

Capitolul 7

Studiu de caz: protocolul CAN (Controller Area Network)

Pe măsură ce dispozitivele electronice de control din automobile au devenit tot mai complexe, cu numeroase legături între ele, utilizarea de legături cablate pentru interconectarea lor a devenit scumpă și mult prea complicată. În condițiile în care dispozitivele de control convenționale dispun de posibilități limitate de interconectare, soluția este utilizarea unei rețele seriale. De aceea, firma Bosch a dezvoltat rețeaua CAN (Controller Area Network), care a fost apoi standardizată (ISO 11898) și a fost implementată în hardware de către mai mulți producători de dispozitive semiconductoare.

CAN conectează noduri echivalente din punct de vedere logic (controlere, senzori, elemente de execuție) printr-o magistrală serială. Protocolul corespunde nivelului legăturii de date în modelul de referință ISO OSI și este capabil să detecteze și să corecteze erorile de comunicație cauzate de perturbațiile de natură electromagnetică. Datorită cerințelor similare ale rețelelor de control industriale (cost scăzut și imunitate la perturbații), aplicațiile CAN au trecut de la mașinile firmei Mercedes-Benz și ale fabricanților de vehicule comerciale americani la realizarea echipamentelor de navigație și pentru agricultură, CAN fiind de asemenea utilizat în aparatura medicală, mașini textile, mașini speciale și echipamente de ridicat. Magistrala serială a CAN permite utilizarea acestuia pentru interconectarea dispozitivelor cu inteligență înglobată, precum și a senzorilor și a elementelor de execuție din instalațiile industriale.

Industria de construcții de mașini textile este una din primele care au utilizat rețeaua CAN. Încă din 1990, unul dintre producători și-a echipat războaiele de țesut cu sisteme de conducere modulare, care comunică în timp real prin intermediul rețelelor CAN.

7.1. Funcționarea rețelei CAN

Atunci când un nod transmite date, acestea sunt recepționate de toate celelalte noduri. Un nod poate constata că mesajul nu este relevant și îl poate ignora.

Tipul datei transmise – turația motorului, temperatura uleiului etc. – este codificat printr-un identificator format din 11 biți amplasat la începutul mesajului. Mai mult, identificatorul definește și prioritatea mesajului. O astfel de schemă de comunicație prin mesaje se numește *adresare bazată pe conținut*.

Fiecare identificator de 11 biți este unic în cadrul rețelei; nu există două noduri care să aibă același identificator. De asemenea, două noduri diferite nu pot transmite

mesaje folosind același identificator. Aceste lucruri sunt importante pentru acordarea accesului la mediu în condițiile în care mai multe noduri concurează pentru a-l obține.

Dacă unitatea centrală a unui nod dorește să transmită un mesaj unuia sau mai multor noduri, atunci înaintează data și identificatorul acesteia către cipul CAN asociat. Acesta construiește mesajul și îl trimite pe linie atunci când nodul primește dreptul de acces la mediu.

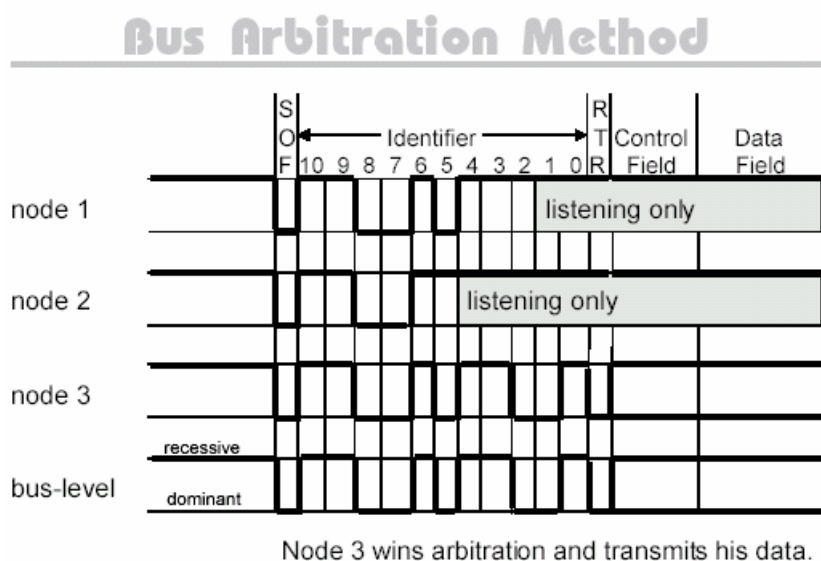
Când are loc acest lucru, toate celelalte noduri devin receptoare. Fiecare nod receptor realizează un test pentru a vedea dacă data este relevantă sau nu. Dacă da, data este acceptată, altfel este ignorată.

Datorită schemei de adresare prin conținut, protocolul CAN nu necesită ca nodurile să aibă adrese fizice. Noi noduri receptoare pot fi introduse în rețea fără ca aceasta să atragă vreo modificare hardware sau software pentru nodurile deja existente.

7.1.1. Arbitrarea accesului la mediu

Pentru ca datele să fie prelucrate în timp real, ele trebuie transmise cu viteză mare. Aceasta necesită un mediu de comunicație adecvat și o schemă rapidă de atribuire a accesului la mediu atunci când mai multe noduri doresc să transmită date simultan. În aplicațiile de timp real, urgența cu care trebui transmise mesajele poate să difere destul de mult. O variabilă care se modifică rapid, cum ar fi de exemplu încărcarea motorului, trebuie să fie transmisă mai frecvent și de aceea cu o mai mică întârziere decât alți parametri, cum ar fi temperatura motorului, care se modifică relativ lent.

Prioritatea cu care un mesaj este transmis este codificată în identificatorul de 11 biți. Identificatorul cu valoarea binară minimă are prioritatea cea mai mare. Aceste priorități (coduri de identificare) sunt specificate în faza de proiectare și sunt fixe, nu pot fi modificate dinamic. Conflictele de acces la magistrală sunt rezolvate printr-o arbitrare la nivel de bit (tehnica CSMA/BA) în cadrul câmpului de identificare al nodurilor.



© CIA

Protocolul CAN este eficient pentru că mediul de comunicație este utilizat numai de către nodurile care au ceva de transmis. Aceste cereri sunt servite în ordinea importanței mesajelor pe ansamblul sistemului. Aceasta se dovedește a fi deosebit de

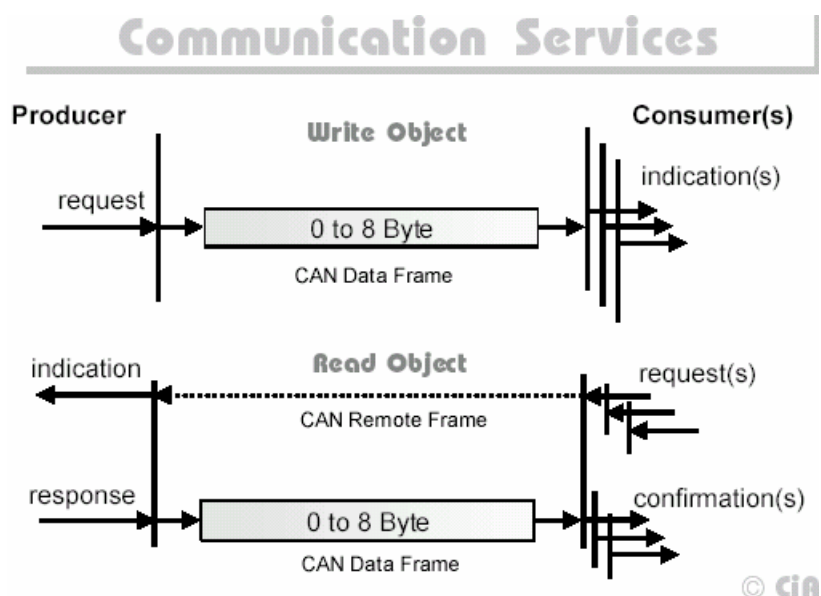
avantajos în condiții de încărcare mare a rețelei. Întrucât accesul la magistrală este arbitrat pe baza priorității mesajelor, se garantează un timp de răspuns mic în sistemele de timp real.

Pentru a depăși problema fiabilității reduse a unui nod master, protocolul implementează un control descentralizat al magistralei.

7.1.2. Servicii de comunicație

Protocolul CAN furnizează două servicii de comunicație:

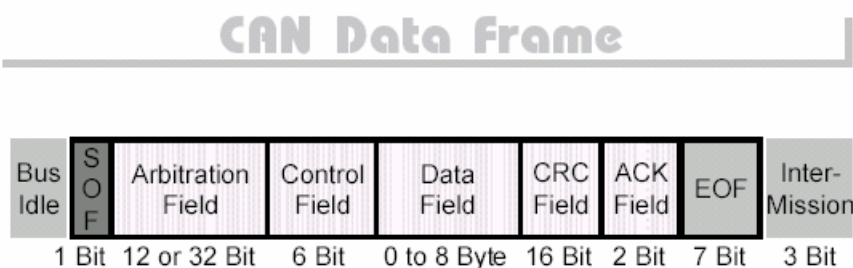
- unul clasic, Write Object - în care producătorul transmite un cadru către unul sau mai mulți consumatori. Aceștia vor accepta cadrul dacă sunt interesați de informația recepționată.
- altul special, Read Object – care asigură cererea unui mesaj specific de către unul sau mai mulți consumatori. Acesta va transmite un așa zis cadru de cerere (Remote Frame). Un mesaj de tip cerere nu conține biți de date. Nodul care deține informația cerută va transmite cadrul corespunzător.



7.1.3. Formatul mesajelor CAN

Un cadru CAN este format din mai multe câmpuri de biți. CAN suportă două formate ale mesajelor. Diferența dintre ele este dată de lungimea câmpului identificator, care este de 11 biți în formatul standard (CAN 2.0a), respectiv de 29 de biți în formatul extins (CAN 2.0b).

Formatul unui cadru de date (Data Frame)



Un cadru de date în formatul standard începe cu un bit dominant de start denumit Start Of Frame (SOF) pentru sincronizarea tuturor nodurilor. Bitul SOF este urmat de un câmp de arbitrare care reflectă conținutul și prioritatea mesajului. Bitul SOF este urmat de un câmp de arbitrare care reflectă conținutul și prioritatea mesajului. Următorul câmp este câmpul de control, care specifică în principal numărul de octeți de date conținut de mesaj în câmpul de date (maxim 8). Câmpul CRC este folosit pentru a detecta posibile erori de transmisie, fiind format din 15 biți urmați de 1 bit delimitator nedominant. În timpul câmpului de confirmare (ACK) nodul transmițător transmite un bit nedominant. El este forțat în 0 (valoarea dominantă) de către acele noduri receptoare care au recepționat corect mesajul în acel moment. Astfel, transmițătorul se asigură că cel puțin un receptor a primit corect mesajul său. Mesajele sunt confirmate de către receptoare înainte de testul de acceptare.

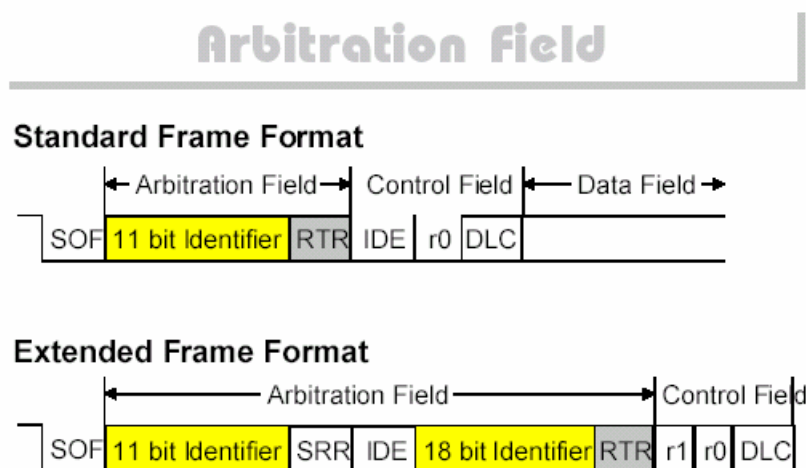
Sfârșitul mesajului este indicat de către câmpul de sfârșit de cadru. Între două mesaje succesive trebuie să existe un număr de 3 biți nedominanți. Dacă nu există mesaje de transmis, linia rămâne inactivă.

Bitul SOF este folosit pentru resincronizarea hardware a receptorului la începutul unui nou cadru. Acesta poate să apară:

- în timp ce linia este liberă;
- în cazul în care transmisia anterioară a fost suspendată;
- după ultimul bit din spațiul minim dintre cadre.

Câmpul de arbitrare este format din codul de identificare de bază de 11 biți (formatul standard) urmat de bitul RTR (Remote Transmission Request). Bitul RTR precizează că mesajul este de tip cerere de date de la un alt nod. În cadrele de tip Data Frame bitul RTR=0 (dominant), iar în cele de tip Remote Frame RTR=1 (nedominant).

În formatul extins, codul de identificare este divizat în două câmpuri: câmpul de identificare de bază (11 biți) și de cel extins (18 biți), separate de alți doi biți SRR (Substitute Remote Request) și IDE (Identifier Extension Bit).



Bitul SRR înlocuiește bitul RTR din formatul standard și are valoarea 1 (nedominantă). Dacă bitul este corupt și transmis dominant, receptorul o va ignora. Aceasta va fi luată totuși în considerare pentru inserția de biți și pentru arbitrare. Întrucât bitul SRR este recepționat înainte de IDE, un receptor nu poate decide imediat

dacă primește un bit RTR sau unul SRR. Aceasta înseamnă ca numai bitul IDE stabilește dacă cadrul este unul standard (IDE=0) sau unul extins (IDE=1).

Control Field

RTR	IDE/r1	r0	DLC3	DLC2	DLC1	DLC0	Data/CRC
-----	--------	----	------	------	------	------	----------

No. of Data Bytes	Data Length Code (DLC)			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d/r	d/r	d/r

Câmpul de control este similar pentru cele două formate. În formatul standard el include biții IDE=0, 4 biți DLC (Data Length Code) și un bit rezervat r0=0. În formatul extins, câmpul de control include cei 4 biți DLC și doi biți rezervați, r1=0 și r0=0.

Câmpul de date poate avea minim 0 și maxim 8 octeți. În unele aplicații este suficientă transmiterea codului de identificare pentru a semnaliza apariția unui eveniment, fără a fi nevoie de octeți de date.

