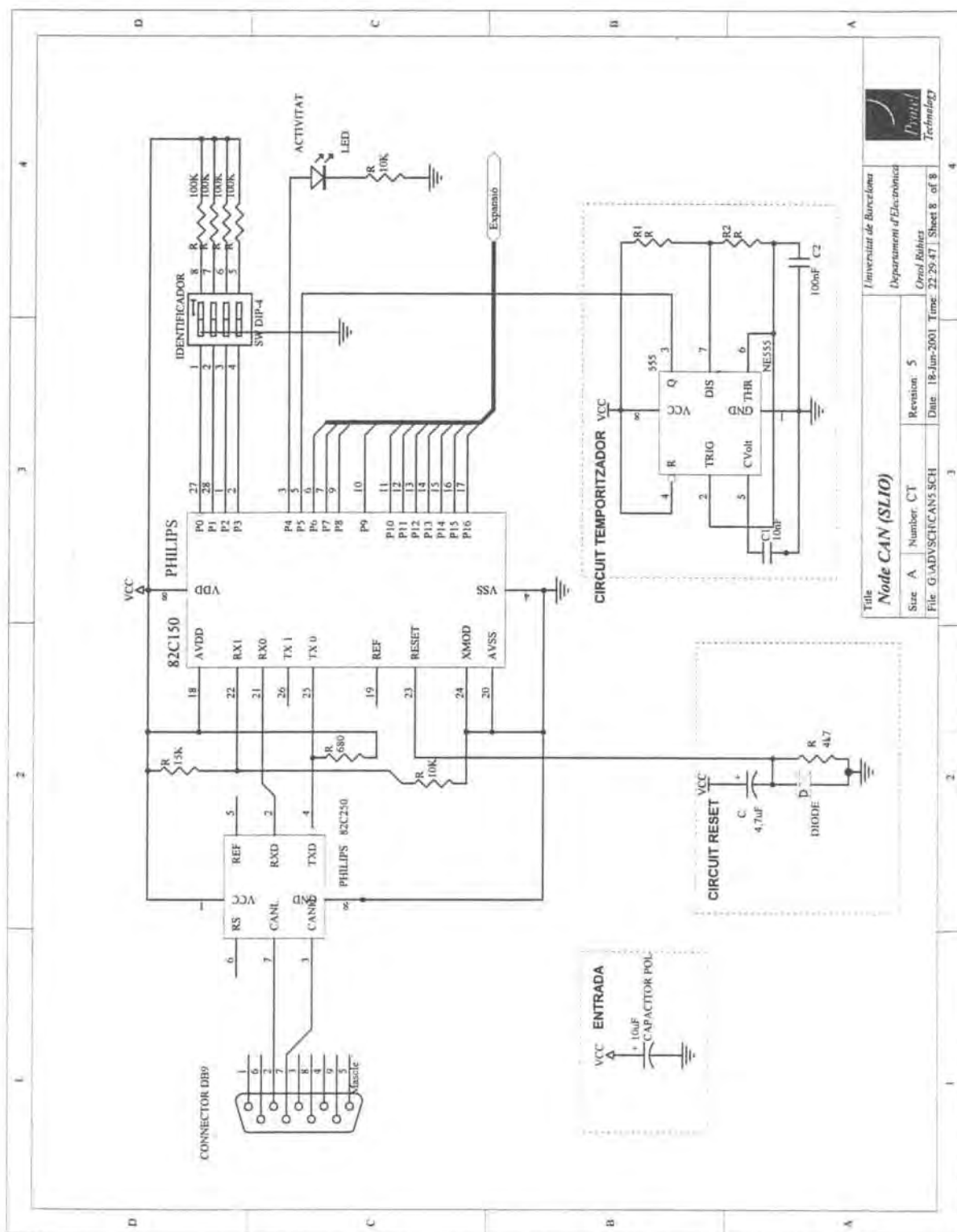
PART II: MEMÒRIA ECONÒMICA

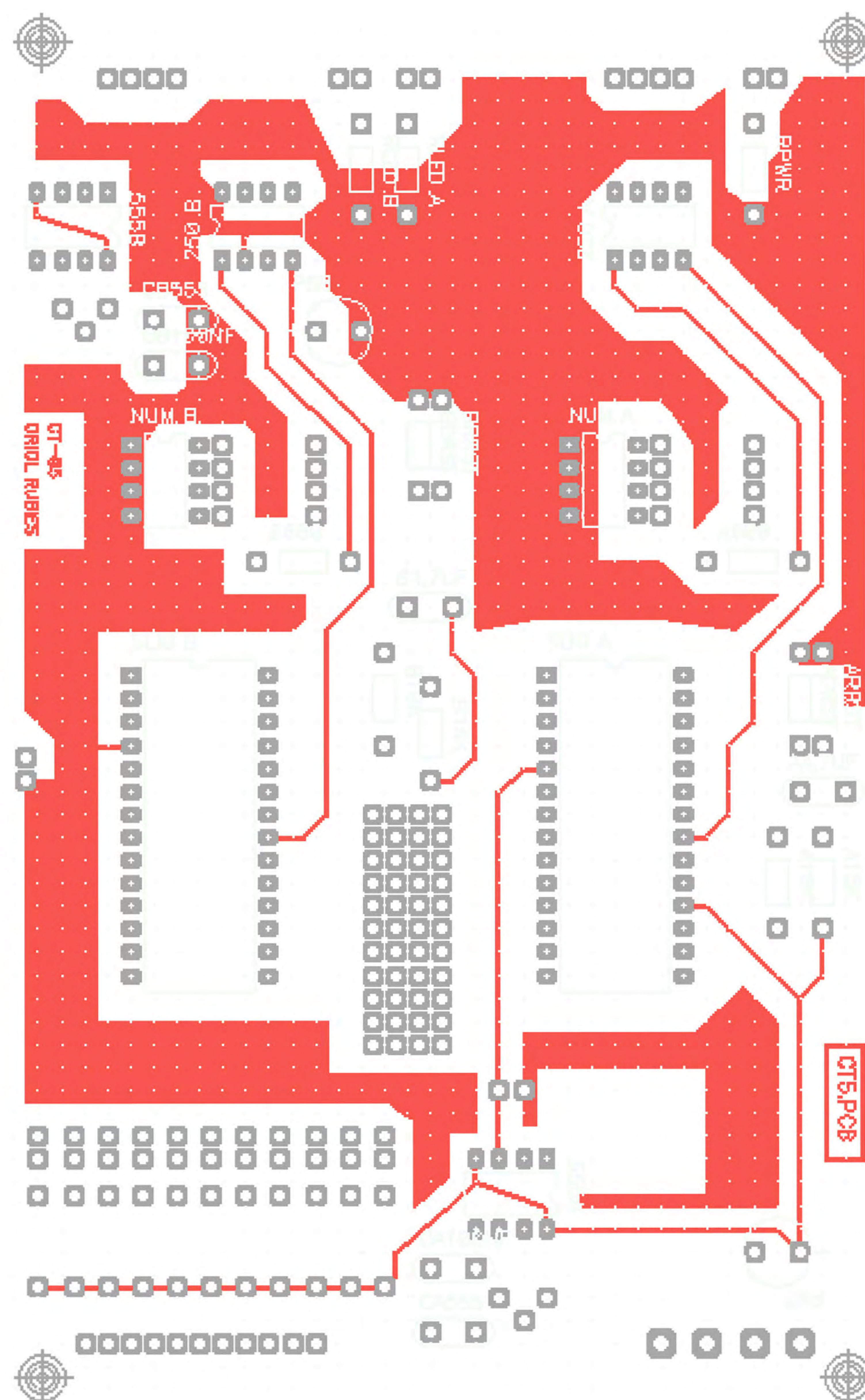
Per avaluar el cost econòmic del projecte només s'ha tingut en compte els materials i components directament i exclusivament adquirits per a la realització del projecte. Ja que la resta de material ja estava disponible al departament d'electrònica, com ara el PC, el CANalyzer, la font d'alimentació, multimetre,... Molt d'aquest material ha estat facilitat directament pel taller del departament i els preus aquí posats han estat extrets de catàlegs o aproximats.

Quantitat	Descripció	Preu unitat	Preu
RACK			
1	Subrack 84E 3U F160 IND (32201312)	4492	4.492 ptes
1	24 Guiacartes de 160 de PC injectat. (32203160)	1791	1.791 ptes
2	Tapa inferior / superior de planxa d'alumini de 1mm (32201716)	1427	2.854 ptes
1	2 Anses d'alumini de 3U/HE(32203003)	761	761 ptes
8	Panels portacartes de (32204306)	609	4.872 ptes
2	Panels portacartes " de 3U, 4E(32204304)	603	1.206 ptes
1	Panels portacartes " de 3U, 14E(32204314)	763	763 ptes
1	Mòdul voluminós 3U, 14E(32205114)	2647	2.647 ptes
2	Perfils Z conversió DIN 41612(22203004)	590	1.180 ptes
MÒDULS CAN			
10	P82C250 "Tranceiver"	490	4.900 ptes
10	P82C150 "SLIO"	1702	17.020 ptes
10	ICM7555PA "555"	127	1.270 ptes
126	Resistències 100KOhms	16	2.016 ptes
12	Resistència 10K	16	192 ptes
6	Resistència 15K	16	96 ptes
6	Resistència 680Ohms	16	96 ptes
24	Capacitats	21	504 ptes
6	Resistència variable	35	210 ptes
6	Selectors DIP8	89	534 ptes
6	Zocals DIP 28	115	690 ptes
18	Zocals DIP 8	30	540 ptes
9	Leds colors	30	270 ptes
12	Diode	9	108 ptes
6	Connectors DB9	105	630 ptes
20	Tires de pins	35	700 ptes
REALITZACIÓ PLAQUES PCB NO METALITZADES			
1	3 + 1 plaques EuroCard + 7 plaques conversor DIP 28-SOP 28	30000	20.000 ptes
		TOTAL	70.342 ptes
		IVA +16%	81.597 ptes

PART III: PLÀNOLS

Schematic



Layout superior



**TREBALL FI DE CARRERA
D'ENGINYERIA EN ELECTRÒNICA**

DEPARTAMENT ELECTRÒNICA

TÍTOL:

PROGRAMABILITAT DEL BUS CAN

AUTOR:

ORIOL RÚBIES

DIRIGIT PER:

DR. JOSEP SAMITIER

Barcelona, 29 de juny de 2001

BIBLIOTECA DE LA UNIVERSITAT DE BARCELONA



0700560385

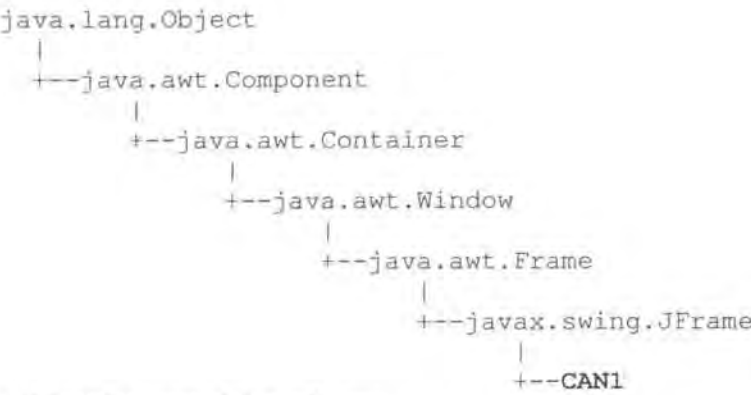


ANNEX I: jCAN1

Package

Class Summary	
<u>CAN1</u>	Genera una interfície gràfica d'usuari amb multiples opcions(*).
<u>FiltreFitxersCAN</u>	Filtre per fitxers de tipus CAN.
<u>GrupMissatges</u>	Un sistema basat en un bus CAN ve descrit per un grup de missatges i dades globals del sistema, tots aquests missatges vene en aquesta classe amb un vector d'objectes <code>Missatge</code> .
<u>Missatge</u>	Un Missatge CAN ve caracteritzat per una serie de dades que la classe <code>Missatge</code> conté.
<u>ShapesPrint</u>	
<u>TaulaMissatges</u>	Aquesta classe és un model abstracte de taula per extendre la funcionalitat de la classe <code>JTable</code> .
<u>Utils</u>	Aquesta classe té la llista d'extensions vàildes i té el mètode capaç d'extreure l'extensió d'un fitxer.