Data Field

Byte 1 Byte 2 Byte 3 Byte 4 Byte 5 Byte 6 Byte 7 Byte 8

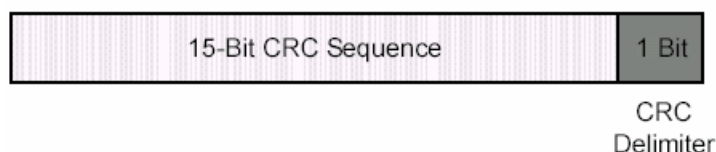
min. length of Data Field = 0 Byte

Byte 1 Byte 2 Byte 3 Byte 4 Byte 5 Byte 6 Byte 7 Byte 8

max. length of Data Field = 8 Byte

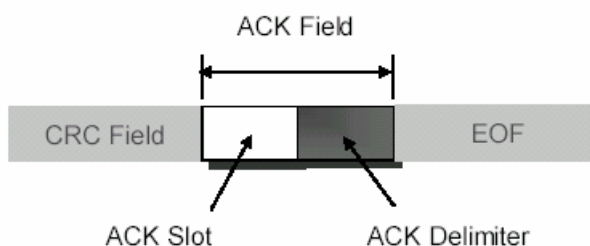
Câmpul de date este urmat de un câmp CRC de 15 biți, urmat de un bit CRC nedominant. Secvența de 15 biți este generată cu ajutorul unui polinom generator adecvat pentru cadre cu un număr de cel mult 127 de biți ($x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$ - codul BCH). Codul CRC are o distanță Hamming egală cu 6, ceea ce înseamnă că pot fi detectate un număr de 5 erori distribuite aleator în câmpurile SOF, de arbitrare, de control și de date. În plus, pot fi detectate erori succesive de până la 15 biți.

CRC Field



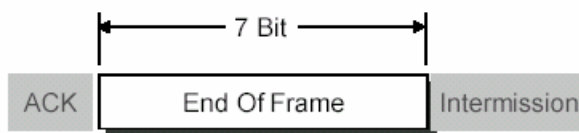
Câmpul de confirmare (ACK – Acknowledgement) este format dintr-un slot ACK (un bit) și dintr-un bit ACK delimiter. Bitul în slotul ACK este pus de către transmițător pe valoarea nedominantă (1 logic). El este forțat în 0 (valoarea dominantă) de către acele noduri receptoare care au recepționat corect mesajul în acel moment. Astfel, transmițătorul se asigură că cel puțin un receptor a primit corect mesajul său. Mesajele sunt confirmate de către receptoare înainte de testul de acceptare.

Acknowledge Field



Sfârșitul fiecărui cadru (End of Frame) este delimitat printr-o secvență de 7 biți nedominanți. Acest indicator a fost introdus deoarece un cadru de eroare cauzat de o eroare CRC trebuie transmis pe durata unui cadru de date sau de tip cerere de date.

End of Frame (EOF)



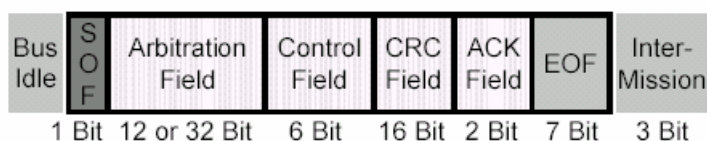
Între două mesaje succesive trebuie să existe un număr minim de perioade de lungimea unui bit. Dacă nu există mesaje de transmis, linia rămâne inactivă.

Formatul unui mesaj de tip cerere de date (Remote Frame)

Un nod destinație poate cere date de la un nod sursă transmitând un mesaj de tip cerere de date, cu un identificator care coincide cu identificatorul cadrului de datei cerute.

Există două diferențe între un cadru de date și unul de cerere date. În primul rând, bitul RTR este transmis ca un bit dominant în cadrul de date și ca unul nedominant în cadrul de cerere. În al doilea rând, în cadrul de cerere lipsește câmpul de date. În cazul în care se transmite simultan un cadru de date și unul de cerere cu același identificator, cadrul de date câștigă arbitrarea datorită bitului RTR dominant. Astfel, nodul care a transmis cadrul de cerere primește data imediat.

CAN Remote Frame

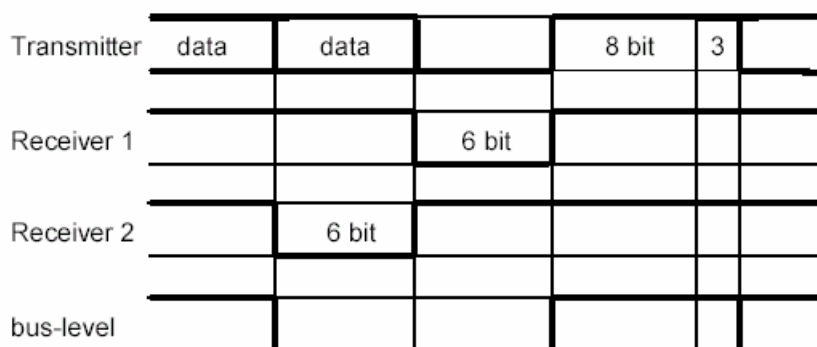


7.1.4. Detectarea erorilor în protocolul CAN

Erorile locale pot să apară la nivelul fiecărui nod din mai multe motive, cum ar fi defazajul punctelor de eșantionare sau dispersia semnalului în timpul propagării pe liniile magistralei. Pentru a garanta consistența datelor la nivelul întregii rețele, erorile locale (detectate de unul sau mai multe noduri) trebuie făcute cunoscute tuturor celorlalte noduri, procedeu cunoscut sub numele de globalizare a erorilor).

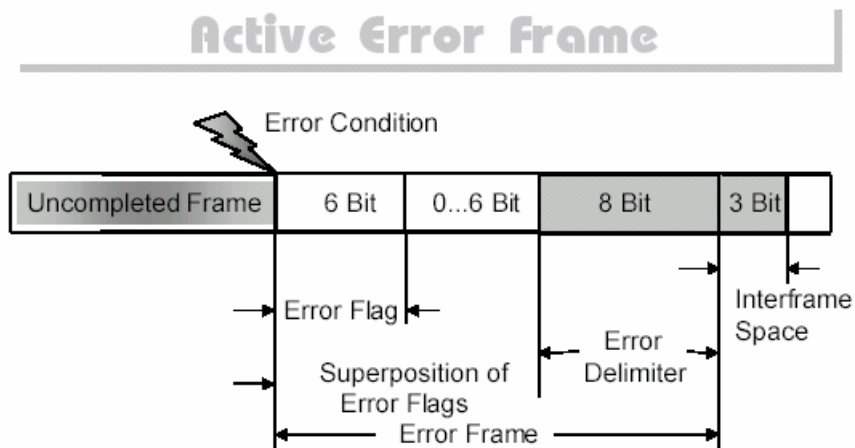
Pentru a indica apariția erorilor locale se folosește regula de inserare a biților, prin transmiterea unui flag de eroare format din 6 biți având aceeași polaritate (dominantă). Astfel, fiecare nod poate detecta această eroare globală de inserare și reacționează prin transmiterea de către fiecare a unui flag de eroare.

Globalization of Local Errors



The Receiver 2 detects an error and makes it public to the other nodes.

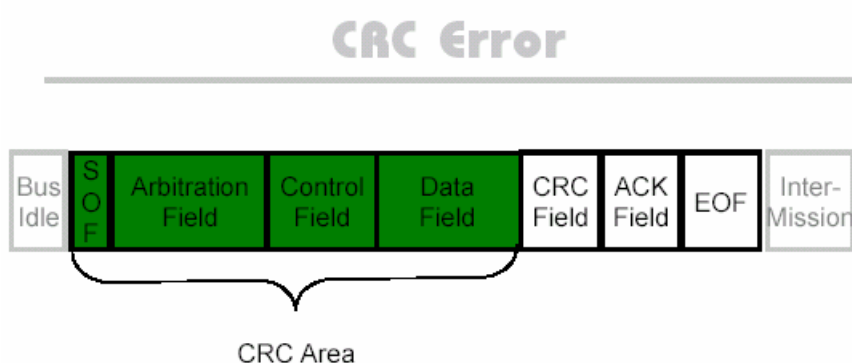
Receptorul 2 detectează o eroare locală și transmite un flag de eroare. Al 6-lea bit al flagului încalcă regula de inserare de biți și eroarea este recunoscută de toate celelalte noduri (receptorul 1), care transmit (simultan) propriile lor flaguri de eroare. După un câmp delimitator al cadrului de eroare de 8 biți și spațiul minim dintre cadre de 3 biți, transmițătorul încearcă retransmisia mesajului eronat.



Spre deosebire de alte rețele de tip magistrală, protocolul CAN nu folosește mesaje de confirmare. Există 5 mecanisme de detectare a erorilor:

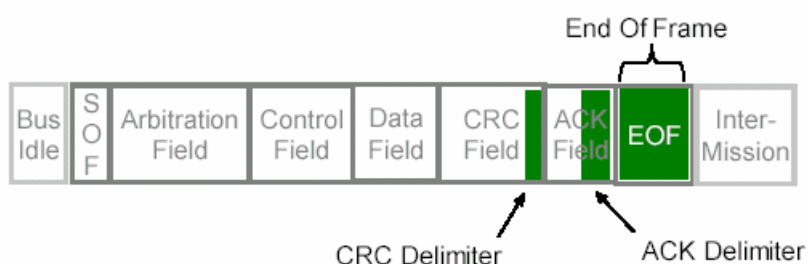
- trei mecanisme la nivel de mesaj:
 - codul CRC (Cyclic Redundancy Check);
 - verificarea încadrării (frame check);
 - bitul de confirmare (ACK bit);
- două mecanisme la nivel de bit:
 - ascultarea liniei (Bus monitoring);
 - inserția de biți (Bit stuffing).

Informația conținută de un cadru poate fi verificată folosind adăugarea de către transmițător, la sfârșitul fiecărui cadru, a unor biți de verificare CRC, care la receptor sunt comparați cu biții recalculați după aceeași regulă. Dacă nu coincid, atunci a apărut o eroare.



Verificarea încadrării (Frame check) presupune verificarea structurii cadrului la transmisie pe baza formatului fix și a lungimii cadrului. Erorile detectate sunt erori de format sau de încadrare.

Form Error



Cadrele recepționate sunt confirmate printr-un bit de confirmare pe nivel dominant. Dacă transmițătorul nu detectează nici o confirmare, acest fapt este interpretat drept un semnal precum că a apărut o eroare detectată numai de către nodurile receptoare, că bitul ACK a fost perturbat sau nu există nici un receptor al cadrului.

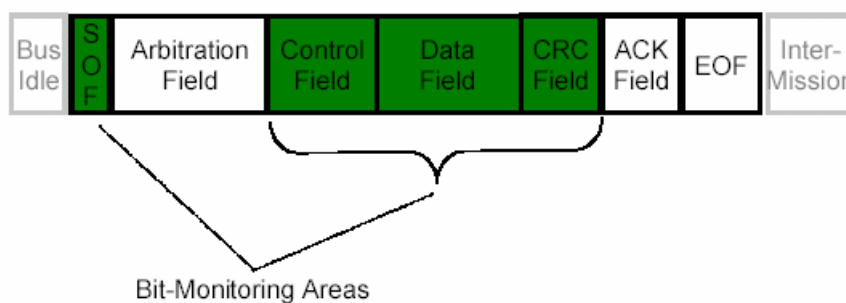
Acknowledgement Error



An ACK Error has to be detected by a Transmitter whenever it does not monitor a dominant bit during ACK Slot

Un nod poate monitoriza propriul său semnal în timpul transmisiei. Astfel, nodul transmițător ascultă nivelul prezent pe magistrală și detectează diferența care poate să apară între bitul transmis și cel recepționat. Aceasta permite detectarea erorilor globale precum și a celor la nivelul transmițătorului.

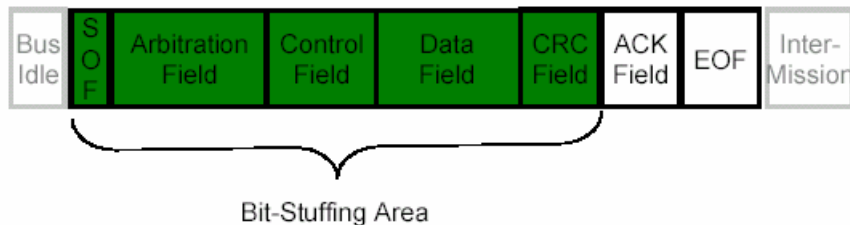
Bit Error



Pentru a obține o eficiență maximă, protocolul CAN folosește reprezentarea NRZ în codificarea biților. Dacă sunt prea mulți biți consecutivi cu aceeași valoare, se poate pierde sincronizarea dintre transmițător și receptor. Pentru a evita acest lucru, sunt generate fronturi de sincronizare prin metoda inserției de biți. După 5 biți consecutivi

de același tip, transmițătorul inserează automat un bit de valoare complementară. La recepție se verifică această regulă și dacă se detectează 6 biți consecutivi de aceeași valoare, atunci se consideră că a apărut o eroare.

Bit-Stuffing Error

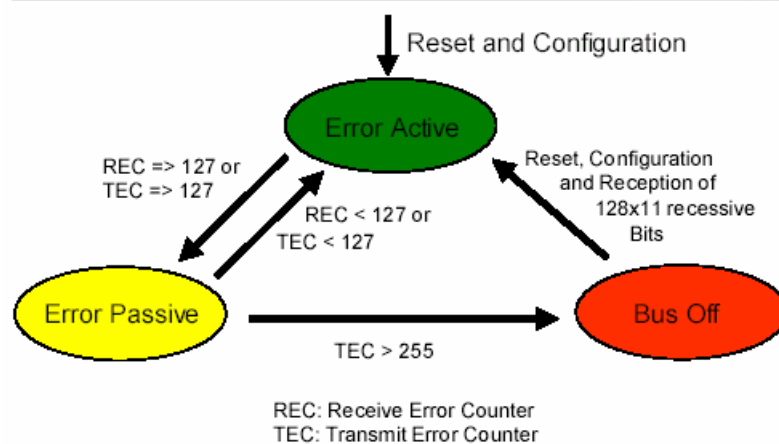


Dacă cel puțin un nod receptor descoperă una sau mai multe erori printr-unul dintre mecanismele descrise mai sus, atunci transmite un flag de eroare care atrage abandonarea transmisiei mesajului. Aceasta evită acceptarea mesajului de către alte stații și asigură astfel păstrarea consistenței datelor în cadrul rețelei.

După ce transmisia unui mesaj eronat a fost abandonată, transmițătorul încearcă în mod automat retransmisia.

Oricât de eficiente ar fi metodele de verificare descrise mai sus, un nod defect poate face ca toate mesajele (inclusiv cele corecte) să fie abandonate, blocând astfel magistrala dacă nu sunt luate măsuri de auto-monitorizare. De aceea, protocolul CAN furnizează un mecanism prin care se poate face distincția între erori ocazionale și erori sistematice sau permanente care permite localizarea defectelor la nivelul nodurilor. Acest lucru se realizează prin analiza statistică a erorilor apărute pentru a detecta defectarea propriului nod și pentru a face posibilă intrarea într-un mod de operare în care restul rețelei să nu fie afectată negativ. Aceasta poate duce până la auto-deconectarea nodului defect pentru a preveni abandonarea mesajelor considerate în mod greșit ca fiind eronate.

Error States of CAN Node



Se utilizează două numărătoare de erori: unul pentru transmisie (TEC) și altul pentru recepție (REC). Cele două numărătoare sunt incrementate la detectarea erorilor în timpul transmisiei, respectiv recepției cadrelor eronate și sunt decrementate la transmisia, respectiv recepția corectă a cadrelor.

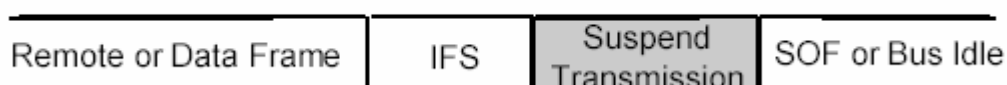
La resetare, starea inițială este Error Active, când controlerul poate transmite flaguri de eroare ACTIVE ERROR FLAG. Dacă erorile se acumulează, controlerul CAN trece în starea Error Pasive, când controlerul poate transmite flaguri de eroare PASSIVE ERROR FLAG.

Un nod receptor în starea Error Pasive nu poate întrerupe transmisia cadrului deoarece transmite un flag de eroare pasiv, format din biți recesivi. Un nod transmițător își poate întrerupe transmisia pentru a transmite flagul de eroare pasiv.

Dacă unul dintre nodurile receptoare este în starea Error Pasive, atunci nu se mai poate garanta consistența datelor din rețea, întrucât o eroare locală detectată doar de noduri pasive nu mai poate fi globalizată.

Pentru a evita blocarea magistralei de către un nod care încearcă fără succes, în mod repetat, să transmită un mesaj cu prioritate ridicată, s-a introdus o întârziere după spațiul obligatoriu de 3 biți dintre cadre, timp în care alte noduri pot inițializa propriile transmisii.

INITIATED BY ERROR PASSIVE NODE



IFS (Interframe Space) = 3 bit
Suspend Transmission = 8 bit

La defectarea controlerului CAN sau dacă erorile cresc semnificativ, se trece în starea BUSOFF, de deconectare. Ieșirea din această stare este posibilă numai printr-o resetare hardware sau software, urmată de reconfigurare și de detectarea unei secvențe de 128×11 biți recesivi pe magistrala CAN înainte de a putea transmite (numai după resetarea software). Există un set de 12 reguli care descriu modul de actualizare a numărătoarelor de erori în funcție de eroarea apărută și de contextul acesteia.

Introducerea sistemelor de siguranță în automobile a adus cu sine cerințe de fiabilitate ridicată a transmisiei datelor pentru a preveni apariția situațiilor în care este pusă în pericol siguranța pasagerilor ca urmare a schimbului de date pe toată durata de viață a vehiculului.

Acest scop este atins dacă fiabilitatea transmisiei datelor este suficient de mare, sau altfel spus probabilitatea de apariție a unei erori ne-detectate este suficient de mică. În cazul magistralei, fiabilitatea poate fi înțeleasă drept capacitatea nodurilor de a identifica datele corupte prin detectarea erorilor de transmisie.

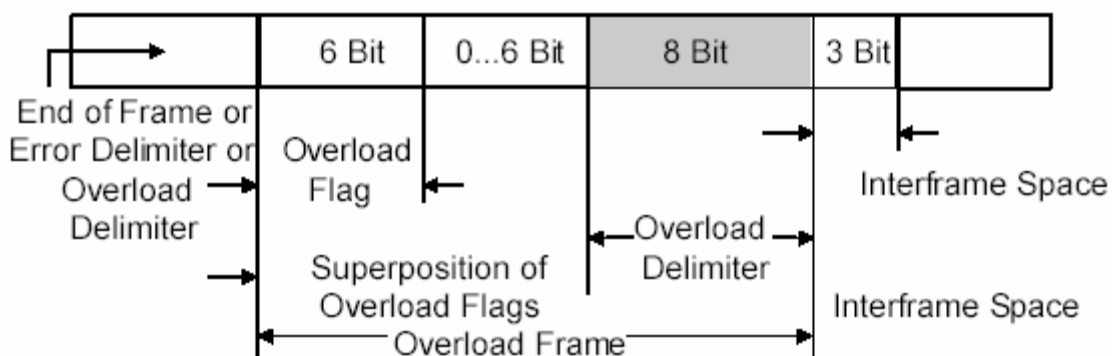
Probabilitatea de nedetectare a unui cadru standard eronat este mai mică de $4,7 \times 10^{-11} \times \text{rata erorii}$. Exemplu: Dacă apare 1 bit eronat la fiecare 0,7 secunde, la o funcționare timp de 8 ore/zi, 365 zile/an la viteza de 500Kbps, o eroare nedetectată apare în medie o dată la 1000 de ani.

7.1.5. Întârzierea transmisiei de cadre de date sau de cerere

Un cadru de supraîncărcare (Overload Frame) este asemănător cu unul de eroare și poate fi transmis în una din următoarele 3 situații:

1. Datorită unor condiții interne, receptorul nu este pregătit să înceapă primirea cadrului următor.
2. Pe durata intervalului de delimitare între cadre (3 biți pe nivel recesiv) este detectat un nivel dominant în unul din primii doi biți.
3. Un mesaj este valid pentru receptor chiar dacă ultimul bit din EOF este găsit dominant. Acest lucru nu este considerat o eroare. Totuși, formatul fix al cadrului nu este respectat și este posibil să fi apărut o desincronizare, ceea ce impune o reacție.

Practic, reacția este aceeași ca și în cazul în care ultimul bit al delimitatorului unui cadru de eroare sau de supraîncărcare este recepționat pe nivel dominant: transmiterea unui cadru de supraîncărcare, care să întârzie transmisia următorului cadru de date sau de cerere.



Spre deosebire de un cadru de eroare, transmisia unui cadru de supraîncărcare nu afectează numărătoarele de erori și nu determină retransmisia mesajului. Numărul maxim de cadre de supraîncărcare consecutive care pot fi transmise de un nod este limitat la 2.

Receptorul consideră mesajul valid cu un bit înaintea transmițătorului. În cazul în care transmițătorul detectează un nivel dominant în ultimul bit din EOF, atunci va retransmite mesajul. Astfel, este posibil ca același mesaj să fie primit și validat de două ori de către nodurile receptoare.

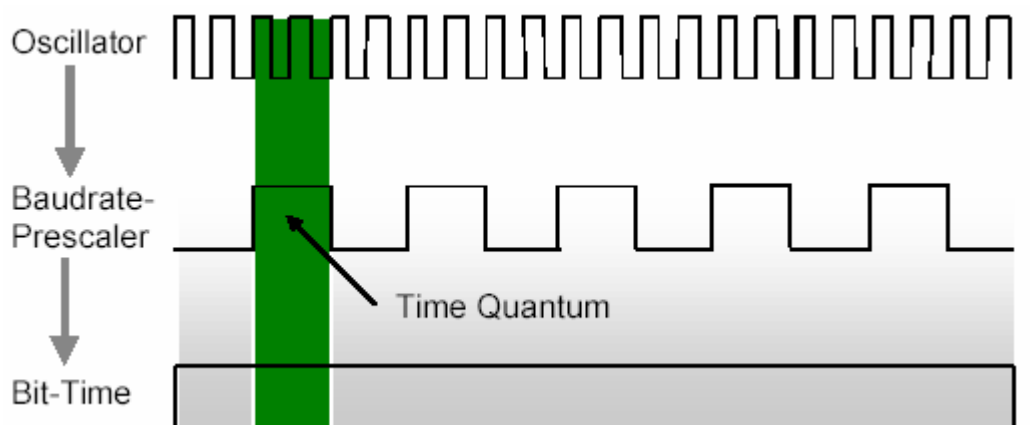
Datorită acestui fenomen de dublare a mesajelor, nu se recomandă utilizarea de mesaje care să comande bascularea unor stări și nici a mesajelor care poartă valori relative care să se adauge la valori absolute cunoscute de către receptor.

Dacă un nod transmițător are un mesaj care așteaptă să fie transmis și detectează în cel de-al 3-lea bit al delimitatorului dintre cadre un bit dominant, atunci considera că a apărut un bit SOF și începe transmisia mesajului direct cu primul bit al identificatorului, fără a mai transmite SOF și fără a mai activa receptorul.

Printre rețelele industriale de control care au la bază protocolul CAN se numără CANOpen (CiA), SDS (Honeywell) și DeviceNet (Allen Bradley). Aria largă de aplicare a acestei rețele face din CAN una dintre interfețele seriale standard integrate în microcontrolere.

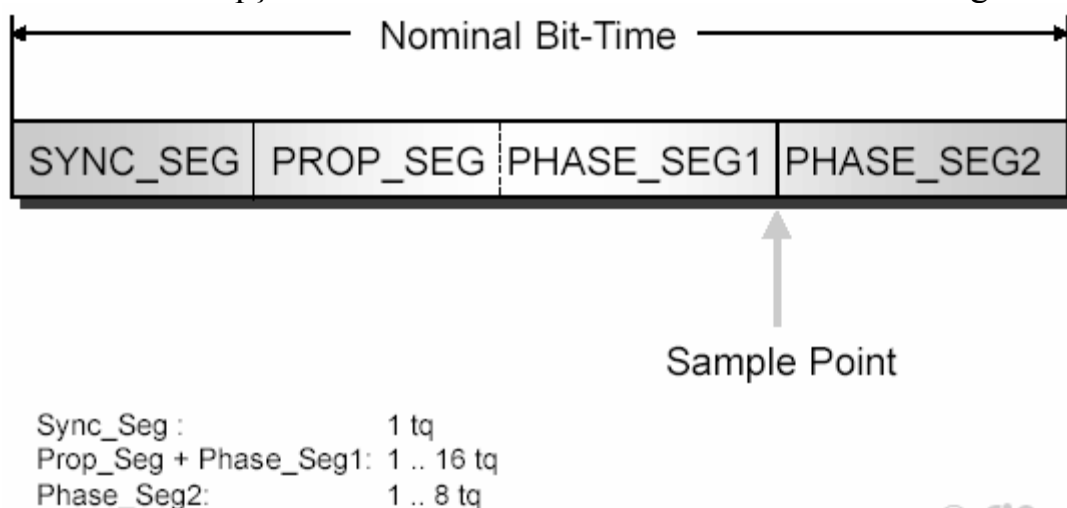
7.2. Protocolul CAN la nivel fizic

Rata nominală de transfer reprezintă numărul de biți pe secundă transmiși în lipsa resincronizării de către un transmițător ideal. Durata nominală a unui bit = $1 / \text{rata nominală de transfer}$.



Durata nominală a unui bit este specificată prin 4 segmente disjuncte, fiecare fiind format dintr-un număr întreg de cuante de timp (tq). O cuantă de timp este perioada semnalului de tact folosit de controlerul CAN și provine de regulă prin divizare din semnalul de tact al nodului.

Durata unui bit este programabilă și poate măsura între 8 și 25 tq. Ea este determinată de durata tq și de numărul de cuante din fiecare din cele 4 segmente.



SYNC_SEG – utilizat pentru a sincroniza nodurile conectate la magistrală.

PROP_SEG – utilizat pentru a compensa întârzierile de propagare a semnalului prin circuite și pe magistrală.

PHASE_SEG1 – este programabil între 1 și 8 t_q și se utilizează pentru a compensa defazajele apărute între fronturi și poate fi alungit prin resincronizare.

PHASE_SEG2 este maximul dintre PHASE_SEG1 și IPT – Information Processing Time (max. $2t_q$). Poate fi folosit, de asemenea, pentru a compensa defazajele și poate fi micșorat în timpul resincronizării.

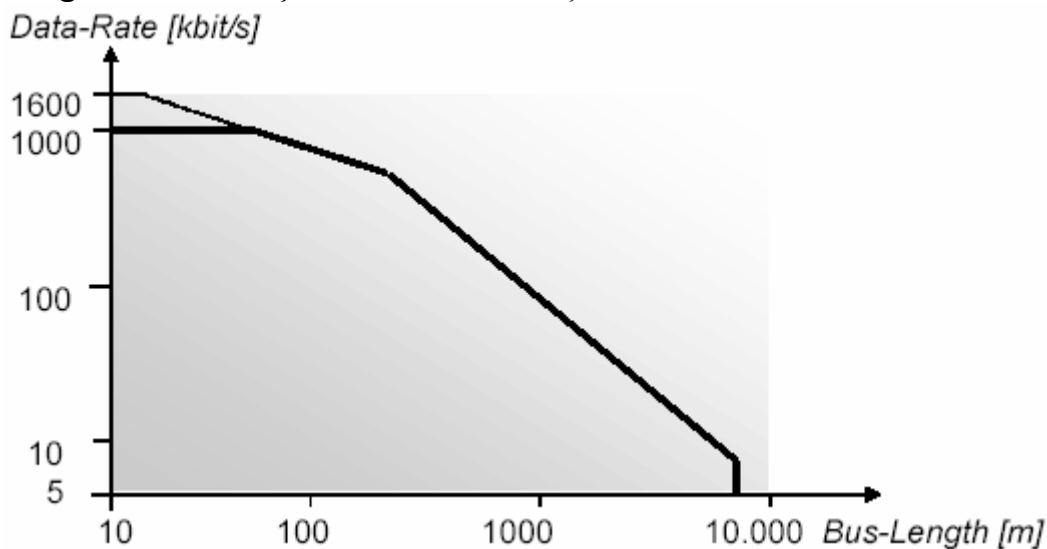
IPT începe în punctual de eșantionare și reprezintă timpul necesar pentru determinarea nivelului pentru bitul următor.

Punctul de eșantionare precizează momentul de timp în cadrul bitului în care este citit nivelul pe magistrală și interpretat conform valorii sale. Facilitatea de programare a poziției punctului de eșantionare în cadrul biților permite optimizarea funcționării protocolului CAN: o eșantionare către sfârșitul bitului este recomandată pentru dimensiunea maximă a liniei, în timp ce o eșantionare timpurie permite utilizarea unor fronturi mai puțin abrupte.