Class CAN1



All Implemented Interfaces:
javax.accessibility.Accessible, java.awt.image.ImageObserver,
java.awt.MenuContainer, javax.swing.RootPaneContainer,
java.io.Serializable, javax.swing.WindowConstants

```
public class CAN1
extends javax.swing.JFrame
```

Genera una interfície gràfica d'usuari amb múltiples opcions(*). La finalitat del qual és poder treballar sobre un problema (un sistema bus CAN) el qual es sotmet a l'estudi de programabilitat. Per això CAN1 facilita una complexa GUI amb facilitat de introducció de dades i mostrant els resultats de forma clara. La GUI permet restaurar i desar sistemes CAN, el que a la classe s'anomena problema, que no és més que una classe GrupMissatges.

See Also:
[Serialized Form](#)

Inner classes inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Inner classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Inner classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Inner classes inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Inner classes inherited from class java.awt.Component

java.awt.Component.AccessibleAWTComponent,
java.awt.Component.AWTTreeLock

Field Summary

private javax.swing.JMenuItem	<u>aboutMenuItem</u>
(package private) double	<u>alfaLimit</u> Constant on indiquem el valor d'alfa crític.
private javax.swing.JToolBar	<u>barraEines</u>
private javax.swing.JProgressBar	<u>barraUs</u>
private javax.swing.JButton	<u>botoAjuda</u>
private javax.swing.JButton	<u>botoCalcular</u>
private javax.swing.JButton	<u>botoCalcularEines</u>
private javax.swing.JButton	<u>botoDesar</u>
private javax.swing.JButton	<u>botoNou</u>
private javax.swing.JButton	<u>botoObrir</u>
private javax.swing.JMenuItem	<u>calcularMenuItem</u>
private javax.swing.JTextField	<u>campFrecuencia</u>
private javax.swing.JMenuItem	<u>contentsMenuItem</u>
private javax.swing.JCheckBoxMenuItem	<u>estandardMenuItem1</u>
private javax.swing.JMenuItem	<u>exitMenuItem</u>
private	<u>extensMenuItem2</u>

javax.swing.JCheckBoxMenuItem	
private javax.swing.JMenu	<u>fileMenu</u>
(package private) java.io.File	<u>fitxerActual</u> Prova de Fitxer Actual epe
(package private) boolean	<u>fitxerObert</u> Si en la sessió de treball ja s'ha especificat un fitxer de treball.
private javax.swing.JMenu	<u>helpMenu</u>
private javax.swing.JLabel	<u>jLabel1</u>
private javax.swing.JScrollPane	<u>jScrollPane1</u>
private javax.swing.JSeparator	<u>jSeparator1</u>
private javax.swing.JSeparator	<u>jSeparator2</u>
private javax.swing.JSeparator	<u>jSeparator3</u>
private javax.swing.JSeparator	<u>jSeparator4</u>
private javax.swing.JLabel	<u>labelAlfa</u>
private javax.swing.JLabel	<u>labelNumMissatges</u>
private javax.swing.JLabel	<u>labelNumMissatgesProgramables</u>
private javax.swing.JLabel	<u>labelResultat</u>
private javax.swing.JLabel	<u>labelUs</u>
private javax.swing.JMenuBar	<u>menuBar</u>
private javax.swing.JMenu	<u>modeMenu</u>
private javax.swing.JMenuItem	<u>nouMenuItem</u>
private javax.swing.JMenuItem	<u>openMenuItem</u>
private javax.swing.JPanel	<u>panelConclusions</u>
private javax.swing.JPanel	<u>panelDades</u>

private javax.swing.JPanel	<u>panelEines</u>
private javax.swing.JPanel	<u>panelM</u>
private javax.swing.JTabbedPane	<u>panelPrincipal</u>
private javax.swing.JPanel	<u>panelResultats</u>
(package private) int	<u>percentatgeUsLimit</u> Constant on indiquem el tant per cent de càrrega del bus crític.
(package private) GrupMissatges	<u>problema</u> En aquest objecte es gaurden totes les dades del problema.
private javax.swing.JRadioButton	<u>rbotoEstandard</u>
private javax.swing.JRadioButton	<u>rbotoExtens</u>
private javax.swing.JMenuItem	<u>saveAsMenuItem</u>
private javax.swing.JMenuItem	<u>saveMenuItem</u>
(package private) javax.swing.JTable	<u>taula</u> Taula on s'introdueixen les dades referents als missatge.
(package private) TaulaMissatges	<u>taulaM</u> La taula de missatges fa servir aquest model abstracte de taula.
private javax.swing.JTextField	<u>textAlfa</u>
private javax.swing.JTextArea	<u>textDescripcio</u>
private javax.swing.JLabel	<u>textFormat</u>
private javax.swing.JLabel	<u>textFrequencia</u>
private javax.swing.JMenu	<u>toolsMenu</u>

Fields inherited from class javax.swing.JFrame
accessibleContext, defaultCloseOperation, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame

base, CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, frameSerializedDataVersion, HAND_CURSOR, icon, ICONIFIED, mbManagement, MOVE_CURSOR, N_RESIZE_CURSOR, nameCounter, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, ownedWindows, resizable, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, serialVersionUID, state, SW_RESIZE_CURSOR, TEXT_CURSOR, title, W_RESIZE_CURSOR, WAIT_CURSOR, weakThis

Fields inherited from class java.awt.Window

active, dbg, focusMgr, inputContext, inputContextLock, OPENED, ownedWindowList, warningString, windowListener, windowSerializedDataVersion

Fields inherited from class java.awt.Container

component, containerListener, containerSerializedDataVersion, dispatcher, layoutMgr, listeningBoundsChildren, listeningChildren, maxSize, ncomponents, printing, printingThreads

Fields inherited from class java.awt.Component

actionListenerK, adjustmentListenerK, appContext, background, BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, changeSupport, componentListener, componentListenerK, componentOrientation, componentSerializedDataVersion, containerListenerK, cursor, dropTarget, enabled, eventMask, focusListener, focusListenerK, font, foreground, graphicsConfig, hasFocus, height, hierarchyBoundsListener, hierarchyBoundsListenerK, hierarchyListener, hierarchyListenerK, incRate, inputMethodListener, inputMethodListenerK, isInc, isPacked, itemListenerK, keyListener, keyListenerK, LEFT_ALIGNMENT, locale, LOCK, metrics, minSize, mouseListener, mouseListenerK, mouseMotionListener, mouseMotionListenerK, name, nameExplicitlySet, newEventsOnly, ownedWindowK, parent, peer, peerFont, popups, prefSize, privateKey, RIGHT_ALIGNMENT, textListenerK, TOP_ALIGNMENT, valid, visible, width, windowClosingException, windowListenerK, x, y

Fields inherited from interface javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary
CAN1() PROVACrea un GUI form CAN1 i <u>inicialitza</u> la <i>taula</i>

Method Summary	
private void	<u>aboutMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) Sobre una finestra amb les dades "Sobre..." el jCAN1.
(package private) void	<u>actualitzarGUI()</u> Actualitza els resultats de la la GUI.
(package private) void	<u>actualitzarTaula()</u> Posar les dades dels missatges del problema, GrupMissatges, a la taula taula de Missatges de la GUI.
private void	<u>botoCalcularActionPerformed</u> (java.awt.event.ActionEvent evt) Inici rutina de càlcul del problema, des de botó: posarMissatges(), calcularProblema(), actualitzarTaula(), actualitzarGUI();
private void	<u>botoCalcularEinesActionPerformed</u> (java.awt.event.ActionEvent evt) Inici del procés de càlcul, iniciat des del botó.
private void	<u>botoDesarActionPerformed</u> (java.awt.event.ActionEvent evt) Inici rutina de desar, des de botó.
private void	<u>botoNouActionPerformed</u> (java.awt.event.ActionEvent evt) Es crear un nou problema, iniciat des de botó.
private void	<u>botoObrirActionPerformed</u> (java.awt.event.ActionEvent evt) Inici rutina d'obrir problema, iniciada des de botó.
(package private) void	<u>calcularProblema()</u> Calcular el problema, GrupMissatges.
private void	<u>campFrequenciaActionPerformed</u> (java.awt.event.ActionEvent evt) Canviem el valor de la freqüència del bus problema, des de camp de text.
private void	<u>contentsMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) S'obre una finestra d'ajuda, amb les instruccions bàsiques d'ús

	del jCAN1.
private void	<u>cutMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) Inici rutina de calcular, iniciada des de Menú.
private void	<u>estandardMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) Seleccionar mode estàndard per Menú.
private void	<u>exitForm</u> (java.awt.event.WindowEvent evt) Sortir de la aplicació.
private void	<u>exitMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) Crida a funció sortida , des de Menú.
private void	<u>extensMenuItem2ActionPerformed</u> (java.awt.event.ActionEvent evt) Seleccionar mode extens per Menú.
private void	<u>initComponents</u> () This method is called from within the constructor to initialize the form.
private void	<u>jButton2ActionPerformed</u> (java.awt.event.ActionEvent evt) Obre finestra amb ajuda.
private void	<u>jRadioButton2StateChanged</u> (javax.swing.event.ChangeEvent evt)
private void	<u>jTextField1ActionPerformed</u> (java.awt.event.ActionEvent evt)
static void	<u>main</u> (java.lang.String[] args) Crear una GUI, la dimensiona i mostra per pantalla.
(package private) void	<u>mostrarProblema</u> () Per Debbbug.
(package private) void	<u>mostrarResultats</u> () Actualitza les cel·les de resultats que té la taula de missatges.
private void	<u>nouMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) Es crear un nou problema, iniciat des de menú.
private void	<u>openMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) Inici rutina obrir fitxer problema, des de Menú.
(package private) void	<u>posarMissatges</u> () Crea un objecte GrupMissatges, inicialitza el problema.
private void	<u>radioBotoExtensActionPerformed</u> (java.awt.event.ActionEvent evt)
private void	<u>radioBotoEstandardActionPerformed</u> (java.awt.event.ActionEvent evt)

private void	<u>rbotoEstandardActionPerformed</u> (java.awt.event.ActionEvent evt) Seleccionar mode estàndard per botó de selecció
private void	<u>rbotoExtensActionPerformed</u> (java.awt.event.ActionEvent evt) Seleccionar mode extens per botó de selecció.
private void	<u>saveAsMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) Desar com, des de Menú.
private void	<u>saveMenuItemActionPerformed</u> (java.awt.event.ActionEvent evt) Desar, des de Menú.
(package private) boolean	<u>sortir</u> (java.lang.String titol, java.lang.String avis, boolean surt) Aquest mètode serveix per a que l'usuari de la GUI sigui preguntat per desar els canvis del treball actual quan pitja sortir, obrir o nou.