Lungimea segmentului PROP_SEG este dată de dublul sumei duratelor de propagare a semnalului prin:

- controlerul CAN (cca. 50ns),
- optocuplor ($40\div140$ ns),
- transceiver ($120\div150$ ns)
- cablu (cca. 5 ns/m).

În figura care urmează se prezintă pe o scară dublu logaritmică dependența dintre lungimea magistralei CAN și rata de comunicație.



În tabelul de mai jos se dau valori practice pentru ratele de comunicație și lungimea maximă a magistralei CAN la care acestea pot fi folosite, dacă se utilizează transceivere standard ISO 11898 și cabluri standard, în absența optocuploarelor.

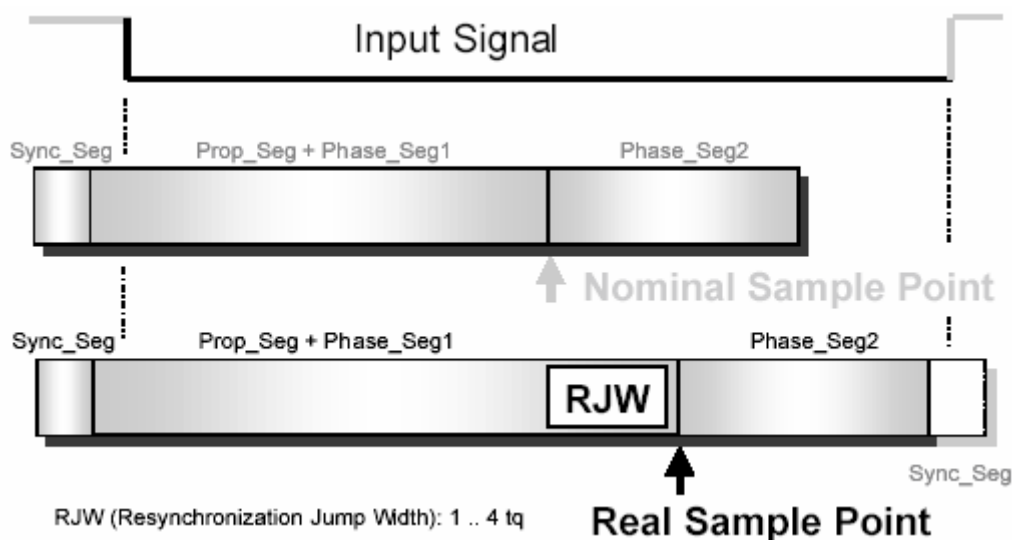
Bit Rate	Bus Length	Nominal Bit-Time
1 Mbit/s	30 m	1 μ s
800 kbit/s	50 m	1,25 μ s
500 kbit/s	100 m	2 μ s
250 kbit/s	250 m	4 μ s
125 kbit/s	500 m	8 μ s
62,5 kbit/s	1000 m	20 μ s
20 kbit/s	2500 m	50 μ s
10 kbit/s	5000 m	100 μ s

Lungimea maximă a magistralei este determinată în principal de următoarele fenomene:

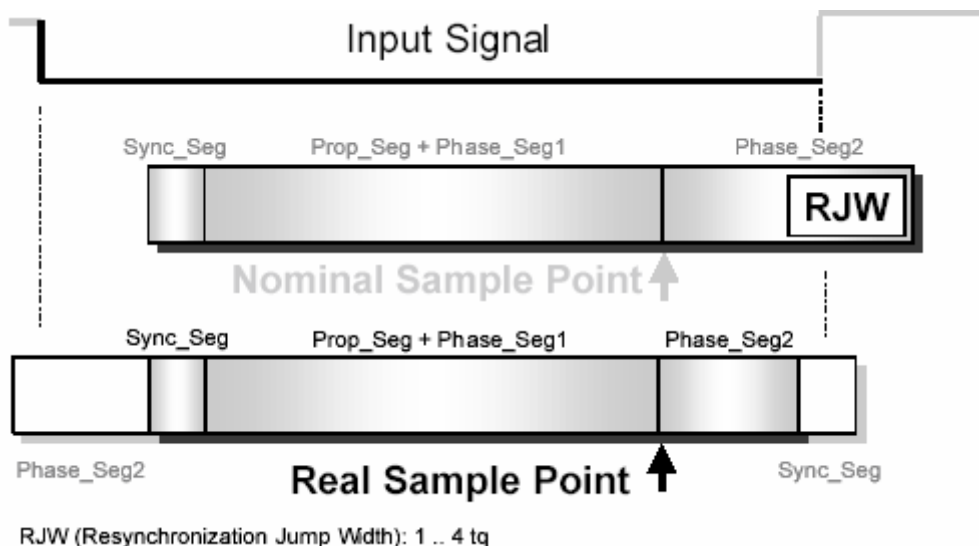
- întârzierile de propagare a semnalului prin noduri și pe magistrală;
- diferențele dintre lungimile t_q cauzate de diferențele dintre frecvențele de tact ale oscilatoarelor din care provin prin divizare;
- căderile de tensiune cauzate de rezistența serie a liniei și de impedanțele de intrare ale nodurilor.

7.2.1. Mecanisme de sincronizare la nivel de bit

O **sincronizare hardware** are loc pe frontul recesiv-dominant care apare în starea BUS IDLE. În acel moment, receptoarele forțează începerea unui bit cu segmentul SYNC_SEG. În acest fel, frontul care determină sincronizarea hardware apare pe durata segmentului SYNC_SEG a bitului restartat.

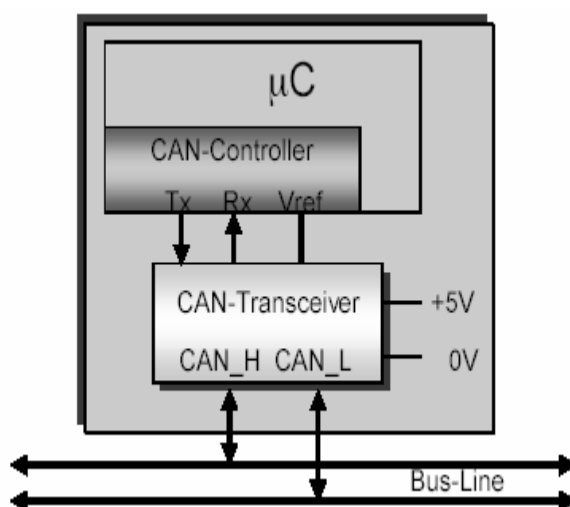
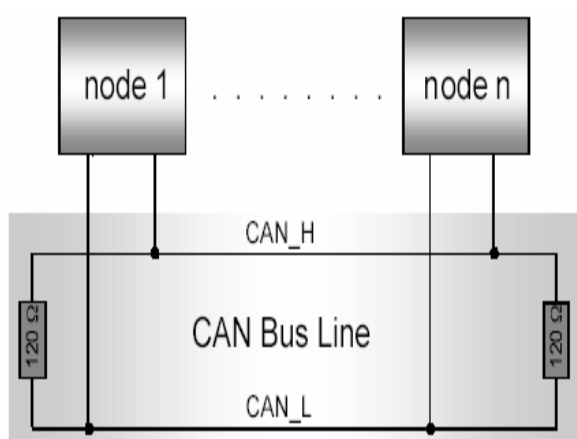


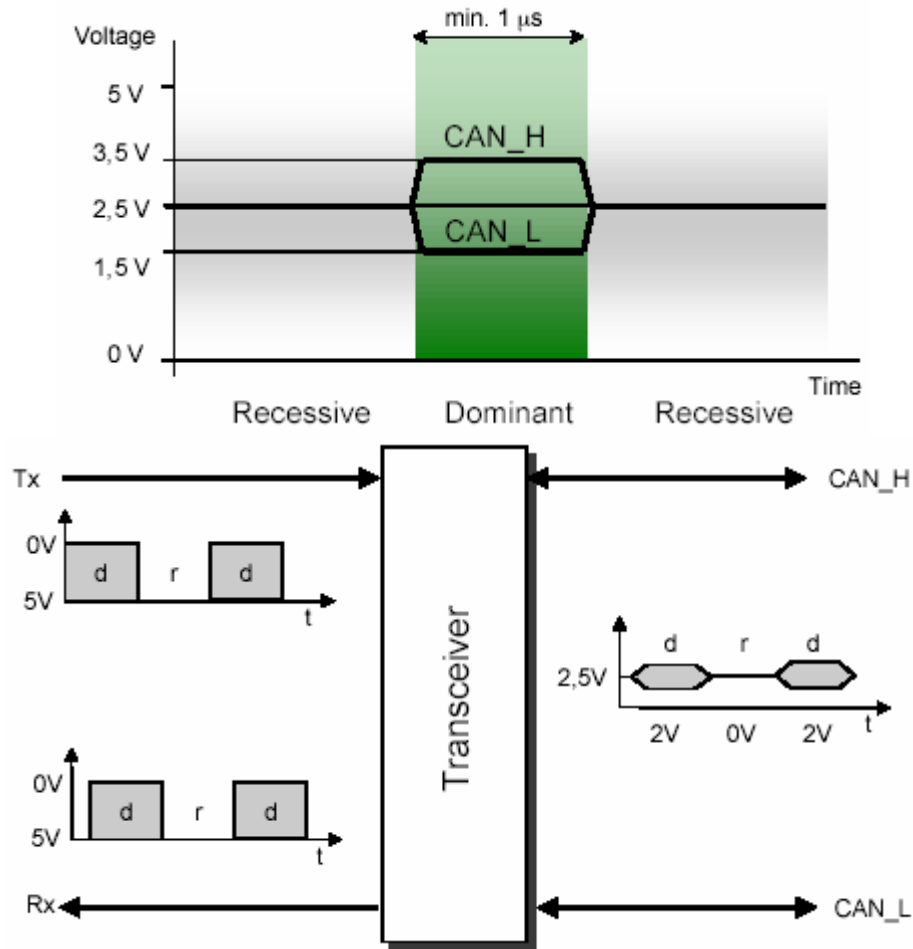
Pe durata SYNC_SEG a unui bit intern se așteaptă de regulă ca să apară un front. Dacă transmițătorul este mai lent, un nou front apare mai târziu, pe durata segmentului PROP_SEG. În acest caz, receptorul prelungește PHASE_SEG1 (resincronizare) cu un număr de cuante t_q cel mult egal cu valoarea programată pentru Resynchronization Jump Width (RJW = $1 \div 4 t_q$).



Dacă transmițătorul este mai rapid, atunci receptorul ar putea detecta frontul pe durata PHASE_SEG2, ceea ce îl va determina să scurteze următorul segment PHASE_SEG2 (resincronizare) cu un număr de cuante cel mult egal cu RJW.

Există mai multe standarde industriale de interconectare a nodurilor printr-o rețea CAN. Cel mai răspândit pentru aplicații generale este standardul CAN High Speed (ISO 11898-2).



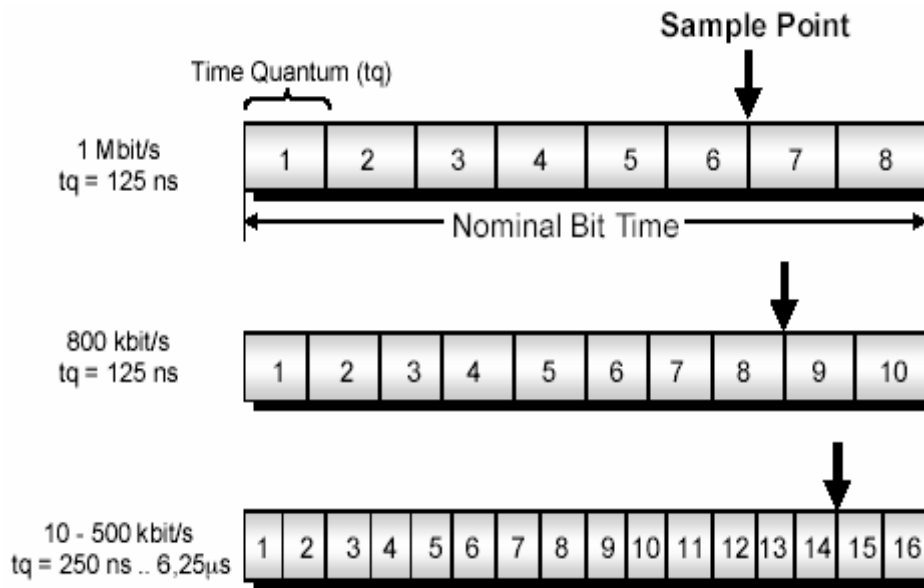


Transceivere CAN (ISO 11898-2):

- CF150B – Bosch
- 82C250, 82C251 – Philips Semiconductors
- L9615 – SGS Thomson

Rate, dimensiuni maxime ale magistralei și structuri de bit recomandate de CiA (CAN in Automation Society):

Bit rate Bus length ⁽¹⁾	Nominal bit time t_b	Number of time quanta per bit	Length of time quantum t_q	Location of sample point
1 Mbit/s 25 m	1 μ s	8	125 ns	6 t_q (750 ns)
800 kbit/s 50 m	1.25 μ s	10	125 ns	8 t_q (1 μ s)
500 kbit/s 100 m	2 μ s	16	125 ns	14 t_q (1.75 μ s)
250 kbit/s 250 m ⁽²⁾	4 μ s	16	250 ns	14 t_q (3.5 μ s)
125 kbit/s 500 m ⁽²⁾	8 μ s	16	500 ns	14 t_q (7 μ s)
50 kbit/s 1000 m ⁽³⁾	20 μ s	16	1.25 μ s	14 t_q (17.5 μ s)
20 kbit/s 2500 m ⁽³⁾	50 μ s	16	3.125 μ s	14 t_q (43.75 μ s)
10 kbit/s 5000 m ⁽³⁾	100 μ s	16	6.25 μ s	14 t_q (87.5 μ s)



7.2.2. Implementarea protocolului CAN în hardware

Modul de realizare al comunicației este identic pentru toate implementările protocolului CAN. Diferențele care există se referă la gradul de integrare a funcțiilor de comunicație pe care controlerul CAN le preia de la microcontroler.

Controlerul CAN cu buffer intermediar (denumite și cipuri BasicCAN) au implementată în hardware logica necesară pentru crearea și verificarea șirului de biți conform protocolului. Funcțiile de filtrare a mesajelor pentru testul de acceptare cad în sarcina microcontrolerului. Aceste cipuri necesită un spațiu mic și de aceea pot fi realizate cu costuri scăzute. În principiu, ele pot accepta toate obiectele dintr-o rețea CAN.