Methods inherited from class javax.swing.JFrame
addImpl, createRootPane, createRootPaneException, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isRootPaneCheckingEnabled, paramString, processKeyEvent, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame
, addNotify, addToFrameList, constructComponentName, finalize, getCursorType, getFrames, getIconImage, getMenuBar, getState, getTitle, initIDs, isResizable, postProcessKeyEvent, readObject, remove, removeFromFrameList, removeNotify, setCursor, setIconImage, setMenuBar, setResizable, setState, setTitle, writeObject

Methods inherited from class java.awt.Window
addOwnedWindow, addWindowListener, adjustListeningChildren, applyResourceBundle, applyResourceBundle, connectOwnedWindow, dispatchEventImpl, dispose, eventEnabled, getFocusOwner, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getOwnedWindows, getOwner, getToolkit, getWarningString, hide, isActive, isShowing, nextFocus, ownedInit, pack, postEvent, postWindowEvent, preProcessKeyEvent, processEvent, removeOwnedWindow, removeWindowListener, resetGC, setCursor.

setFocusOwner, setWarningString, show, toBack, toFront, transferFocus

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyOrientation, checkGD, countComponents, createChildHierarchyEvents, createHierarchyEvents, deliverEvent, dispatchEventToSelf, doLayout, findComponentAt, findComponentAt, findComponentAt, getAccessibleAt, getAccessibleChild, getAccessibleChildrenCount, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents_NoClientCode, getComponents, getInsets, getLayout, getMaximumSize, getMinimumSize, getMouseEventTarget, getPreferredSize, getWindow, insets, invalidate, invalidateTree, isAncestorOf, layout, lightweightPaint, lightweightPrint, list, list, locate, minimumSize, numListening, paint, paintComponents, paintHeavyweightComponents, postsOldMouseEvents, preferredSize, print, printComponents, printHeavyweightComponents, processContainerEvent, proxyEnableEvents, proxyRequestFocus, remove, removeAll, removeContainerListener, setFont, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, addPropertyChangeListener, areInputMethodsEnabled, bounds, checkImage, checkImage, checkWindowClosingException, coalesceEvents, contains, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, getAccessibleIndexInParent, getAccessibleStateSet, getBackground, getBounds, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont_NoClientCode, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getInputMethodRequests, getLocation, getLocation, getLocationOnScreen_NoTreeLock, getLocationOnScreen, getName, getNativeContainer, getParent_NoClientCode, getParent, getPeer, getSize, getSize, getToolkitImpl, getTreeLock, getWidth, getWindowForObject, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isDisplayable, isDoubleBuffered, isEnabled, isEnabledImpl, isFocusTraversable, isLightweight, isOpaque, isRecursivelyVisible, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseMotionEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, removePropertyChangeListener, repaint.

`repaint, repaint, repaint, requestFocus, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus`

Methods inherited from class java.lang.Object

`clone, equals, getClass, hashCode, notify, notifyAll, registerNatives, wait, wait, wait`

Methods inherited from interface java.awt.MenuContainer

`getFont, postEvent`

Field Detail

problema

GrupMissatges **problema**
En aquest objecte es gaurden totes les dades del problema. També, aquest objecte instància de la classe `GrupMissatges`és el que conté la majoria de mètodes càlcul sobre aquestes dades.

fitxerObert

`boolean fitxerObert`
Si en la sessió de treball ja s'ha especificat un fitxer de treball.

fitxerActual

`java.io.File fitxerActual`
Prova de Fitxer Actual epe

taula

`javax.swing.JTable taula`
Taula on s'introdueixen les dades referents als missatge. En aquesta taula també es mostren els resultats relatius a cada missatge.

taulaM

`TaulaMissatges` `taulaM`

La taula de missatges fa servir aquest model abstracte de taula.

percentatgeUsLimit

`int` `percentatgeUsLimit`

Constant on indiquem el tant per cent de càrrega del bus crític. Percentatges d'ús del bus CAN per sobre d'aquest percentatge es considerar crític. Aquest sencer indica el límit.

alfaLimit

`double` `alfaLimit`

Constant on indiquem el valor d'alfa crític. Valors d'alfa inferiors aquest valor es considerar crític. Aquest sencer indica el límit.

menuBar

`private javax.swing.JMenuBar` `menuBar`

fileMenu

`private javax.swing.JMenu` `fileMenu`

nouMenuItem

`private javax.swing.JMenuItem` `nouMenuItem`

openMenuItem

`private javax.swing.JMenuItem` `openMenuItem`

jSeparator1

`private javax.swing.JSeparator` `jSeparator1`

saveMenuItem

`private javax.swing.JMenuItem` `saveMenuItem`

saveAsMenuItem

```
private javax.swing.JMenuItem saveAsMenuItem
```

jSeparator2

```
private javax.swing.JSeparator jSeparator2
```

exitMenuItem

```
private javax.swing.JMenuItem exitMenuItem
```

toolsMenu

```
private javax.swing.JMenu toolsMenu
```

calcularMenuItem

```
private javax.swing.JMenuItem calcularMenuItem
```

jSeparator4

```
private javax.swing.JSeparator jSeparator4
```

modeMenu

```
private javax.swing.JMenu modeMenu
```

estandardMenuItem1

```
private javax.swing.JCheckBoxMenuItem estandardMenuItem1
```

extensMenuItem2

```
private javax.swing.JCheckBoxMenuItem extensMenuItem2
```

helpMenu

```
private javax.swing.JMenu helpMenu
```

contentsMenuItem

```
private javax.swing.JMenuItem contentsMenuItem
```

jSeparator3

```
private javax.swing.JSeparator jSeparator3
```

aboutMenuItem

```
private javax.swing.JMenuItem aboutMenuItem
```

panelEines

```
private javax.swing.JPanel panelEines
```

barraEines

```
private javax.swing.JToolBar barraEines
```

botoNou

```
private javax.swing.JButton botoNou
```

botoObrir

```
private javax.swing.JButton botoObrir
```

botoDesar

```
private javax.swing.JButton botoDesar
```

botoCalcularEines

```
private javax.swing.JButton botoCalcularEines
```

botoAjuda

```
private javax.swing.JButton botoAjuda
```

panelPrincipal

```
private javax.swing.JTabbedPane panelPrincipal
```

panelDades

```
private javax.swing.JPanel panelDades
```

campFrequencia

```
private javax.swing.JTextField campFrequencia
```

textFrequencia

```
private javax.swing.JLabel textFrequencia
```

rbotoExtens

```
private javax.swing.JRadioButton rbotoExtens
```

textFormat

```
private javax.swing.JLabel textFormat
```

rbotoEstandard

```
private javax.swing.JRadioButton rbotoEstandard
```

jLabel1

```
private javax.swing.JLabel jLabel1
```

jScrollPane1

```
private javax.swing.JScrollPane jScrollPane1
```

textDescripcio

```
private javax.swing.JTextArea textDescripcio
```

panelM

```
private javax.swing.JPanel panelM
```

panelResultats

```
private javax.swing.JPanel panelResultats
```

panelConclusions

```
private javax.swing.JPanel panelConclusions
```

labelResultat

```
private javax.swing.JLabel labelResultat
```

barraUs

```
private javax.swing.JProgressBar barraUs
```

labelUs

```
private javax.swing.JLabel labelUs
```

labelAlfa

```
private javax.swing.JLabel labelAlfa
```

textAlfa

```
private javax.swing.JTextField textAlfa
```

labelNumMissatges

```
private javax.swing.JLabel labelNumMissatges
```

labelNumMissatgesProgramables

```
private javax.swing.JLabel labelNumMissatgesProgramables
```

botoCalcular

```
private javax.swing.JButton botoCalcular
```

Constructor Detail

CAN1

```
public CAN1()
```

PROVACrea un GUI form CAN1 i inicialitza la taula

Method Detail

initComponents

```
private void initComponents()
```

This method is called from within the constructor to initialize the form.

WARNING: Do NOT modify this code. The content of this method is always regenerated by the FormEditor.

contentsMenuItemActionPerformed

```
private void
```

```
contentsMenuItemActionPerformed(java.awt.event.ActionEvent evt)
```

S'obre una finestra d'ajuda, amb les instruccions bàsiques d'ús del jCAN1.

Parameters:

evt - Event

aboutMenuItemActionPerformed

```
private void
```

```
aboutMenuItemActionPerformed(java.awt.event.ActionEvent evt)
```

Sobre una finestra amb les dades "Sobre..." el jCAN1.

estandardMenuItem1ActionPerformed

```
private void
```

```
estandardMenuItem1ActionPerformed(java.awt.event.ActionEvent evt)
```

Seleccionar mode estàndard per Menú.

extensMenuItem2ActionPerformed

```
private void
```

```
extensMenuItem2ActionPerformed(java.awt.event.ActionEvent evt)
```

Seleccionar mode extens per Menú.

botoCalcularEinesActionPerformed

```
private void  
botoCalcularEinesActionPerformed(java.awt.event.ActionEvent evt)  
    Inici del procés de càlcul, iniciat des del botó.
```

botoDesarActionPerformed

```
private void  
botoDesarActionPerformed(java.awt.event.ActionEvent evt)  
    Inici rutina de desar, des de botó.
```

nouMenuItemActionPerformed

```
private void  
nouMenuItemActionPerformed(java.awt.event.ActionEvent evt)  
    Es crear un nou problema, iniciat des de menú.
```

botoNouActionPerformed

```
private void botoNouActionPerformed(java.awt.event.ActionEvent evt)  
    Es crear un nou problema, iniciat des de botó. Primer es crida a la rutina sortir  
    per desar possibles canvis.
```

botoObrirActionPerformed

```
private void  
botoObrirActionPerformed(java.awt.event.ActionEvent evt)  
    Inici rutina d'obrir problema, iniciada des de botó.
```

rbotoExtensActionPerformed

```
private void  
rbotoExtensActionPerformed(java.awt.event.ActionEvent evt)  
    Seleccionar mode extens per botó de selecció.
```

rbotoEstandardActionPerformed

```
private void  
rbotoEstandardActionPerformed(java.awt.event.ActionEvent evt)  
    Seleccionar mode estàndard per botó de selecció
```

botoCalcularActionPerformed


```
private void  
botoCalcularActionPerformed(java.awt.event.ActionEvent evt)  
    Inici rutina de càlcul del problema, des de botó: posarMissatges(),  
    calcularProblema(), actualitzarTaula(), actualitzarGUI();
```

saveAsMenuItemActionPerformed

```
private void  
saveAsMenuItemActionPerformed(java.awt.event.ActionEvent evt)  
    Desar com, des de Menú.
```

saveMenuItemActionPerformed

```
private void  
saveMenuItemActionPerformed(java.awt.event.ActionEvent evt)  
    Desar, des de Menú.
```

campFrequenciaActionPerformed

```
private void  
campFrequenciaActionPerformed(java.awt.event.ActionEvent evt)  
    Canviem el valor de la freqüència del bus problema, des de camp de text.
```

jRadioButton2StateChanged

```
private void  
jRadioButton2StateChanged(javax.swing.event.ChangeEvent evt)
```

cutMenuItemActionPerformed

```
private void  
cutMenuItemActionPerformed(java.awt.event.ActionEvent evt)  
    Inici rutina de calcular, iniciada des de Menú.
```

openMenuItemActionPerformed

```
private void  
openMenuItemActionPerformed(java.awt.event.ActionEvent evt)  
    Inici rutina obrir fitxer problema, des de Menú.
```

radioBotoExtensActionPerformed

```
private void  
radioBotoExtensActionPerformed(java.awt.event.ActionEvent evt)
```

radioButoEstandardActionPerformed

```
private void  
radioButoEstandardActionPerformed(java.awt.event.ActionEvent evt)
```

jButton2ActionPerformed

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)  
    Obre finestra amb ajuda.
```

jTextField1ActionPerformed

```
private void  
jTextField1ActionPerformed(java.awt.event.ActionEvent evt)
```

exitMenuItemActionPerformed

```
private void  
exitMenuItemActionPerformed(java.awt.event.ActionEvent evt)  
    Crida a funció sortida , des de Menú.
```

exitForm

```
private void exitForm(java.awt.event.WindowEvent evt)  
    Sortir de la aplicació.  
    Parameters:  
    evt - Event de sortida
```

main

```
public static void main(java.lang.String[] args)  
    Crear una GUI, la dimensiona i mostra per pantalla.  
    Parameters:  
    args - Els arguments passats des de la línia de commandes.
```

posarMissatges

```
void posarMissatges()  
    Crea un objecte GrupMissatges, inicialitza el problema. I li posa les dades  
    que tenim a la GUI.
```

calcularProblema

```
void calcularProblema()
```

Calcular el problema, GrupMissatges.

actualitzarTaula

```
void actualitzarTaula()  
    Posar les dades dels missatges del problema, GrupMissatges, a la taula taula  
    de Missatges de la GUI.
```

actualitzarGUI

```
void actualitzarGUI()  
    Actualitza els resultats de la GUI. Conclusions, dades bus, resultats,...  
    Aquesta funció no s'utilitza a l'obrir un fitxer anteriorment desat.
```

mostrarResultats

```
void mostrarResultats()  
    Actualitza les cel·les de resultats que té la taula de missatges. Els temps de  
    lliurament i si els missatge són programables.
```

mostrarProblema

```
void mostrarProblema()  
    Per Debbbug. Mostra per sortida de systema les dades del problema.
```

sortir

```
boolean sortir(java.lang.String titol,  
               java.lang.String avis,  
               boolean surt)
```

Aquest mètode serveix per a que l'usuari de la GUI sigui preguntat per desar els canvis del treball actual quan pitja sortir, obrir o nou.

Parameters:

titol - Text que apareixerà en la finestra.

avis - Missatge d'avis que es donarà a l'usuari.

surt - Si després de contestar s'ha de sortir del programa.

Returns:

Valor que identifica l'èxit de l'operació.

Class FiltreFitxersCAN

```
java.lang.Object
|
+--javax.swing.filechooser.FileFilter
|
+--FiltreFitxersCAN
```

```
public class FiltreFitxersCAN
extends javax.swing.filechooser.FileFilter
```

Filtre per fitxers de tipus CAN. Un filtre de fitxer permet que des d'una finestra de selecció de fitxers `JFileChooser` només se'ns mostrin els fitxers amb extensions amb sentit per al programa.

Constructor Summary

FiltreFitxersCAN()

Method Summary	
boolean	<u>accept</u> (java.io.File f) Mètode en carregat de dir quins fitxers són del programa, per ser llistats.
java.lang.String	<u>getDescription</u> () Obtenir la cadena de text descriptiva dels tipus de fitxers que seran mostrats.

Methods inherited from class java.lang.Object
, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructor Detail

FiltreFitxersCAN

```
public FiltreFitxersCAN()
```

Method Detail

accept

```
public boolean accept(java.io.File f)
```

Mètode en carregat de dir quins fitxers són del programa, per ser llistats.

Overrides:

`accept` in class `javax.swing.filechooser.FileFilter`

Parameters:

`f` - Fitxer subjecte a ser validat com a fitxer de programa, se li aplicarà el filtre.

Returns:

Si el fitxer és del programa, que ha superat el filtre.

getDescription

```
public java.lang.String getDescription()
```

Obtenir la cadena de text descriptiva dels tipus de fitxers que seran mostrats.

Overrides:

`getDescription` in class `javax.swing.filechooser.FileFilter`

Returns:

Retorna la cadena de text descriptiva.

Class GrupMissatges

```
java.lang.Object
|
+--GrupMissatges
All Implemented Interfaces:
    java.io.Serializable
```

```
public class GrupMissatges
extends java.lang.Object
implements java.io.Serializable
```

Un sistema basat en un bus CAN ve descrit per un grup de missatges i dades globals del sistema, tots aquests missatges vene en aquesta classe amb un vector d'objectes Missatge. Aquesta classe conté aquestes dades i els mètodes necessaris per treballar sobre aquestes dades. Les dades ha de ser obtingudes de l'objecte directament; però la seva inicialització, canvis, càlculs i operacions es duen a terme a través dels diversos mètodes que proporciona aquesta classe.

See Also:
[Serialized Form](#)

Field Summary	
(package private) double	<u>alfa</u> Variable que indica la qualitat del bus.
(package private) java.lang.String	<u>descripcio</u> Descripció o comentari sobre el sistema problema.
(package private) int	<u>frequencia</u> Freqüència en Kbits per segon a la que funciona el bus.
(package private) Missatge[]	<u>grup</u> Vector d'objectes Missatge.
(package private) int	<u>numeroMissatges</u> Número de missatges de difenent identificador que hi ha al sistema.
(package private) int	<u>numeroMissatgesProgramables</u> Número de missatges, diferent identificador, que són programables al sistema plantejat, bus en qüestió,
(package private) double	<u>percentatgeUs</u> Percentatge de temps del bus ocupat per missatge.
(package private) boolean	<u>programable</u> Si tots els missatges del bus són programables.
(package private) boolean	<u>standardCAN</u> Si el bus treballa en mode estàndard o en mode extens.

Constructor Summary

GrupMissatges(int fre, int nuM, boolean stC, java.lang.String des)
Constructor del problema.

Method Summary

int	bloqueig (int a) Calcula el temps de bloqueig al cual pot estar sotmés un missatge per un altre de menys prioritat.
void	calc () Per cada Missatge es calcula el número màxim de bits de stuff que poden tenir.
void	calcAlfa () Calcula el factor de qualitat alfa que té els sistema, bus problema.
void	calcBloqueig () Es calcula per cada missatge el temps de bloqueig que pot tenir.
void	calcEspera () Calcula per cada missatge del problema el temps total màxim que segons el model el missatge pot estar pendent de ser lliurat.
void	calcNumeroMissatgesProgramables () CALcula el número de missatges que són programables entre tots els que té el bus-sistema problema.
void	calcUs () Càcula el percentatge d'ús (temps) que cada missatge fa del bus.
void	printDadesText () Funció amb finalitats de debug.
void	setMissatge (int index, int nu, java.lang.String na, int ji, int pe, int dL, int nB) S'encarrega de entrar les dades de cada missatge.

Methods inherited from class java.lang.Object

, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

grup

Missatge[] grup

Vector d'objectes `Missatge`. Conté tots els missatges del bus indexat, de més a menys prioritat.

percentatgeUs

`double percentatgeUs`

Percentatge de temps del bus ocupat per missatge. Càrrega del bus.

alfa

`double alfa`

Variable que indica la qualitat del bus. Valors superiors a 1 indiquen que el bus és programable.

programable

`boolean programable`

Si tots els missatges del bus són programables.

frequencia

`int frequencia`

Freqüència en Kbits per segon a la que funciona el bus.

numeroMissatges

`int numeroMissatges`

Número de missatges de diferent identificador que hi ha al sistema.

numeroMissatgesProgramables

`int numeroMissatgesProgramables`

Número de missatges, diferent identificador, que són programables al sistema plantejat, bus en qüestió,

standardCAN

`boolean standardCAN`

Si el bus treballa en mode estàndard o en mode extens.

descripcio

java.lang.String **descripcio**

Descripció o comentari sobre el sistema problema. Aquest text és lliure.

Constructor Detail

GrupMissatges

```
public GrupMissatges(int fre,  
                    int nuM,  
                    boolean stC,  
                    java.lang.String des)
```

Constructor del problema. Dades comunes a tots els missatges del bus (Freqüència, mode, descripció) i número de missatges que el componen.

Parameters:

fre - Freqüència en Kilobits a la qual treballa el bus.

nuM - Número de missatges que conté el sistema (bus problema).

stC - Mode de bus estàndard (False indica mode extens).

des - Descripció del sistema. Qualsevol text útil per l'usari.

Method Detail

setMissatge

```
public void setMissatge(int index,  
                      int nu,  
                      java.lang.String na,  
                      int ji,  
                      int pe,  
                      int dL,  
                      int nB)
```

S'encarrega de entrar les dades de cada missatge. Valida les dades.

Converteix els temps expressats en microsegons a número de períodes de bit.

Parameters:

index - Posició al vector de missatge.

nu - Identificador del missatge.

na - Nom o descripció del missatge.

ji - Temps de jitter.

pe - Període del missatge.

dL - Temps de deadline.

nB - Número de Bytes del missatge.

calc

```
public void calc()
```

Per cada Missatge es calcula el número màxim de bits de stuff que poden tenir.

calcBloqueig

```
public void calcBloqueig()
```


Es calcula per cada missatge el temps de bloqueig que pot tenir.

calcEspera

```
public void calcEspera()
```

Calcula per cada missatge del problema el temps total màxim que segons el model el missatge pot estar pendent de ser lliurat. En número de períodes de bit.

calcUs

```
public void calcUs()
```

Càlcula el percentatge d'ús (temps) que cada missatge fa del bus.

calcAlfa

```
public void calcAlfa()
```

Calcula el factor de qualitat alfa que té el sistema, bus problema.

printDadesText

```
public void printDadesText()
```

Funció amb finalitats de debug. Treu per pantalla el contingut de tots els missatges.

calcNumeroMissatgesProgramables

```
public void calcNumeroMissatgesProgramables()
```

Calcula el número de missatges que són programables entre tots els que té el bus-sistema problema.

bloqueig

```
public int bloqueig(int a)
```

Calcula el temps de bloqueig al qual pot estar sotmés un missatge per un altre de menys prioritat.

Parameters:

a - Número del missatge en el vector.

Returns:

Temps de bloqueig que pot tenir el missatge per altres de menys prioritat.

Class Missatge

```
java.lang.Object
|
+--Missatge
All Implemented Interfaces:
    java.io.Serializable
```

```
public class Missatge
extends java.lang.Object
implements java.io.Serializable
```

Un Missatge CAN ve caracteritzat per una serie de dades que la classe Missatge conté. Porta associats una sèrie de mètodes, com el de calcular que s'encarrega de fer aquells càlculs que només afecten a un missatge en concret. Per posar les dades és pot utilitzar un mètode estàndard tipus set. Però per llegir les dades s'ha simplificat la tasca de programació fent les variables públiques i deixant als objectes pares treballar directament sobre les variables en lloc de procedir amb una colecció de mètodes `get()` i `set()`

See Also:
[Serialized Form](#)

Field Summary	
(package private) int	<u>bitsFixes</u> Número de bits fixes que conté el missatge, independents del número de bits que pugui tenir segons la grandaria de les dades.
(package private) int	<u>deadLine</u> El temps màxim que pot esperar un missatge per ser lliurat.
(package private) int	<u>jitter</u> El temps el qual trigar un node en tindre el missatge preparat per lliurar des de que es produeix la necessitat de ser lliurat.
(package private) java.lang.String	<u>name</u> Nom descriptiu del missatge.
(package private) int	<u>number</u> Número identificador del missatge, marca la prioritat.
(package private) int	<u>numBytes</u> El número de bytes de dades que conté el missatge.
(package private) boolean	<u>ok</u> Indica si el missatge és programable.
(package private) java.lang.String	<u>percentatgeUs</u> Percentatge que té el missatge sobre el bus.
(package private)	<u>periode</u>

int	El temps que trigar un missatge a tornar a ser generat, en número de períodes de bit.
(package private) int	<u>stuffableBits</u> Número de bits succeptibles a ser sotmesos a la tècnica de stuff.
(package private) int	<u>stuffBits</u> Número màxim de bits de stuff que té el missatge.
(package private) int	<u>tempsBloqueig</u> Temps el qual el missatge pot estar aturat abans de ser transmés per culpa de que un missatge de menys prioritat està ocupant el bus.
(package private) int	<u>tempsEspera</u> Temps total màxim que segons el model el missatge pot estar aturat per culpa de que el bus està ocupat.
(package private) int	<u>tempsPropagacio</u> Temps que trigarà el missatge en propagar-se pel bus.
(package private) int	<u>tempsReal</u> Temps total màxim que segons el model el missatge pot estar pendent de ser lliurat.

Constructor Summary

Missatge()

Method Summary

void	<u>calc</u> () Calcula el número de bits de stuff màxim que pot tenir el missatge.
void	<u>setMissatge</u> (int nu, java.lang.String na, int ji, int pe, int dL, int nB, boolean sC) Introdueix les dades necessaries per tenir ben definit un missatge.

Methods inherited from class java.lang.Object

, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

numBytes