Obiectele sunt formate în principal din 3 componente:

- Identificatorul
- DLC
- datele propriu-zise.

Controlerul CAN care stochează mesajele sub formă de obiecte (denumite și cipuri FullCAN) funcționează asemănător cu controlerul cu buffere intermediare, dar pot să execute și câteva funcții de gestiune a protocolului. Spre exemplu, atunci când sunt mai multe cereri de transmisie pot să determine care obiect trebuie transmis primul. De asemenea, ele pot să se ocupe de testul de acceptare prin filtrarea obiectelor recepționate.

Aceste controlere CAN degreveză microcontrolerul de o parte cât mai mare din activitățile aferente protocolului de comunicație. Ele necesită un spațiu de implementare mai mare și de aceea sunt și mai scumpe. În plus, ele pot să administreze numai un număr limitat de obiecte.

În prezent sunt disponibile cipuri care combină ambele tipuri de implementare. Ele au capacitate de stocare a obiectelor, dintre care unul poate fi definit ca buffer intermediar. Nu mai există un punct de diferențiere de genul BasicCAN/FullCAN.

7.3. MicroPac C515C – controlerul CAN

Magistrala CAN și protocolul asociat permit comunicația eficientă între un număr de noduri interconectate:

- Rate de transfer – până la 1 Mbit/sec.
- Garantează integritatea datelor
- Degreveză procesorul de multe din activitățile legate de comunicație
- Utilizează un mecanism puternic și flexibil de comunicație prin mesaje.

Interfața CAN integrată în C515C este complet compatibilă cu modulul CAN de la microcontrolerele pe 16 biți C167CR, care a fost adaptat pentru a corespunde cerințelor de interfațare cu arhitectura microcontrolerelor de 8 biți din familia C500 (compatibilă MCS 51).

Este compusă din două blocuri principale:

- controlerul CAN
- interfața cu magistrala internă.

Controlerul CAN furnizează toate resursele necesare pentru execuția protocolului CAN standard (identificatori pe 11 biți) și extins (identificatori pe 29 de biți). El furnizează un strat la nivelul obiectelor CAN pentru a elibera UCP de cât mai multe sarcini legate de gestionarea a 15 obiecte de tip mesaj. Acestea sunt: arbitrarea accesului la magistrală, retransmisia mesajelor eronate, detecția erorilor, generarea întreruperilor etc. Pentru implementarea nivelului fizic trebuie utilizate componente hardware externe.

Interfața cu magistrala internă conectează controlerul CAN la magistrala internă a microcontrolerului. Regiștrii și locațiile de date ale interfeței CAN sunt mapate într-un spațiu de 256 de octeți în memoria externă de date (F700÷F7FFh) și pot fi accesate cu instrucțiuni MOVX.

7.3.1. Funcțiile de bază ale controlerului CAN

Controlerul de comunicații CAN integrat în structura microcontrolerului Infineon C515C este format din câteva blocuri funcționale care lucrează în paralel (vezi fig.1). Controlerul CAN permite stocarea a până la 15 obiecte de tip mesaj având o lungime de 8 octeți.

Fiecare dintre aceste obiecte are un identificator unic, precum și propriul set de biți de control și de stare. Fiecare obiect poate fi configurat ca fiind de intrare sau de ieșire, cu excepția ultimului dintre cele 15 care poate fi doar un buffer de intrare cu un registru special de mascare.

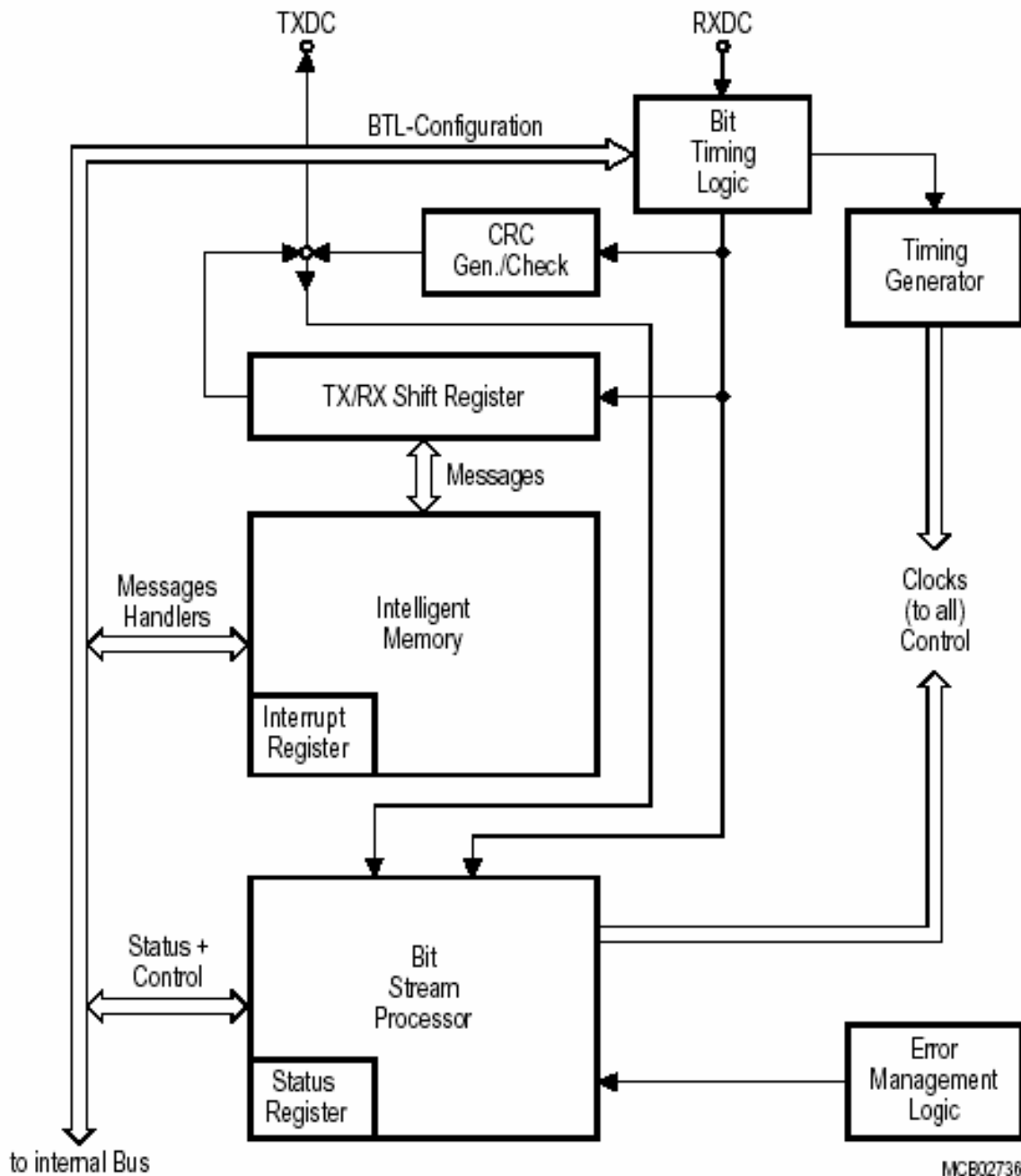
Un obiect de ieșire poate fi configurat pentru a fi transmis în mod automat ori de câte ori este recepționat un cadru de interogare (remote frame) cu un identificator corespunzător (luând în considerare registrul de mascare global).

Prin cererea transmiterii unui mesaj configurat ca intrare, se poate transmite un cadru prin care se cere ca acel obiect să fie transmis de un alt nod din rețea.

Fiecare obiect are biți de stare și de întrerupere separați pentru transmisie și recepție, ceea ce permite o mare flexibilitate microcontrolerului în detectarea momentului în care un cadru (remote/data) a fost transmis sau recepționat.

Există două măști care pot fi programate pentru filtrarea acceptării mesajelor, una pentru identificatori pe 11 biți, alta pentru identificatori pe 29 de biți. Microcontrolerul trebuie să configureze prin bitul XTD (Normal or Extended Frame Identifier) în registrul de configurare a mesajelor pentru a se determina ce fel de mesaje vor fi acceptate (format normal sau extins).

Ultimul obiect are propria sa mască ca filtru de acceptare, permitând ca un mare număr de mesaje mai puțin frecvente să poată fi gestionate de către sistem.



Structura internă a obiectelor a fost gândită astfel ca utilizarea obiectelor să fie cât mai simplă cu putință. Bufferul mesajului este implementat într-o zonă de memorie cu dublu acces care poate fi adresată atât de controlerul CAN, cât și de microcontroler. Conținutul diferiților biți din cadrul obiectului este folosit pentru a realiza funcțiile de filtrare a mesajelor acceptate, de generare a întreruperilor, de transmisie și de

determinare a terminării operațiilor de transfer. În acest buffer încap 15 mesaje a câte 8 octeți fiecare.

Controlerul CAN oferă o informație de stare extinsă față de versiunile anterioare, permițând o diagnoză mai ușoară a stării rețelei.

Registrul cu deplasare Tx/Rx

Conține șirul de biți fără biții de control de pe magistrală pentru a permite accesul paralel la date sau cadrul de cerere pentru realizarea testului de acceptare și pentru transferul cadrului spre și dinspre zona de memorie (Intelligent Memory)

Procesorul pentru șiruri de biți este un secvențiator care controlează transmisia șirului de biți între registrul cu deplasare Tx/Rx, registrul CRC și magistrală. De asemenea, BSP controlează logica de gestionare a erorilor EML și transmisia paralelă a datelor între registrul de deplasare Tx/Rx și memorie, astfel încât operațiile de recepție, de arbitrare, de transmisie și de semnalizare a erorilor sunt realizate conform protocolului CAN. Retransmisia automată a mesajelor eronate este gestionată de BSP.

Registrul CRC

Generează codul de redundanță ciclică ce se transmite după biții de date și verifică codul CRC al mesajelor care sosesc pe linie, prin împărțirea șirului de biți la polinomul generator de cod.

Logica de gestionare a erorilor

Răspunde de tratarea erorilor la nivelul controlerului CAN. Numărătoarele acestora (de erori la transmisie și de erori la recepție) sunt incrementate și decrementate prin comenzi primite de la procesorul de biți (BSP). În funcție de valorile acestor numărătoare, controlerul CAN este trecut în starea *error active*, *error passive* și *busoff*. Controlerul trece în starea *error active* dacă ambele numărătoare se află sub limita de 128.

Dacă unul dintre numărătoare atinge 128, controlerul trece în starea *error passive*. Trece în starea **busoff** dacă numărul de erori de transmisie atinge valoarea 256. În această stare rămâne până când se efectuează o secvență specială de deblocare. Mai există un bit în registrul de stare, EWRN, care este setat atunci când unul dintre contoarele de eroare atinge limita de avertizare de 96 și va fi resetat atunci când contoarele coboară sub această limită.

Logica de sincronizare la nivel de bit (BTL)

Monitorizează linia de intrare RXDC și gestionează operațiile la nivel de bit conform protocolului CAN. BTL se sincronizează pe o tranziție recesiv-dominant la bitul Start of Frame (sincronizare hard), precum și pe orice tranziție de același tip, dacă controlerul CAN nu transmite el însuși un bit dominant (resincronizare). BTL permite de asemenea programarea unor segmente de timp pentru a compensa timpul de propagare și defazajele și pentru a defini poziția momentelor de eșantionare în interiorul

unui bit. Programarea acestor intervale depinde de viteza de comunicație și de întârzierile de propagare a semnalelor pe magistrală.

Memoria adresabilă prin conținut permite stocarea unui număr de până la 15 obiecte de tip mesaj de 8 octeți. Fiecare dintre aceste obiecte are un identificator unic și propriul set de biți de control și de stare. După configurarea inițială, aceasta memorie „inteligentă” poate gestiona recepția și transmisia datelor fără nici o intervenție din partea microcontrolerului.

7.3.2. Organizarea registrelor și a obiectelor de tip mesaj

Toate registrele și obiectele de tip mesaj ale controlerului CAN sunt amplasate într-o zonă de memorie de 256 de octeți mapată în memoria externă a microcontrolerului C515C (F700÷F7FFh).

CAN Register Address Area		General Registers	
F700 _H	General Registers	Control Register	F700 _H
F710 _H	Message Object 1	Status Register	F701 _H
F720 _H	Message Object 2	Interrupt Register	F702 _H
F730 _H	Message Object 3	Reserved	F703 _H
F740 _H	Message Object 4	Bit Timing Register Low	F704 _H
F750 _H	Message Object 5	Bit Timing Register High	F705 _H
F760 _H	Message Object 6	Global Mask Short Register Low	F706 _H
F770 _H	Message Object 7	Global Mask Short Register High	F707 _H
F780 _H	Message Object 8	Upper Global Mask Long Register Low	F708 _H
F790 _H	Message Object 9	Upper Global Mask Long Register High	F709 _H
F7A0 _H	Message Object 10	Lower Global Mask Long Register Low	F70A _H
F7B0 _H	Message Object 11	Lower Global Mask Long Register High	F70B _H
F7C0 _H	Message Object 12	Upper Mask of Last Message Register Low	F70C _H
F7D0 _H	Message Object 13	Upper Mask of Last Message Register High	F70D _H
F7E0 _H	Message Object 14	Lower Mask of Last Message Register Low	F70E _H
F7F0 _H	Message Object 15	Lower Mask of Last Message Register High	F70F _H

În continuare vor fi descrise registrele CAN prezentându-se numele acestora, adresele, valoarea după reset și semnificația fiecărui bit sau câmp de biți, împreună cu tipul accesului:

- R – citire
- W – scriere
- RW – citire și scriere

După reset, unele registre CAN se încarcă cu valori prestabilite, altele își păstrează valorile anterioare (UUh), iar altele sunt nedefinite (XXh). Locațiile care rămân neschimbate după reset (UUh) sunt nedefinite (XXh) după o resetare la conectarea alimentării.

Registrele CAN se împart în registre generale și registre asociate fiecărui obiect de tip mesaj.

În număr de 15, registrele generale sunt amplasate în spațiul F700÷F70Fh. Fiecare obiect de tip mesaj, notat cu $n=1\div 15$ (sau $1\div F$ în hexazecimal), este format dintr-un număr de 15 registre amplasate într-un spațiu de 16 locații la adrese de forma F7n0÷F7nFh.