```
int numBytes
```

El número de bytes de dades que conté el missatge.

deadLine

```
int deadLine
```

El temps màxim que pot esperar un missatge per ser lliurat. En número de períodes de bit.

name

```
java.lang.String name
```

Nom descriptiu del missatge.

number

```
int number
```

Número identificador del missatge, marca la prioritat.

jitter

```
int jitter
```

El temps el qual trigar un node en tindre el missatge preparat per lliurar des de que es produeix la necessitat de ser lliurat. En número de períodes de bit.

periode

```
int periode
```

El temps que trigar un missatge a tornar a ser generat, en número de períodes de bit.

bitsFixes

```
int bitsFixes
```

Número de bits fixes que conté el missatge, independents del número de bits que pugui tenir segons la grandaria de les dades.

stuffBits

```
int stuffBits
```

Número màxim de bits de stuff que té el missatge.

tempsPropagacio

`int tempsPropagacio`

Temps que trigarà el missatge en propagar-se pel bus. És equivalent al número de bits totals que conté el missatge; ja que s'expressa en número de períodes de bit.

stuffableBits

`int stuffableBits`

Número de bits succeptibles a ser sotmesos a la tècnica de stuff.

tempsBloqueig

`int tempsBloqueig`

Temps el qual el missatge pot estar aturat abans de ser transmés per culpa de que un missatge de menys prioritat està ocupant el bus. En número de períodes de bit.

tempsEspera

`int tempsEspera`

Temps total màxim que segons el model el missatge pot estar aturat per culpa de que el bus està ocupat. En número de períodes de bit.

tempsReal

`int tempsReal`

Temps total màxim que segons el model el missatge pot estar pendent de ser lliurat. En número de períodes de bit.

percentatgeUs

`java.lang.String percentatgeUs`

Percentatge que té el missatge sobre el bus.

ok

`boolean ok`

Indica si el missatge és programable.

Constructor Detail

Missatge

public Missatge()

Method Detail

setMissatge

```
public void setMissatge(int nu,
                        java.lang.String na,
                        int ji,
                        int pe,
                        int dL,
                        int nB,
                        boolean sC)
```

Introdueix les dades necessaries per tenir ben definit un missatge.

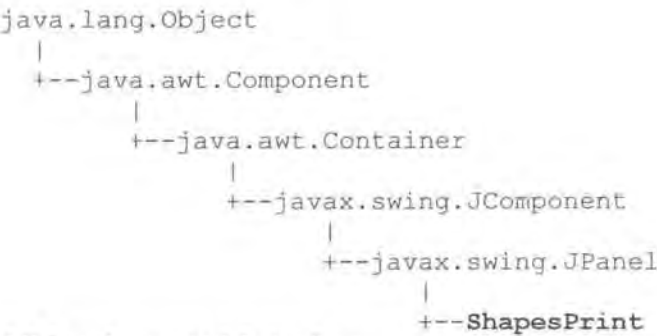
Parameters:

- nu - Identificador Missatge.
- na - Nom descriptiu del missatge.
- ji - Temps de Jitter.
- pe - període del missatge.
- dL - Deadline del Missatge.
- nB - Número de Bytes.
- sC - Si és mode de bus estàndard o extens.

calc

```
public void calc()
    Calcula el número de bits de stuff màxim que pot tenir el missatge.
```


Class ShapesPrint



All Implemented Interfaces:
javax.accessibility.Accessible, java.awt.event.ActionListener,
java.util.EventListener, java.awt.image.ImageObserver,
java.awt.MenuContainer, java.awt.print.Printable, java.io.Serializable

public class **ShapesPrint**
extends javax.swing.JPanel
implements java.awt.print.Printable, java.awt.event.ActionListener
See Also:
 [Serialized Form](#)

Inner classes inherited from class javax.swing.JPanel
javax.swing.JPanel.AccessibleJPanel

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent, javax.swing.JComponent.ActionStandin, javax.swing.JComponent.IntVector, javax.swing.JComponent.KeyboardState

Inner classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Inner classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.AWTTreeLock

Field Summary	
(package private) static java.awt.Color	<u>bg</u>
(package private) static javax.swing.JButton	<u>button</u>
(package private) static float[]	<u>dash1</u>
(package private) static java.awt.BasicStroke	<u>dashed</u>
(package private) static java.awt.Color	<u>fg</u>
(package private) static java.awt.Color	<u>red</u>
(package private) static java.awt.BasicStroke	<u>stroke</u>
(package private) static java.awt.Color	<u>white</u>
(package private) static java.awt.BasicStroke	<u>wideStroke</u>

Fields inherited from class javax.swing.JPanel
defaultLayout, uiClassID

Fields inherited from class javax.swing.JComponent
<code>_bounds, accessibleContext, actionMap, ACTIONMAP_CREATED, alignmentX, alignmentY, ANCESTOR_INPUTMAP_CREATED, ANCESTOR_USING_BUFFER, ancestorInputMap, ancestorNotifier, autoscroller, border, changeSupport, clientProperties, CREATED_DOUBLE_BUFFER, flags, FOCUS_INPUTMAP_CREATED, focusInputMap, HAS_FOCUS, htmlKey, htmlView, inputVerifier, IS_DOUBLE_BUFFERED, IS_OPAQUE, IS_PAINTING_TILE, IS_PRINTING, IS_PRINTING_ALL, KEY_EVENTS_ENABLED, KEYBOARD_BINDINGS_KEY, listenerList, maximumSize, minimumSize, NEXT_FOCUS, paintImmediatelyClip, paintingChild, preferredSize, readObjectCallbacks, REQUEST_FOCUS_DISABLED, tmpRect, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, verifyInputWhenFocusTarget, vetoableChangeSupport, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW, WHEN_IN_FOCUSED_WINDOW_BINDINGS, WIF_INPUTMAP_CREATED, windowInputMap</code>

Fields inherited from class java.awt.Container

component, containerListener, containerSerializedDataVersion, dbg, dispatcher, layoutMgr, listeningBoundsChildren, listeningChildren, maxSize, ncomponents, printing, printingThreads, serialVersionUID

Fields inherited from class java.awt.Component

actionListenerK, adjustmentListenerK, appContext, background, BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, componentListener, componentListenerK, componentOrientation, componentSerializedDataVersion, containerListenerK, cursor, dropTarget, enabled, eventMask, focusListener, focusListenerK, font, foreground, graphicsConfig, hasFocus, height, hierarchyBoundsListener, hierarchyBoundsListenerK, hierarchyListener, hierarchyListenerK, incRate, inputMethodListener, inputMethodListenerK, isInc, isPacked, itemListenerK, keyListener, keyListenerK, LEFT_ALIGNMENT, locale, LOCK, metrics, minSize, mouseListener, mouseListenerK, mouseMotionListener, mouseMotionListenerK, name, nameExplicitlySet, newEventsOnly, ownedWindowK, parent, peer, peerFont, popups, prefSize, privateKey, RIGHT_ALIGNMENT, textListenerK, TOP_ALIGNMENT, valid, visible, width, windowClosingException, windowListenerK, x, y

Fields inherited from interface java.awt.print.Printable

NO_SUCH_PAGE, PAGE_EXISTS

Fields inherited from interface java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ShapesPrint()

Method Summary

void	<u>actionPerformed</u> (java.awt.event.ActionEvent e)
void	<u>drawShapes</u> (java.awt.Graphics2D g2)

static void	<u>main</u> (java.lang.String[] s)
void	<u>paintComponent</u> (java.awt.Graphics g)
int	<u>print</u> (java.awt.Graphics g, java.awt.print.PageFormat pf, int pi)

Methods inherited from class javax.swing.JPanel

```
, getAccessibleContext, getUIClassID, paramString, updateUI,
writeObject
```

Methods inherited from class `javax.swing.JComponent`

_paintImmediately, addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, adjustPaintFlags, alwaysOnTop, checkIfChildObscuredBySibling, clearFocusOwners, componentInputMapChanged, computeVisibleRect, computeVisibleRect, compWriteObjectNotify, contains, createToolTip, disable, enable, enableSerialization, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getActionMap, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperties, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getCreatedDoubleBuffer, getDebugGraphicsOptions, getFlag, getGraphics, getHeight, getInputMap, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVerifyInputWhenFocusTarget, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, hide, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidRoot, paint, paintBorder, paintChildren, paintImmediately, paintImmediately, paintWithBuffer, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processFocusEvent, processKeyBinding, processKeyBindings, processKeyBindingsForAllComponents, processKeyEvent, processMouseEvent, putClientProperty, readObject, rectangleIsObscured, rectangleIsObscuredBySibling, registerKeyboardAction, registerKeyboardAction, registerWithKeyboardManager, registerWithKeyboardManager, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions.


```

reshape, revalidate, scrollRectToVisible, setActionMap,
setAlignmentX, setAlignmentY, setAutoscrolls, setBackground,
setBorder, setCreatedDoubleBuffer, setDebugGraphicsOptions,
setDoubleBuffered, setEnabled, setFlag, setFont, setForeground,
setInputMap, setInputVerifier, setMaximumSize, setMinimumSize,
setNextFocusableComponent, setOpaque, setPaintingChild,
setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI,
setVerifyInputWhenFocusTarget, setVisible, shouldDebugGraphics,
superProcessMouseEvent, unregisterKeyboardAction,
unregisterWithKeyboardManager, unregisterWithKeyboardManager, update

```

Methods inherited from class java.awt.Container

```

add, add, add, add, add, addContainerListener, addImpl,
adjustListeningChildren, applyOrientation, checkGD, countComponents,
createChildHierarchyEvents, createHierarchyEvents, deliverEvent,
dispatchEventImpl, dispatchEventToSelf, doLayout, eventEnabled,
findComponentAt, findComponentAt, findComponentAt, getAccessibleAt,
getAccessibleChild, getAccessibleChildrenCount, getComponent,
getComponentAt, getComponentAt, getComponentCount,
getComponents_NoClientCode, getComponents, getLayout,
getMouseEventTarget, getWindow, initIDs, insets, invalidate,
invalidateTree, isAncestorOf, layout, lightweightPaint,
lightweightPrint, list, list, locate, minimumSize, nextFocus,
numListening, paintComponents, paintHeavyweightComponents,
postProcessKeyEvent, postsOldMouseEvents, preferredSize,
preProcessKeyEvent, printComponents, printHeavyweightComponents,
processContainerEvent, processEvent, proxyEnableEvents,
proxyRequestFocus, remove, remove, removeAll,
removeContainerListener, setFocusOwner, setLayout, transferFocus,
validate, validateTree

```

Methods inherited from class java.awt.Component

```

action, add, addComponentListener, addFocusListener,
addHierarchyBoundsListener, addHierarchyListener,
addInputMethodListener, addKeyListener, addMouseListener,
addMouseMotionListener, areInputMethodsEnabled, bounds, checkImage,
checkImage, checkWindowClosingException, coalesceEvents,
constructComponentName, contains, createImage, createImage,
disableEvents, dispatchEvent, enable, enableEvents,
enableInputMethods, getAccessibleIndexInParent,
getAccessibleStateSet, getBackground, getBounds, getColorModel,
getComponentOrientation, getCursor, getDropTarget,
getFont_NoClientCode, getFont, getFontMetrics, getForeground,
getGraphicsConfiguration, getInputContext, getInputMethodRequests,
getLocale, getLocation, getLocationOnScreen_NoTreeLock,
getLocationOnScreen, getName, getNativeContainer,
getParent_NoClientCode, getParent, getPeer, getSize, getToolkit,
getToolkitImpl, getTreeLock, getWindowForObject, gotFocus,
handleEvent, imageUpdate, inside, isDisplayable, isEnabled,
isEnabledImpl, isLightweight, isRecursivelyVisible, isShowing,
isValid, isVisible, keyDown, keyUp, list, list, list, location,
lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove,

```


mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, repaint, repaint, repaint, resetGC, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, wait, wait, wait

Field Detail

bg

static final java.awt.Color **bg**

fg

static final java.awt.Color **fg**

red

static final java.awt.Color **red**

white

static final java.awt.Color **white**

stroke

static final java.awt.BasicStroke **stroke**

wideStroke

static final java.awt.BasicStroke **wideStroke**

dash1

```
static final float[] dash1
```

dashed

```
static final java.awt.BasicStroke dashed
```

button

```
static final javax.swing.JButton button
```

Constructor Detail

ShapesPrint

```
public ShapesPrint()
```

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
    Specified by:
        actionPerformed in interface java.awt.event.ActionListener
```

paintComponent

```
public void paintComponent(java.awt.Graphics g)
    Overrides:
        paintComponent in class javax.swing.JComponent
```

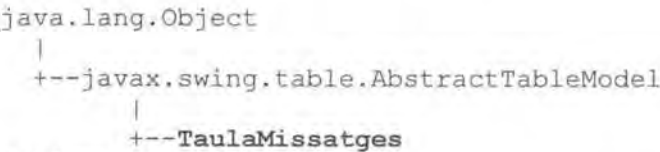
drawShapes

```
public void drawShapes(java.awt.Graphics2D g2)
```

print

```
public int print(java.awt.Graphics g,
                 java.awt.print.PageFormat pf,
                 int pi)
    throws java.awt.print.PrinterException
    Specified by:
        print in interface java.awt.print.Printable
```

Class TaulaMissatges



All Implemented Interfaces:
java.io.Serializable, javax.swing.table.TableModel

```
public class TaulaMissatges
extends javax.swing.table.AbstractTableModel
```

Aquesta classe és un model abstracte de taula per estendre la funcionalitat de la classe *JTable*. En concret es defineix tota la funcionalitat de la taula d'introducció de missatges. Una taula de 9 columnes i 150 files. On cada columna té unes propietats diferents. Hi ha que són editables i hi ha que no. Cada columna alberga objectes de diferents tipus (sencers, cadenes de text,...)

See Also:
[Serialized Form](#)

Field Summary	
(package private) java.lang.String[]	<u>columnNames</u> Vector de cadenes de text que conté els títols de les 9 columnes.
(package private) java.lang.Object[][]	<u>data</u> Matriu d'objectes de 150x9.
private boolean	<u>DEBUG</u> Variable amb finalitat de debug.

Fields inherited from class javax.swing.table.AbstractTableModel
listenerList

Constructor Summary
<u>TaulaMissatges()</u> Constructor de la classe, inicialitza la taula als valors per defecte.

Method Summary	
java.lang.Class	<u>getColumnClass</u> (int c) Per obtenir el tipus de objecte, la classe, que defineix una columna.
int	<u>getColumnCount</u> () Número de columnes de la taula.
java.lang.String	<u>columnName</u> (int col) Per obtenir el nom d'una columna en concret.
int	<u>getRowCount</u> () Retorna el número de files de la taula.
java.lang.Object	<u>getValueAt</u> (int row, int col) Per obtenir el valor d'un objecte d'una cel·la en concret.
boolean	<u>isCellEditable</u> (int row, int col) Per coneixer si una cel·la és editable.
private void	<u>printDebugData</u> () Mètode per debug.
void	<u>setNumBytes</u> (javax.swing.table.TableColumn numBytesColumn) Crea un JComboBox per la columna del número de Bytes del missatge per tal de tenir només uns valors possibles.
void	<u>setValueAt</u> (java.lang.Object value, int row, int col) Aquest mètode s'encarrega de posar el valor passat com argument a la cel·la, indicat per un número de columna i un número de fila passats com arguments.

Methods inherited from class javax.swing.table.AbstractTableModel
addTableModelListener, findColumn, fireTableCellUpdated, fireTableChanged, fireTableDataChanged, fireTableRowsDeleted, fireTableRowsInserted, fireTableRowsUpdated, fireTableStructureChanged, getListeners, removeTableModelListener

Methods inherited from class java.lang.Object
, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

DEBUG

```
private boolean DEBUG
```

Variable amb finalitat de debug. Activa el debug.

data

```
final java.lang.Object[][] data
```

Matriu d'objectes de 150x9. Vector de 150 vectors de 9 objectes.

columnNames

```
final java.lang.String[] columnNames
```

Vector de cadenes de text que conté els títols de les 9 columnes.

Constructor Detail

TaulaMissatges

```
public TaulaMissatges()
```

Constructor de la classe, inicialitza la taula als valors per defecte.

Method Detail

getColumnCount

```
public int getColumnCount()
```

Número de columnes de la taula. En aquest model sempre serà 9.

Overrides:

getColumnCount in class javax.swing.table.AbstractTableModel

Returns:

Retorna el número de columnes.

getRowCount

```
public int getRowCount()
```

Retorna el número de files de la taula.

Overrides:

getRowCount in class javax.swing.table.AbstractTableModel

Returns:

Número de files.

getColumnName


```
public java.lang.String getColumnName(int col)
```

Per obtenir el nom d'una columna en concret.

Overrides:

getColumnName in class javax.swing.table.AbstractTableModel

Parameters:

col - Número de la columna a llegir.

Returns:

Retorna una cadena amb el nom de la columna.

getValueAt

```
public java.lang.Object getValueAt(int row,  
                                   int col)
```

Per obtenir el valor d'un objecte d'una cel·la en concret. Retorna l'objecte.

Overrides:

getValueAt in class javax.swing.table.AbstractTableModel

Parameters:

row - Número de fila.

col - Número de columna.

Returns:

Retorna l'objecte contingut.

getColumnClass

```
public java.lang.Class getColumnClass(int c)
```

Per obtenir el tipus de objecte, la classe, que defineix una columna. Si no es fes servir no podríem mostrar resultats com Ok, amb un check box.

Overrides:

getColumnClass in class javax.swing.table.AbstractTableModel

Parameters:

c - Número de la columna.

Returns:

Retorna el tipus de classe.

isCellEditable

```
public boolean isCellEditable(int row,  
                              int col)
```

Per coneixer si una cel·la és editable.

Overrides:

isCellEditable in class javax.swing.table.AbstractTableModel

Parameters:

row - Columna de la cel·la.

col - Columna de la cel·la.

Returns:

Retorna si és editable (*true*) o no (*false*).

setValueAt

```
public void setValueAt(java.lang.Object value,  
                        int row,  
                        int col)
```

Aquest mètode s'encarrega de posar el valor passat com argument a la cel·la, indicat per un número de columna i un número de fila passats com arguments.

Overrides:

setValueAt in class javax.swing.table.AbstractTableModel

Parameters:

value - Valor, objecte, a posar a la cel·la.

row - Número de la columna.

col - Número de la fila.

printDebugData

```
private void printDebugData()
```

Mètode per debug. Mostra per pantalla les dades actuals de la taula.

setNumBytes

```
public void  
setNumBytes(javax.swing.table.TableColumn numBytesColumn)
```

Crea un JComboBox per la columna del número de Bytes del missatge per tal de tenir només uns valors possibles.

Parameters:

numBytesColumn - Columna pel número de Bytes del missatge.

Class Utils

```
java.lang.Object
|
+--Utils
```

```
public class Utils
extends java.lang.Object
```

Aquesta classe té la llista d'extensions vàildes i té el mètode capaç d'extreure l'extensió d'un fitxer.

Field Summary	
static java.lang.String	<u>can</u> Extensió estàndard de fitxers CAN.
static java.lang.String	<u>ctf</u> Extensió optativa pels fitxers del programa.
static java.lang.String	<u>sbc</u> Extensió dels fitxers amb els quals el programa treballa.

Constructor Summary	
<u>Utils()</u>	

Method Summary	
static java.lang.String	<u>getExtension</u> (java.io.File f) Mètode public que permet obtenir l'extensió del fitxer passat com argument.

Methods inherited from class java.lang.Object	
, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait	

Field Detail

ctf

```
public static final java.lang.String ctf
```

Extensió optativa pels fitxers del programa. CAN Tool File.

can

```
public static final java.lang.String can
```

Extensió estàndard de fitxers CAN.

sbc

```
public static final java.lang.String sbc
```

Extensió dels fitxers amb els quals el programa treballa. On es guarda totes les dades d'un bus, sistema problema. Simulació Bus CAN.

Constructor Detail

Utils

```
public Utils()
```

Method Detail

getExtension

```
public static java.lang.String getExtension(java.io.File f)
```

Mètode public que permet obtenir l'extensió del fitxer passat com argument.

Parameters:

f - Fitxer del qual es vol obtenir l'extensió.

Returns:

Retorna una cadena de text que conté l'extensió del fitxer. En cas negatiu retorna un `null`.

Hierarchy For Package

Class Hierarchy

- class java.lang.Object
 - class javax.swing.table.AbstractTableModel (implements java.io.Serializable, javax.swing.table.TableModel)
 - class TaulaMissatges
 - class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
 - class java.awt.Container
 - class javax.swing.JComponent (implements java.io.Serializable)
 - class javax.swing.JPanel (implements javax.accessibility.Accessible)
 - class **ShapesPrint** (implements java.awt.event.ActionListener, java.awt.print.Printable)
 - class java.awt.Window (implements javax.accessibility.Accessible)
 - class java.awt.Frame (implements java.awt.MenuContainer)
 - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
 - class CAN1
 - class javax.swing.filechooser.FileFilter
 - class **FiltreFitxersCAN**
 - class **GrupMissatges** (implements java.io.Serializable)
 - class Missatge (implements java.io.Serializable)
 - class Utils

Índex

A

aboutMenuItem - Variable in class CAN1

aboutMenuItemActionPerformed(ActionEvent) - Method in class CAN1

Sobre una finestra amb les dades "Sobre..." el jCAN1.

accept(File) - Method in class FiltreFitxersCAN

Mètode en carregat de dir quins fitxers són del programa, per ser llistats.

actionPerformed(ActionEvent) - Method in class ShapesPrint

actualitzarGUI() - Method in class CAN1

Actualitza els resultats de la GUI.

actualitzarTaula() - Method in class CAN1

Posar les dades dels missatges del problema, GrupMissatges, a la taula de Missatges de la GUI.

alfa - Variable in class GrupMissatges

Variable que indica la qualitat del bus.

alfaLimit - Variable in class CAN1

Constant on indiquem el valor d'alfa crític.