7.3.2.1. Registrele generale

Registrul de control CR

	D7	D6	D5	D4	D3	D2	D1	D0		
CR	TEST	CCE	0	0	EIE	SIE	IE	INIT	F700h	01h
	rw	rw	r	r	rw	rw	rw	rw		

TEST	Test mode	Trebuie să fie întotdeauna 0 (“1” doar în regim de testare la fabricație)
CCE	Configuration Change Enable	Permite / inhibă accesul la regiștrii de configurare a temporizărilor la nivel de bit.
EIE	Error interrupt enable	Validează / dezactivează generarea întreruperilor la modificarea stării biților BOFF și EWRN în registrul de stare
SIE	Status change interrupt enable	Validează / dezactivează generarea întreruperilor la terminarea transferului (transmisie sau recepție) unui mesaj sau la apariția unei erori (după înregistrarea ei în registrul de stare)
IE	Interrupt enable	Validează / dezactivează transmiterea cererilor de întrerupere de la modulul CAN la controlerul de întreruperi al C515C. În plus, bitul ECAN din SFR IEN2 și bitul EAL din SFR IEN0 trebuie să fie setați pentru ca o întrerupere de la modulul CAN să fie acceptată.
INIT	Initialization	Declanșează inițializarea controlerului CAN, atunci când este setat.

Registrul de stare SR

	D7	D6	D5	D4	D3	D2	D1	D0		
SR	BOFF	EWRN	-	RxOK	TxOK	LEC			F701h	XXh
	r	r	r	rw	rw	rw				

BOFF	Busoff status	Indică faptul că modulul CAN este în starea bus off (vezi EML)
EWRN	Error warning status	Indică faptul că cel puțin unul din număratoarele din EML a atins limita de avertizare de 96.
RxOK	Received message successfully	Indică faptul că un mesaj a fost recepționat cu succes de la ultima resetare a acestui bit de către microcontroller (nu e resetat de controlerul CAN).
TxOK	Transmitted message successfully	Indică faptul că un mesaj a fost transmis cu succes (fără erori și confirmat de cel puțin un alt nod) de la ultima resetare a acestui bit de către microcontroller (nu e resetat de controlerul CAN).
LEC	Last error code	Acest câmp conține un cod care indică tipul ultimei erori apărute pe magistrala CAN. Codul 111 (7) nu este folosit și poate fi scris de către microcontroller pentru a verifica dacă acest camp a fost actualizat.
LEC0÷2	Cod eroare	Descriere
0 0 0	Nici o eroare	Șters după transmisia sau recepția fără eroare a unui mesaj.
0 0 1	Eroare de umplere	Mai mult de 5 biți consecutivi de același tip au apărut într-o porțiune a mesajului în care acest lucru nu este permis.
0 1 0	Eroare de format	O parte a unui mesaj cu format fix dintr-un cadru recepționat are un format incorect.
0 1 1	Eroare de confirmare	Un mesaj transmis de acest nod nu a fost confirmat de nici un alt nod.
1 0 0	Eroare bit 1	În timpul transmisiei unui mesaj (cu excepția câmpului de arbitrare), nodul a dorit să transmită un nivel recesiv (1), dar pe magistrală a apărut un nivel dominant (0).
1 0 1	Eroare bit 0	În timpul transmisiei unui mesaj (sau a bitului de confirmare, a flagului active error sau a flagului overload), nodul a dorit să transmită un nivel dominant (0), dar pe magistrală a apărut un nivel recesiv (1). În timpul revenirii din starea busoff, acest cod este setat de fiecare dată când s-a detectat apariția unei secvențe 11 de biți recesivi. Aceasta permite ca microcontrollerul să monitorizeze desfășurarea procedurii de revenire din starea busoff (indicând faptul că magistrala nu este blocată pe nivel dominant sau continuu perturbată).
1 1 0	Eroare CRC	Cod CRC incorrect la recepția unui mesaj.

Citirea registrului de stare atunci când o cerere de întrerupere este în curs de tratare determină resetarea acestei cereri (achitarea întreruperii).

Registrul de întrerupere IR

	D7	D6	D5	D4	D3	D2	D1	D0		
IR	INTID								F703h	XXh
	r									

INTID	Interrupt identifier	Precizează cauza care a produs cererea de întrerupere. Când nu este nici o întrerupere în curs de tratare, valoarea va fi 00h.
--------------	-------------------------	--

Regiștrii de temporizare la nivel de bit BTR0 (low) și BTR1 (high)

	D7	D6	D5	D4	D3	D2	D1	D0		
BTR0	SJW		BRP						F704h	UUh
	rw		rw							

	D7	D6	D5	D4	D3	D2	D1	D0		
BTR1	0	TSEG2			TSEG1				F705h	0U...Ub
	r	rw			rw					

SJW	(Re)Synchronization jump width	Ajustează durata unui bit cu SJW+1 cuante de timp pentru resincronizare
BRP	Baud rate prescaler	Pentru generarea cuantei de timp, semnalul de la oscillator e divizat cu BRP+1
TSEG2	Time segment after sample point	$TSEG2=1\div 7$; Există TSEG2+1 cuante de timp după eșantionare
TSEG1	Time segment before sample point	$TSEG2=2\div 15$; Există TSEG1+1 cuante de timp înainte de eșantionare

BTR0 și BTR1 pot fi înscrise numai dacă bitul CCE din CR este setat.

Regiștrii de mascare

Mesajele pot folosi identificatori standard sau extinși. Cadrele care sosec pe linie sunt mascate cu ajutorul unor măști globale corespunzătoare. Bitul IDE al mesajului sosit determină dacă este utilizată o mască standard, scurtă (pe 11 biți) sau dacă se va utiliza o mască extinsă, lungă (de 29 de biți). Biții de '0' din măști înseamnă că bitul din respectiva poziție din identificatorul mesajului nu contează; orice valoare ar lua, mesajul va fi acceptat.

Ultimul obiect de tip mesaj (15) are o mască suplimentară de acceptare, programabilă individual și denumită masca ultimului mesaj. Aceasta permite ca anumite clase de mesaje să fie recepționate în acest obiect prin mascarea anumitor biți ai identificatorului. Masca ultimului mesaj este înmulțită logic (AND), bit cu bit, cu masca globală corespunzătoare mesajului recepționat.

Regiștrii măștii globale scurte GMS0 (low) și GMS1 (high)

	D7	D6	D5	D4	D3	D2	D1	D0		
GMS0	ID28÷21							F706h	UUh	
	rw									

	D7	D6	D5	D4	D3	D2	D1	D0		
GMS1	ID20÷18		1	1	1	1	1	F707h	UUU1...1b	
	rw		r	r	r	r	r			

ID28÷18	Identificier (11 bit)	Mască pentru filtrarea mesajelor recepționate cu identificator standard (11 biți)
----------------	-----------------------	---

Regiștrii măștii globale lungi:

UGML0 (upper low), UGML1 (upper high), LGML0 (lower low), LGML1 (lower high)

	D7	D6	D5	D4	D3	D2	D1	D0		
UGML0	ID28÷21							F708h	UUh	
	rw									

	D7	D6	D5	D4	D3	D2	D1	D0		
UGML1	ID20÷13							F709h	UUh	
	rw									

	D7	D6	D5	D4	D3	D2	D1	D0		
LGML0	ID12÷5							F70Ah	UUh	
	rw									

	D7	D6	D5	D4	D3	D2	D1	D0		
LGML1	ID4÷0				0	0	0	F70Bh	U...U000b	
	rw				r	r	r			

ID28÷0	Identificier (29 bit)	Mască pentru filtrarea mesajelor recepționate cu identificator extins (29 biți)
---------------	-----------------------	---

Registrii măștii ultimului mesaj:

UMLM0 (upper low), UMLM1(upper high), LMLM0(lower low), LMLM1(lower high)

	D7	D6	D5	D4	D3	D2	D1	D0		
UMLM0	ID28÷21								F70Ch	UUh
	rw									

	D7	D6	D5	D4	D3	D2	D1	D0		
UMLM1	ID20÷18				ID17÷13				F70Dh	UUh
	rw				rw					

	D7	D6	D5	D4	D3	D2	D1	D0		
LMLM0	ID12÷5								F70Eh	UUh
	rw									

	D7	D6	D5	D4	D3	D2	D1	D0		
LMLM1	ID4÷0				0	0	0	F70Fh	U...U000b	
	rw				r	r	r			

ID28÷0	Identifier (29 bit)	Mască pentru filtrarea mesajului 15 (ultimul obiect de tip mesaj) recepționat cu identificator standard (11 biți) sau extins (29 biți)
---------------	---------------------	--

7.3.2.2. Registrii obiectelor de tip mesaj

Obiectul de tip mesaj este mijlocul de bază de comunicație dintre microcontroller și controlerul CAN. Fiecare din cele 15 obiecte folosesc 15 octeți consecutivi care încep la o adresă multiplu de 16. Toate obiectele de tip mesaj trebuie să fie inițializate de către C515C, chiar și cele care nu vor fi utilizate, înainte de ștergerea bitului INIT din CR.

Fiecare element din registrele de control ale mesajelor este format din doi biți complementari. Acest mecanism special permite setarea sau resetarea selectivă a unor anumite elemente, fără a le modifica pe celelalte și fără a fi necesare cicluri citire-modificare-înscrisere. Nici unul dintre aceste elemente nu va fi afectat de reset.

Valoare câmp	Funcția la scriere	Semnificația la citire
0 0	Rezervat	Rezervat
0 1	Resetare element	Elementul este resetat
1 0	Setare element	Elementul este setat
1 1	Lasă elementul nemodificat	Rezervat

Offset

Message Control Reg. Low	+0
Message Control Reg. High	+1
Upper Arbitration Reg. Low	+2
Upper Arbitration Reg. High	+3
Lower Arbitration Reg. Low	+4
Lower Arbitration Reg. High	+5
Message Configuration Reg.	+6
Data Byte 0	+7
Data Byte 1	+8
Data Byte 2	+9
Data Byte 3	+10
Data Byte 4	+11
Data Byte 5	+12
Data Byte 6	+13
Data Byte 7	+14
Reserved	+15

Registrii de control ai obiectelor de tip mesaj MCR0 (low) și MCR1 (high)

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR0	MSGVAL		TXIE		RXIE		INTPND		F7n0h	UUh
	rw		rw		rw		rw			

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR1	RMTPND		TXRQ		MSGLST CPUUPD		NEWDAT		F7n1h	UUh
	rw		rw		rw		rw			

MSGVAL	Message valid	Indică dacă obiectul mesaj este valid sau nu. Controlerul CAN lucrează doar cu obiectele de tip mesaj care sunt valide la un moment dat.
TXIE	Transmit interrupt enable	Precizează dacă bitul INTPND va fi setat după transmisia cu success a unui cadru.
RXIE	Receive interrupt enable	Precizează dacă bitul INTPND va fi setat după recepția cu success a unui cadru (obiectul 15 are acest bit forțat în 0 pentru a împiedica transmisia).
INTPND	Interrupt pending	Indică dacă acest mesaj a generat o cerere întrerupere de la ultima resetare a lui de către microcontroller (dacă sunt validate întreruperile la transmisie sau recepție).
RMTPND	Remote pending (transmit-objects)	Indica faptul că transmisia acestui obiect de tip mesaj a fost cerută de către un nod, dar datele nu au fost încă transmise. Atunci când RMTPND este setat, controlerul setează și TXRQ. Ambele elemente sunt resetate atunci când datele obiectului au fost transmise cu success.
TXRQ	Transmit request	Indică faptul că transmisia acestui obiect este cerută de către UCP sau printr-un cadru de tip cerere și nu s-a terminat încă. Atunci când UCP dorește transmisia unui obiect de intrare, se transmite un cadru de tip cerere în locul unui de tip data. Acest bit va fi resetat de către controlerul CAN împreună cu RMTPND la transmisia cu success a cadrului cerere, dacă bitul NEWDAT nu a fost încă setat. Dacă sunt mai multe obiecte valide cu cereri de transmisie în curs de rezolvare, se va transmite mai întâi mesajul corespunzător obiectului având numărul de ordine ($n=1\div 15$) cel mai mic. TXRQ poate fi dezactivat de către CPUUPD. Pentru mesajul 15, acest element este forțat în 0.
MSGLST CPUUPD	Message lost (receive-objects)	Indică faptul că controlerul CAN a memorat un nou mesaj în acest obiect, în timp ce NEWDAT era deja setat, adică mesajul anterior s-a pierdut.
	CPU update (transmit-objects)	Indică faptul că mesajul nu poate fi transmis acum. Microcontrolerul setează acest bit pentru a inhiba transmisia unui mesaj care este în curs de actualizare, sau pentru a controla răspunsul automat la cererile de transmisie primite.

NEWDAT	New data	<p>Indică faptul că au fost înscrise date noi în zona de date a mesajului de către microcontroller (obiectele de ieșire) sau de către controlerul CAN (obiectele de intrare) de la ultima resetare a acestui bit.</p> <p>Controlerul CAN poate să modifice zona de date a unui obiect de intrare și abia apoi activează acest bit. Microcontrollerul trebuie să reseteze acest bit înainte de a începe să preia datele mesajului și să verifice dacă acesta a rămas resetat la terminarea lucrului, pentru a evita să lucreze cu o parte din datele unui mesaj vechi și o altă parte dintr-un mesaj nou.</p> <p>La transmisie, microcontrollerul va seta acest bit odată cu resetarea bitului CPUUPD. Dacă mesajul current este în curs de transmisie la momentul actualizării de către microcontroller, atunci controlerul CAN nu va reseta bitul TXRQ. În acest fel, bitul TXRQ va fi resetat abia în momentul în care datele actualizate sunt transmise cu succes.</p>
---------------	----------	---

Registrii de arbitrare

UAR0 (upper low), UAR1 (upper high), LAR0 (lower low), LAR1 (lower high)

	D7	D6	D5	D4	D3	D2	D1	D0		
UAR0	ID28÷21								F7n2h	UUh
	rw									

	D7	D6	D5	D4	D3	D2	D1	D0		
UAR1	ID20÷18				ID17÷13				F7n3h	UUh
	rw				rw					

	D7	D6	D5	D4	D3	D2	D1	D0		
LAR0	ID12÷5								F7n4h	UUh
	rw									

	D7	D6	D5	D4	D3	D2	D1	D0		
LAR1	ID4÷0				0	0	0	F7n5h	U...U000b	
	rw				r	r	r			

ID28÷0	Identificier (29 bit)	Mască pentru filtrarea mesajului n recepționat cu identificator standard (11 biți – ID17÷0 sunt indiferenți) sau extins (29 biți).
---------------	-----------------------	--

Acești regiștri sunt utilizați pentru filtrarea mesajelor recepționate și pentru definirea identificatorului mesajelor transmise. Un mesaj primit este stocat într-un

obiect valid având un identificator potrivit și DIR=0 (data frame) sau DIR=1 (remote frame). Cadrele în format extins pot fi stocate numai în obiecte cu XTD=1, iar cadrele standard numai în obiecte cu XTD=0. Pentru compararea identificatorilor, este luată în considerare și masca globală (iar pentru obiectul 15 și masca ultimului mesaj). Dacă un mesaj recepționat (data frame sau remote frame) corespunde la mai mult de un obiect valid, este stocat în cel care are numărul de ordine mai mic ($n=1\div 15$).