B

barraEines - Variable in class CAN1

barraUs - Variable in class CAN1

bg - Static variable in class ShapesPrint

bitsFixes - Variable in class Missatge

Número de bits fixes que conté el missatge, independents del número de bits que pugui tenir segons la grandària de les dades.

bloqueig(int) - Method in class GrupMissatges

Calcula el temps de bloqueig al qual pot estar sotmés un missatge per un altre de menys prioritat.

botoAjuda - Variable in class CAN1

botoCalcular - Variable in class CAN1

botoCalcularActionPerformed(ActionEvent) - Method in class CAN1

Inici rutina de càlcul del problema, des de botó: posarMissatges(), calcularProblema(), actualitzarTaula(), actualitzarGUI();

botoCalcularEines - Variable in class CAN1

botoCalcularEinesActionPerformed(ActionEvent) - Method in class CAN1

Inici del procés de càlcul, iniciat des del botó.

botoDesar - Variable in class CAN1

botoDesarActionPerformed(ActionEvent) - Method in class CAN1

Inici rutina de desar, des de botó.

botoNou - Variable in class CAN1

botoNouActionPerformed(ActionEvent) - Method in class CAN1

Es crear un nou problema, iniciat des de botó.

botoObrir - Variable in class CAN1

botoObrirActionPerformed(ActionEvent) - Method in class CAN1

Inici rutina d'obrir problema, iniciada des de botó.

button - Static variable in class ShapesPrint

C

calc() - Method in class GrupMissatges

Per cada Missatge es calcula el número màxim de bits de stuff que poden tenir.

calc() - Method in class Missatge

Calcula el número de bits de stuff màxim que pot tenir el missatge.

calcAlfa() - Method in class GrupMissatges

Calcula el factor de qualitat alfa que té els sistema, bus problema.

calcBloqueig() - Method in class GrupMissatges

Es calcula per cada missatge el temps de bloqueig que pot tenir.

calcEspera() - Method in class GrupMissatges

Calcula per cada missatge del problema el temps total màxim que segons el model el missatge pot estar pendent de ser lliurat.

calcNumeroMissatgesProgramables() - Method in class GrupMissatges

Calcula el número de missatges que són programables entre tots els que té el bus-sistema problema.

calcularMenuItem - Variable in class CAN1

calcularProblema() - Method in class CAN1

Calcular el problema, GrupMissatges.

calcUs() - Method in class GrupMissatges

Càlcula el percentatge d'ús (temps) que cada missatge fa del bus.

campFrecuencia - Variable in class CAN1

campFrecuenciaActionPerformed(ActionEvent) - Method in class CAN1

Canviem el valor de la freqüència del bus problema, des de camp de text.

can - Static variable in class Utils

Extensió estàndard de fitxers CAN.

CAN1 - class CAN1.

Genera una interfície gràfica d'usuari amb multiples opcions(*).

CAN1() - Constructor for class CAN1

PROVACrea un GUI form CAN1 i inicialitza la *taula*

columnNames - Variable in class TaulaMissatges

Vector de cadenes de text que conté els títols de les 9 columnes.

contentsMenuItem - Variable in class CAN1

contentsMenuItemActionPerformed(ActionEvent) - Method in class CAN1

S'obre una finestra d'ajuda, amb les instruccions bàsiques d'ús del jCAN1.

ctf - Static variable in class Utils

Extensió optativa pels fitxers del programa.

cutMenuItemActionPerformed(ActionEvent) - Method in class CAN1

Inici rutina de calcular, iniciada des de Menú.

D

dash1 - Static variable in class ShapesPrint

dashed - Static variable in class ShapesPrint

data - Variable in class TaulaMissatges

Matriu d'objectes de 150x9.

deadLine - Variable in class Missatge

El temps màxim que pot esperar un missatge per ser lliurat.

DEBUG - Variable in class TaulaMissatges

Variable amb finalitat de debug.

descripcio - Variable in class GrupMissatges

Descripció o comentari sobre el sistema problema.

drawShapes(Graphics2D) - Method in class ShapesPrint

E

estandardMenuItem1 - Variable in class CAN1

estandardMenuItem1ActionPerformed(ActionEvent) - Method in class CAN1

Seleccionar mode estàndard per Menú.

exitForm(WindowEvent) - Method in class CAN1

Sortir de la aplicació.

exitMenuItem - Variable in class CAN1

exitMenuItemActionPerformed(ActionEvent) - Method in class CAN1

Crida a funció sortida , des de Menú.

extensMenuItem2 - Variable in class CAN1

extensMenuItem2ActionPerformed(ActionEvent) - Method in class CAN1

Seleccionar mode extens per Menú.

F

fg - Static variable in class ShapesPrint

fileMenu - Variable in class CAN1

FiltreFitxersCAN - class FiltreFitxersCAN.

Filtre per fitxers de tipus CAN.

FiltreFitxersCAN() - Constructor for class FiltreFitxersCAN

fitxerActual - Variable in class CAN1

Prova de Fitxer Actual *epe*

fitxerObert - Variable in class CAN1

Si en la sessió de treball ja s'ha especificat un fitxer de treball.

frecuencia - Variable in class GrupMissatges

Freqüència en Kbits per segon a la que funciona el bus.

G

getColumnClass(int) - Method in class TaulaMissatges

Per obtenir el tipus de objecte, la classe, que defineix una columna.

getColumnCount() - Method in class TaulaMissatges

Número de columnes de la taula.

getColumnName(int) - Method in class TaulaMissatges

Per obtenir el nom d'una columna en concret.

getDescription() - Method in class FiltreFitxersCAN

Obtenir la cadena de text descriptiva dels tipus de fitxers que seran mostrats.

getExtension(File) - Static method in class Utils

Mètode public que permet obtenir l'extensió del fitxer passat com argument.

getRowCount() - Method in class TaulaMissatges

Retorna el número de files de la taula.

getValueAt(int, int) - Method in class TaulaMissatges

Per obtenir el valor d'un objecte d'una cel·la en concret.

grup - Variable in class GrupMissatges

Vector d'objectes *Missatge*.

GrupMissatges - class GrupMissatges.

Un sistema basat en un bus CAN ve descrit per un grup de missatges i dades globals del sistema, tots aquests missatges vene en aquesta classe amb un vector d'objectes *Missatge*.

GrupMissatges(int, int, boolean, String) - Constructor for class GrupMissatges

Constructor del problema.

H

helpMenu - Variable in class CAN1

I

initComponents() - Method in class CAN1

This method is called from within the constructor to initialize the form.

isCellEditable(int, int) - Method in class TaulaMissatges

Per conèixer si una cel·la és editable.

J

jButton2ActionPerformed(ActionEvent) - Method in class CAN1

Obre finestra amb ajuda.

jitter - Variable in class Missatge

El temps el qual trigar un node en tindre el missatge preparat per lliurar des de que es produeix la necessitat de ser lliurat.

jLabel1 - Variable in class CAN1

jButton2StateChanged(ChangeEvent) - Method in class CAN1

jScrollPane1 - Variable in class CAN1

jSeparator1 - Variable in class CAN1

jSeparator2 - Variable in class CAN1

jSeparator3 - Variable in class CAN1

jSeparator4 - Variable in class CAN1

jTextField1ActionPerformed(ActionEvent) - Method in class CAN1

L

labelAlfa - Variable in class CAN1

labelNumMissatges - Variable in class CAN1

labelNumMissatgesProgramables - Variable in class CAN1

labelResultat - Variable in class CAN1

labelUs - Variable in class CAN1

M

main(String[]) - Static method in class CAN1

Crear una GUI, la dimensiona i mostra per pantalla.

main(String[]) - Static method in class ShapesPrint

menuBar - Variable in class CAN1

Missatge - class Missatge.

Un Missatge CAN ve caracteritzat per una serie de dades que la classe Missatge conté.

Missatge() - Constructor for class Missatge

modeMenu - Variable in class CAN1

mostrarProblema() - Method in class CAN1

Per Debbug.

mostrarResultats() - Method in class CAN1

Actualitza les cel·les de resultats que té la taula de missatges.

N

name - Variable in class Missatge

Nom descriptiu del missatge.

nouMenuItem - Variable in class CAN1

nouMenuItemActionPerformed(ActionEvent) - Method in class CAN1

Es crear un nou problema, iniciat des de menú.

number - Variable in class Missatge

Número identificador del missatge, marca la prioritat.

numBytes - Variable in class Missatge

El número de bytes de dades que conté el missatge.

numeroMissatges - Variable in class GrupMissatges

Número de missatges de difenent identificador que hi ha al sistema.

numeroMissatgesProgramables - Variable in class GrupMissatges

Número de missatges, diferent identificador, que són programables al sistema plantejat, bus en qüestió,

O

ok - Variable in class Missatge

Indica si el missatge és programable.

openMenuItem - Variable in class CAN1

openMenuItemActionPerformed(ActionEvent) - Method in class CAN1

Inici rutina obrir fitxer problema, des de Menú.

P

paintComponent(Graphics) - Method in class ShapesPrint

panelConclusions - Variable in class CAN1

panelDades - Variable in class CAN1

panelEines - Variable in class CAN1

panelM - Variable in class CAN1

panelPrincipal - Variable in class CAN1

panelResultats - Variable in class CAN1

percentatgeUs - Variable in class GrupMissatges

Percentatge de temps del bus ocupat per missatge.

percentatgeUs - Variable in class Missatge

Percentatge que té el missatge sobre el bus.

percentatgeUsLimit - Variable in class CAN1

Constant on indiquem el tant per cent de càrrega del bus crític.

periode - Variable in class Missatge

El temps que trigar un missatge a tornar a ser generat, en número de períodes de bit.

posarMissatges() - Method in class CAN1

Crea un objecte GrupMissatges, inicialitza el problema.

print(Graphics, PageFormat, int) - Method in class ShapesPrint

printDadesText() - Method in class GrupMissatges

Funció amb finalitats de debug.

printDebugData() - Method in class TaulaMissatges

Mètode per debug.

problema - Variable in class CAN1

En aquest objecte es gaurden totes les dades del problema.

programable - Variable in class GrupMissatges

Si tots els missatges del bus són programables.

R

radioBotoExtensActionPerformed(ActionEvent) - Method in class CAN1

radioButoEstandardActionPerformed(ActionEvent) - Method in class CAN1

rbotoEstandard - Variable in class CAN1

rbotoEstandardActionPerformed(ActionEvent) - Method in class CAN1

Seleccionar mode estàndard per botó de selecció

rbotoExtens - Variable in class CAN1

rbotoExtensActionPerformed(ActionEvent) - Method in class CAN1

Seleccionar mode extens per botó de selecció.

red - Static variable in class ShapesPrint

S

saveAsMenuItem - Variable in class CAN1

saveAsMenuItemActionPerformed(ActionEvent) - Method in class CAN1

Desar com, des de Menú.

saveMenuItem - Variable in class CAN1

saveMenuItemActionPerformed(ActionEvent) - Method in class CAN1

Desar, des de Menú.

sbc - Static variable in class Utils

Extensió dels fitxers amb els quals el programa treballa.

setMissatge(int, int, String, int, int, int, int) - Method in class GrupMissatges

S'encarrega de entrar les dades de cada missatge.

setMissatge(int, String, int, int, int, int, boolean) - Method in class Missatge

Introdueix les dades necessaries per tenir ben definit un missatge.

setNumBytes(TableColumn) - Method in class TaulaMissatges

Crea un JComboBox per la columna del número de Bytes del missatge per tal de tenir només uns valors possibles.

setValueAt(Object, int, int) - Method in class TaulaMissatges

Aquest mètode s'encarrega de posar el valor passat com argument a la cel·la, indicat per un número de columna i un número de fila passats com arguments.