Atunci când controlerul CAN stochează un cadru sunt reținute în cadrul obiectului nu numai octeții de date, ci și întregul identificator, precum și lungimea mesajului (mesajele cu identificatori standard completează cu 0 biții ID17÷ID0. Aceasta este implementată pentru a păstra împreună octeții de date și identificatorul, chiar dacă se utilizează regiștrii de mascare. Atunci când controlerul CAN stochează un cadru de tip cerere, numai lungimea mesajului este reținută în obiectul corespunzător. Identificatorul și octeții de date rămân nemodificați.

Nu trebuie să existe mai mult de un obiect valid asociat la un anumit identificator, în orice moment. Dacă anumiți biți sunt mascați de masca globală, atunci identificatorii obiectelor valide trebuie să difere în biții rămași și care sunt utilizați pentru filtrare. Dacă un cadru recepționat este stocat într-un obiect, identificatorul acelui obiect este actualizat. Dacă unii dintre biții identificatorului sunt mascați cu ajutorul măștii globale (și a celei a ultimului mesaj), acești biți pot fi modificați în obiectul de intrare. Dacă este recepționat un cadru de tip cerere, identificatorul obiectului de ieșire rămâne nemodificat, cu excepția ultimului obiect, care nu poate fi transmis. În acest caz, biții identificatorului corespunzători biților indiferenți din masca ultimului obiect pot fi modificați de mesajul recepționat.

Registrul de configurare a mesajului MCFG

	D7	D6	D5	D4	D3	D2	D1	D0		
MCFG	DLC				DIR	XTD	0	0	F7n6h	U...U00b
	rw				rw	rw	r	r		

DLC	Data length code	Valori permise: 0÷8
DIR	Message direction	DIR=1: obiect de ieșire. La TXRQ, mesajul asociat este transmis. La receptia unui cadru de cerere cu identificator potrivit, biții TXRQ și RMTPND ai mesajului sunt setați. DIR=0: obiect de intrarea. La TXRQ, se transmite un cadru de cerere având identificatorul acestui mesaj. La recepția unui cadru de date având un identificator corespunzător, cadrul este stocat în acest obiect.
XTD	Extended identifier	Precizează că acest obiect va folosi un identificator extins (29 biți) sau standard (11 biți).

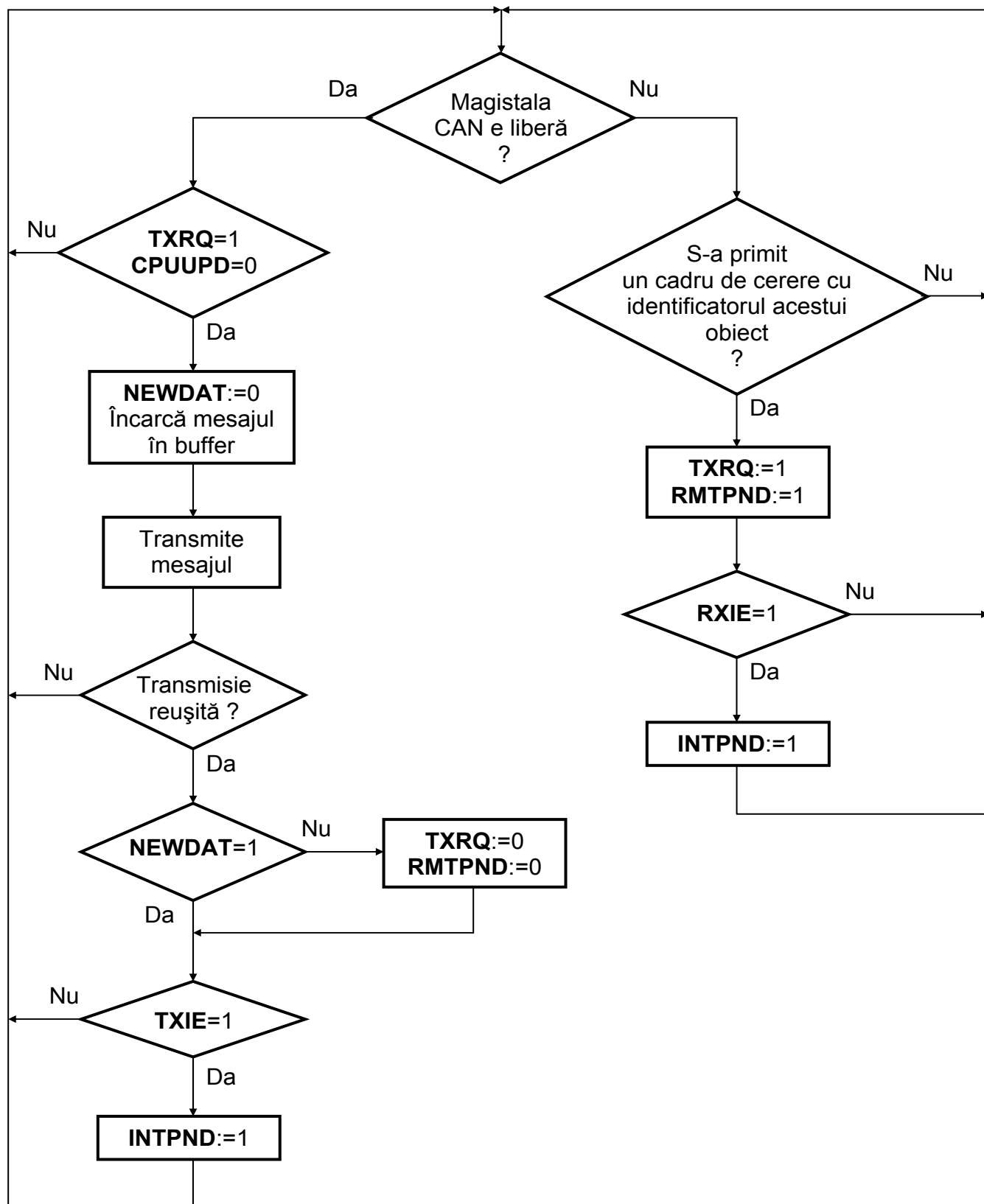
Registrii de date DB0÷DB7

	D7	D6	D5	D4	D3	D2	D1	D0		
DB0÷DB7	.7	.6	.5	.4	.3	.2	.1	.0	F7n7÷F7nEh	XXh
	rw									

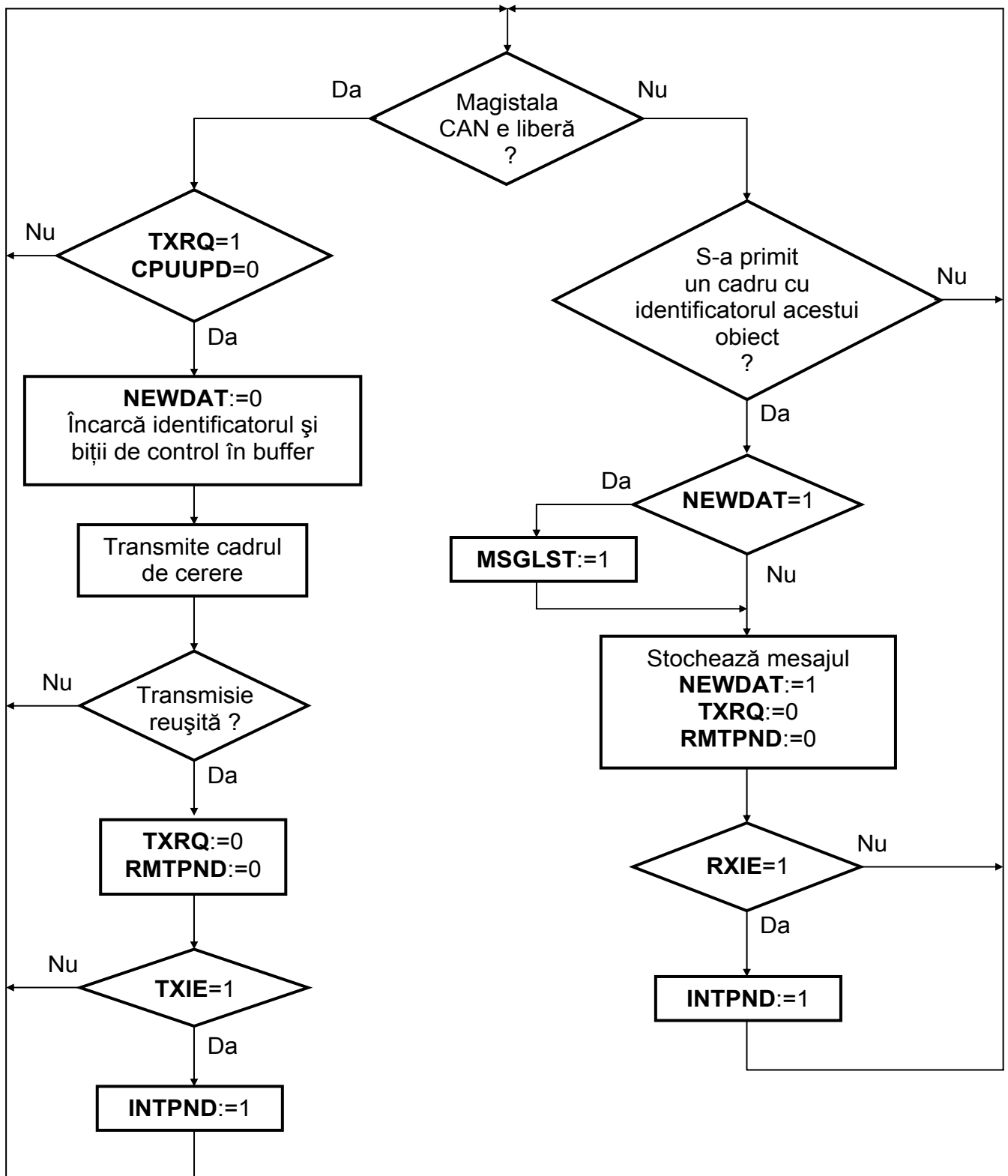
Datele pentru ultimul mesaj (15) vor fi înscrise într-un buffer cu două niveluri pentru a preveni pierderea unui mesaj atunci când un al doilea mesaj a fost recepționat înainte ca microcontrolerul să-l citească pe primul.

7.3.3. Gestionarea mesajelor

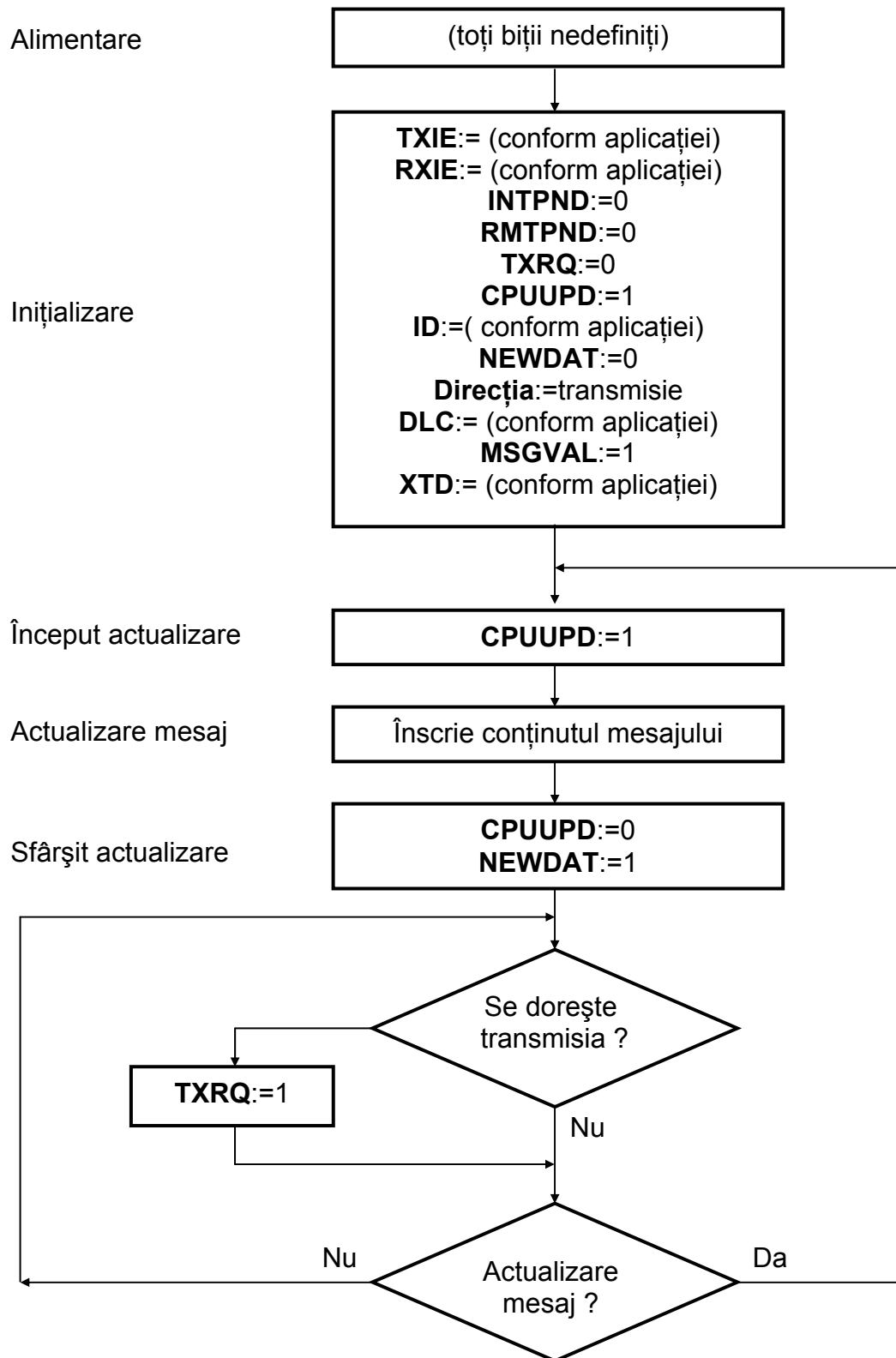
Următoarele diagrame ilustrează acțiunile care trebuie efectuate, atât de către controlerul CAN cât și de microcontroler, pentru a se putea transmite și recepționa mesaje în rețeaua CAN.