ShapesPrint - class ShapesPrint.

ShapesPrint() - Constructor for class ShapesPrint

sortir(String, String, boolean) - Method in class CAN1

Aquest mètode serveix per a que l'usuari de la GUI sigui preguntat per desar els canvis del treball actual quan pitja sortir, obrir o nou.

standardCAN - Variable in class GrupMissatges

Si el bus treballa en mode estàndard o en mode extens.

stroke - Static variable in class ShapesPrint

stuffleableBits - Variable in class Missatge

Número de bits susceptibles a ser sotmesos a la tècnica de stuff.

stuffBits - Variable in class Missatge

Número màxim de bits de stuff que té el missatge.

T

taula - Variable in class CAN1

Taula on s'introdueixen les dades referents als missatges.

taulaM - Variable in class CAN1

La taula de missatges fa servir aquest model abstracte de taula.

TaulaMissatges - class TaulaMissatges.

Aquesta classe és un model abstracte de taula per extendre la funcionalitat de la classe *JTable*.

TaulaMissatges() - Constructor for class TaulaMissatges

Constructor de la classe, inicialitza la taula als valors per defecte.

tempsBloqueig - Variable in class Missatge

Temps el qual el missatge pot estar aturat abans de ser transmés per culpa de que un missatge de menys prioritat està ocupant el bus.

tempsEspera - Variable in class Missatge

Temps total màxim que segons el model el missatge pot estar aturat per culpa de que el bus està ocupat.

tempsPropagacio - Variable in class Missatge

Temps que trigarà el missatge en propagar-se pel bus.

tempsReal - Variable in class Missatge

Temps total màxim que segons el model el missatge pot estar pendent de ser lliurat.

textAlfa - Variable in class CAN1

textDescripcio - Variable in class CAN1

textFormat - Variable in class CAN1

textFrequencia - Variable in class CAN1

toolsMenu - Variable in class CAN1

U

Utils - class Utils.

Aquesta classe té la llista d'extensions vàildes i té el mètode capaç d'extreure l'extensió d'un fitxer.

Utils() - Constructor for class Utils

W

white - Static variable in class ShapesPrint

wideStroke - Static variable in class ShapesPrint

ANNEX II: CAPL - EinaCAN

VARIABLES GLOBALS

```

variables {
//Temporitzadors
msTimer calibrant;
msTimer fases;
msTimer ledson;
msTimer actiu;
// Varibales per conèixer si el sistema està en activitat de LEDs i fase de
les mesures
int activitat = 0;
int mesures = 0;
int fase;
message CAN1.170 calibracio = {dlc=2, dir=Tx, byte(0)=0xaa, byte(1)=0x04};
// Missatge per cada node on XXX (644, 668, 700, 932, 956):
// oeXXX = Habilitar la sortida de pins del SLIO
// peXXX i neXXX = Permet habilitar la captura de missatge provinents del
circuit temporitzador 555 de forma que només s'agafen el canvi a positiu dels
senyals
// doXXX = Activar el LED
// dfXXX = Desactivar el LED
message CAN1.644 oe644 = {dlc=3, dir=Tx, byte(0)=0x04, byte(1)=0xff,
byte(2)=0xff};
message CAN1.644 pe644 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x20};
message CAN1.644 ne644 = {dlc=3, dir=Tx, byte(0)=0x02, byte(1)=0x00,
byte(2)=0x00};
message CAN1.644 no644 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x00};
message CAN1.644 do644 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x10};
message CAN1.644 df644 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x00};
message CAN1.668 oe668 = {dlc=3, dir=Tx, byte(0)=0x04, byte(1)=0xff,
byte(2)=0xff};
message CAN1.668 pe668 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x20};
message CAN1.668 ne668 = {dlc=3, dir=Tx, byte(0)=0x02, byte(1)=0x00,
byte(2)=0x00};
message CAN1.668 no668 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x00};
message CAN1.668 do668 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x10};
message CAN1.668 df668 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x00};
message CAN1.700 oe700 = {dlc=3, dir=Tx, byte(0)=0x04, byte(1)=0xff,
byte(2)=0xff};
message CAN1.700 pe700 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x20};
message CAN1.700 ne700 = {dlc=3, dir=Tx, byte(0)=0x02, byte(1)=0x00,
byte(2)=0x00};
message CAN1.700 no700 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x00};
message CAN1.700 do700 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x10};
message CAN1.700 df700 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x00};
message CAN1.932 oe932 = {dlc=3, dir=Tx, byte(0)=0x04, byte(1)=0xff,
byte(2)=0xff};
message CAN1.932 pe932 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x20};

```



```

message CAN1.932 ne932 = {dlc=3, dir=Tx, byte(0)=0x02, byte(1)=0x00,
byte(2)=0x00};
message CAN1.932 no932 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x00};
message CAN1.932 do932 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x10};
message CAN1.932 df932 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x00};
message CAN1.956 oe956 = {dlc=3, dir=Tx, byte(0)=0x04, byte(1)=0xff,
byte(2)=0xff};
message CAN1.956 pe956 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x20};
message CAN1.956 ne956 = {dlc=3, dir=Tx, byte(0)=0x02, byte(1)=0x00,
byte(2)=0x00};
message CAN1.956 no956 = {dlc=3, dir=Tx, byte(0)=0x01, byte(1)=0x00,
byte(2)=0x00};
message CAN1.956 do956 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x10};
message CAN1.956 df956 = {dlc=3, dir=Tx, byte(0)=0x03, byte(1)=0x00,
byte(2)=0x00};
// Missatge comentats, per iniciar proves d'alta càrrega
//message CAN1.644 ie = {dlc=3, dir=Tx, byte(0)=0x00, byte(1)=0x ,
byte(2)=0x };
//message CAN2.700 LwMsg = {dlc=1, dir=Tx, byte(0)=0xFF};
//message CAN2.10 HiMsg = {dlc=8, dir=Tx, byte(0)=0xaa, byte(1)=0xaa,
byte(2)=0xaa, byte(3)=0xaa, byte(4)=0xaa, byte(5)=0xaa, byte(6)=0xaa,
byte(7)=0xff};
}

```

SYSTEM

```

on start
{
//rutina iniciada al arrancar el CANalyzer
int i;
fase = 0;
resetCan();
// Missatges d'opcions disponibles
write("Entorn Laboratori.");
write("1-5 :      Iniciar nodes.");
write("q-w-e-r-t :  LEDs nodes");
write("z-x:      Encednre apagar LEDs");
write("c:      Indicador d'activitat");
write("j :      Iniciar mesures");
//Configurar el bus
for(i=1; i<=4 ; i++) output(calibracio);
setTimer(calibrant,50);
}

```


CAN-MESSAGE

//Respostes a cada missatge, un cop iniciat els bus cada node es dona a conèixer i el CAN1 respon amb aquests missatges de configuració.

```
on message 645
{
/*Aquest procés s'ocupa de configurar el node corresponent,
un cop rebut el missatge de calibració*/
    if(THIS.dir!=Rx) return;
    if(THIS.byte(0)==0xC0)
    {
        configura644();
    }
}

on message 669
{
/*Aquest procés s'ocupa de configurar el node corresponent,
un cop rebut el missatge de calibració*/
    if(THIS.dir!=Rx) return;
    if(THIS.byte(0)==0xC0)
    {
        configura668();
    }
}

on message 701
{
/*Aquest procés s'ocupa de configurar el node corresponent,
un cop rebut el missatge de calibració*/
    if(THIS.dir!=Rx) return;
    if(THIS.byte(0)==0xC0)
    {
        configura700();
    }
}

on message 933
{
/*Aquest procés s'ocupa de configurar el node corresponent,
un cop rebut el missatge de calibració*/
    if(THIS.dir!=Rx) return;
    if(THIS.byte(0)==0xC0)
    {
        configura932();
    }
}

on message 957
{
/*Aquest procés s'ocupa de configurar el node corresponent,
un cop rebut el missatge de calibració*/
    if(THIS.dir!=Rx) return;
    if(THIS.byte(0)==0xC0)
    {
        configura956();
    }
}
```


TIMER

```
// Timer per mostrar activitat al bus amb els LEDs
on timer actiu
{
switch(activitat){
    case(0):
        ledsoff();
        break;
    case(1):
        ledon644();
        setTimer(actiu,20);
        activitat = 2;
        break;
    case(2):
        ledon956();
        setTimer(actiu,20);
        activitat = 3;
        break;
    case(3):
        ledon668();
        setTimer(actiu,20);
        activitat = 4;
        break;
    case(4):
        ledon932();
        setTimer(actiu,20);
        activitat = 5;
        break;
    case(5):
        ledon700();
        setTimer(actiu,20);
        activitat = 6;
        break;
    case(6):
        ledoff644();
        setTimer(actiu,20);
        activitat = 7;
        break;
    case(7):
        ledoff956();
        setTimer(actiu,20);
        activitat = 8;
        break;
    case(8):
        ledoff668();
        setTimer(actiu,20);
        activitat = 9;
        break;
    case(9):
        ledoff932();
        setTimer(actiu,20);
        activitat = 10;
        break;
    case(10):
        ledoff700();
        setTimer(actiu,20);
        activitat = 1;
        break;
    case(11):
        ledson();
        activitat = 0;
        break;
}
```



```
on timer calibrant
{
  // Aquests procediments s'encarrega d'enviar un missatge de calibració cada 40
  ms. Aquest valor pot variar segons les necessitats de càrrega que es vulgui
  simular al bus.
    output(calibracio);
    setTimer(calibrant,40);
}

on timer fases
// Timer per iniciar les mesures de forma ordenada. Permet fer estudi en el
// temps i càrrega.
{
  output(calibracio);
  switch(fase){
    case(0):
      write("FASE 0: 956");
      on956();
      setTimer(fases,20000);
      break;
    case(1):
      //write("FASE 1: 956");
      //on956();
      setTimer(fases,500);
      break;
    case(2):
      write("FASE 2: 932");
      on932();
      setTimer(fases,500);
      break;
    case(3):
      write("FASE 3: 700");
      on700();
      setTimer(fases,500);
      break;
    case(4):
      write("FASE 4: 668");
      on668();
      setTimer(fases,500);
      break;
    case(5):
      write("FASE 5: 644");
      on644();
      setTimer(fases,500);
      break;
    case(6):
      write("SISTEMA A PLENA CARRGA");
      //write("FASE 5");
      //output(no644);
      break;
  }
  fase++;
}

on timer ledson
{
  ledson();
}
```


KEYBOARD

```
//Accions dutes a càrrec per cada tecla

on key 'l'
{
    output(pe644);
    output(ne644);
}

on key 'c'
{
    if (activitat == 0) setTimer(actiu,1);
    if (activitat == 0) activitat = 1; else activitat = 0;
}

on key 'q'
{
    ledon644();
}

on key 'z'
{
    //Per fer leds on
    ledon644();
    ledon668();
    ledon700();
    ledon932();
    ledon956();
}

on key 'x'
{
    //Per fer leds off
    ledoff644();
    ledoff668();
    ledoff700();
    ledoff932();
    ledoff956();
}

on key 'w'
{
    ledon668();
}

on key 'j'
{
    setTimer(fases,1000);
    setTimer(ledson, 1);
    activitat = 11;
    mesures = 1;
}
```


FUNCTION

```
configura644 ()
{
    output(oe644);
}

ledoff644 ()
{
    output(df644);
}

ledon700 ()
{
    output(do700);
}

ledsoff ()
{
    ledoff644();
    ledoff668();
    ledoff700();
    ledoff932();
    ledoff956();
}

on932 ()
{
    output(pe932);
    output(ne932);
}
```