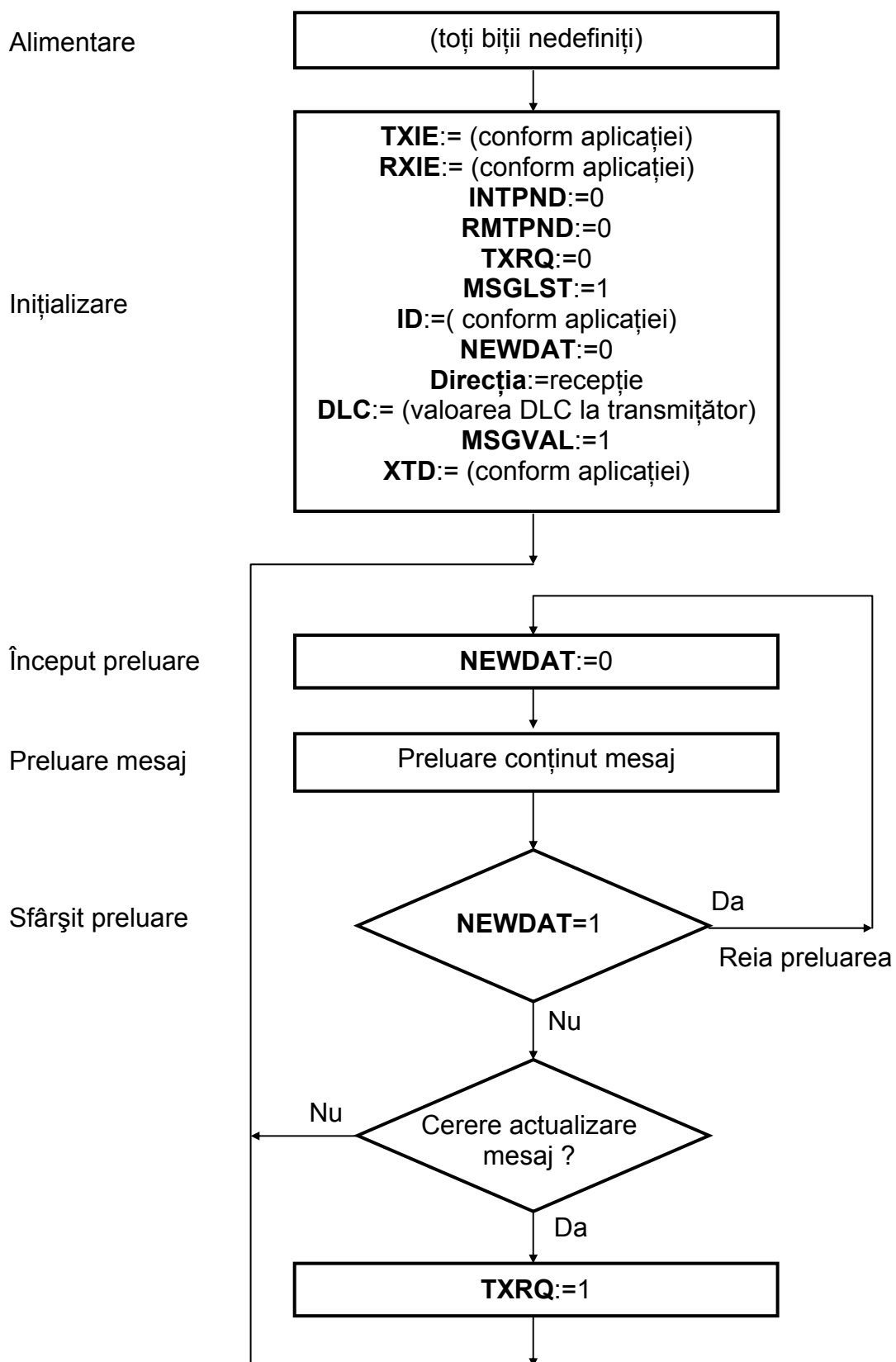
Gestionarea de către controlerul CAN a obiectelor de ieșire



Gestionarea de către controlerul CAN a obiectelor de intrare



Gestionarea de către microcontroler a obiectelor de ieșire



Gestionarea de către microcontroler a obiectelor de intrare

7.3.4. Inițializarea și resetarea

Controlerul CAN este resetat printr-un reset hardware sau printr-un reset comandat de ceasul de gardă. La resetare, controlerul CAN realizează următoarele operații:

- comandă ieșirea TXDC pe nivel 1 (recesiv)
- șterge numărătoarele de erori
- șterge starea busoff
- încarcă registrul de control (low) cu 01h
- lasă registrul de control (high) și registrul de întreruperi nedefinite
- nu modifica celelalte registre, inclusive registrele mesajelor

Primul reset hardware de după conectarea alimentării lasă registrele nemodificate într-o stare nedefinită. Valoarea 01h în registrul de control (low) pregătește inițializarea software.

Inițializarea software

Este validată prin setarea bitului INIT din registrul de control. Aceasta se poate realiza de către microcontroler prin program, sau automat, de către controlerul CAN - la resetare hardware, sau dacă EML comută în starea busoff. În timp ce INIT este setat:

- toate transferurile de mesaje sunt oprite
- ieșirea TXDC este forțată în 1 (recesiv)
- biții de control NEWDAT și RMTPND ai ultimului mesaj sunt resetați
- numărătoarele EML sunt lăsate nemodificate

În plus, setarea bitului CCE permite modificarea configurației în registrul de temporizare la nivel de bit.

Pentru inițializarea controlerului CAN trebuie ca microcontrolerul să execute următoarele operații:

- configurarea registrului de temporizare la nivel de bit
- setarea registrelor măștii globale
- inițializarea fiecărui obiect.

Dacă un obiect nu este utilizat, este suficient să se șteargă bitul de mesaj valid (MSGVAL). Altfel, întregul mesaj trebuie inițializat. După terminarea secvenței de inițializare, microcontrolerul trebuie să șteargă bitul INIT. În acest moment BSP se sincronizează cu transferul de date pe magistrala CAN așteptând apariția unei secvențe formate din doi biți recesivi consecutivi (11) care să informeze despre starea bus idle, înainte de a putea să ia parte la activități și să înceapă să transmită mesaje.

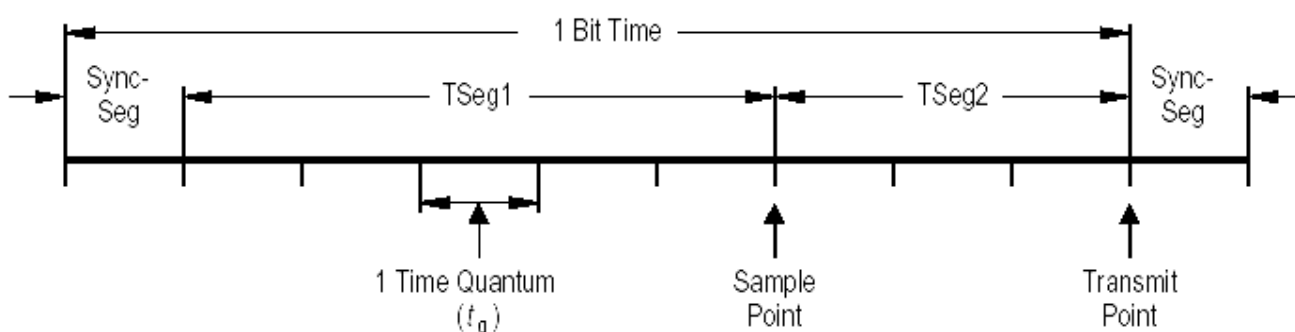
Inițializarea mesajelor este independentă de starea bitului INIT și poate fi realizată din mers. Totuși, trebuie ca toate mesajele valide să primească identificatorii înainte ca BSP să înceapă transmisia mesajelor. Pentru a modifica configurația unui mesaj în timpul funcționării normale, microcontrolerul mai întâi șterge MSGVAL, definind mesajul ca invalid. După reconfigurare, MSGVAL este setat din nou.

De notat faptul că secvența de revenire din starea busoff nu poate fi scurtată prin setarea sau resetarea bitului INIT. Dacă dispozitivul trece în starea busoff, el va seta INIT din proprie inițiativă, oprind toate activitățile de pe magistrală. Odata ce INIT a fost șters de microcontroller, dispozitivul va aștepta apariția de 129 de ori a stării bus idle înainte de a-și relua operarea normală. La finalul secvenței de revenire din busoff, număratoarele EML vor fi șterse.

În timpul așteptării de după resetarea lui INIT, de fiecare dată când este detectată o secvență 11, codul de eroare Bit 0 este înscris în registrul de control, permițând microcontrolerului să verifice dacă magistrala CAN este blocată pe nivel dominant sau este perturbată continuu, precum și pentru a monitoriza evoluția secvenței de revenire din busoff.

7.3.5. Configurarea registrului de temporizare la nivel de bit

Conform specificației CAN, durata unui bit este împărțită în 4 intervale. Fiecare interval este multiplu al cuantei de timp t_q . Intervalul de sincronizare Sync-Seg are întotdeauna lungimea unei cuante t_q . Intervalul de timp de propagare și segmentul de fază 1 sunt cumulate în Tseg1 și definesc intervalul de timp de dinaintea momentului eșantionării, în timp ce segmentul de fază 2 (Tseg2) definește intervalul de timp de după momentul eșantionării. Lungimea acestor segmente este programabilă.



Durata unui bit este determinată de către perioada de tact a microcontrolerului - CLP, divizorul pentru rata de comunicație și numărul de cuante de timp al unui bit:

$$\text{bit time} = t_{\text{Sync-Seg}} + t_{\text{TSeg1}} + t_{\text{TSeg2}}$$

$$t_{\text{Sync-Seg}} = 1 \cdot t_q$$

$$t_{\text{TSeg1}} = (\text{TSEG1} + 1) \cdot t_q \quad (= \min. 4 \cdot t_q)$$

$$t_{\text{TSeg2}} = (\text{TSEG2} + 1) \cdot t_q \quad (= \min. 3 \cdot t_q)$$

$$t_q = (\text{BRP} + 1) \cdot \text{CLP} \quad (= \min. \text{CLP})$$

TSEG1, TSEG2 și BRP sunt valorile numerice programate pentru respectivele câmpuri din registrul de temporizare la nivel de bit.

7.3.5.1. Sincronizarea și resincronizarea hardware

Pentru a compensa defazajele dintre oscilatoarele diferitelor controlere CAN, fiecare controller CAN trebuie să se sincronizeze pe orice front descrescător (recesiv către dominant) dacă frontal apare între momentul eșantionării și următorul segment de sincronizare, precum și pe orice front dacă el însuși nu transmite un nivel dominant.

Dacă sincronizarea hardware este validată (la începutul cadrului) durata bitului este reinițializată la momentul segmentului de sincronizare. Altfel, intervalul de salt pentru resincronizare (SJW) definește numărul maxim de cuante de timp cu care durata unui bit poate fi scurtată sau prelungită la resincronizare. Durata bitului current este ajustată cu

$$t_{SJW} = (SJW + 1) \cdot t_q,$$

unde SJW este valoarea numerică programată în câmpul corespunzător din registrul de temporizare la nivel de bit.

7.3.5.2. Calcularea duratei unui bit

Programarea duratei unui bit conform specificațiilor CAN depinde de rata de comunicație dorită, de perioada de tact a microcontrolerului și de întârzierile introduce de driverul de magistrală, de liniile de comunicație și de intrarea comparatorului. Aceste întârzieri sunt grupate în segmentul de propagare t_{Prop} , unde t_{Prop} este de două ori maximul sumei dintre întârzierile introduce de magistrală, de intrarea comparatorului și de driverul de ieșire rotunjită la cel mai apropiat multiplu de t_q .

Pentru a îndeplini cerințele specificațiilor CAN trebuie asigurate următoarele condiții:

$$t_{TSeg2} \geq 3 \times t_q = \text{Information Processing Time}$$

$$t_{TSeg2} \geq t_{SJW}$$

$$t_{TSeg1} \geq 4 \times t_q$$

$$t_{TSeg1} \geq t_{SJW} + t_{Prop}$$

Pentru a se obține o funcționare corectă conform protocolului CAN, durata unui bit trebuie să fie cel puțin egală cu $8t_q$, adică

$$t_{TSeg1} + t_{TSeg2} \geq 7 t_q$$

Astfel, pentru a lucra cu 1Mbit/sec, frecvența de lucru a microcontrolerului trebuie să fie de cel puțin 8MHz. Toleranța maximă pentru CLP depinde de PB1 (phase buffer segment1), PB2 (phase buffer segment2) și de saltul de resincronizare SJW:

$$df \leq \frac{\min(PB1, PB2)}{2 \times (13 \times \text{bit time} - PB2)} \quad \text{și} \quad df \leq \frac{t_{SJW}}{20 \times \text{bit time}}$$

7.3.6. Gestionarea întreruperilor generate de CAN

Controlerul CAN are o ieșire de întrerupere care este conectată la controlerul de întreruperi al microcontrolerului C515C. Această sursă de întrerupere poate fi validată/inhibată cu ajutorul bitului ECAN din SFR IEN2. În plus, trei biți din registrul de control RC din XRAM (F701h) pot fi folosiți pentru a valida/inhiba anumite surse de întrerupere interne ale CAN.

Întrucât o cerere de întrerupere de la modulul CAN poate fi generată de diferite surse interne, cauza exactă care a produs cererea de întrerupere poate fi determinată în rutina de tratare a întreruperii prin citirea registrului de stare SR. Identificatorul întreruperii se află localizat în registrul de întrerupere IR (câmpul INTID). Atunci când nu este nici o cerere de întrerupere în așteptare, acest registru are valoarea "00h". Dacă valoarea INTID este diferită de "00h", înseamnă că există o întrerupere în așteptare.

Dacă bitul IE din registrul de control este setat, atunci linia de întrerupere de la modulul CAN este activată. Această linie rămâne activă până când fie INTID este pus pe "00h" (indicând tratarea cererii), fie IE este resetat (la dezactivarea întreruperilor generate de CAN). Cererea cu codul INTID cel mai mic este cea mai prioritară. Dacă o cerere de prioritate mai mare apare înainte de tratarea întreruperii curente, atunci INTID este actualizat și noua cerere se suprapune peste ultima.

În tabelul care urmează sunt prezentate valorile valide pentru INTID și sursele interne de întrerupere corespunzătoare.

INTID	Cauza întreruperii	Descriere
00	Interrupt Idle	Nu există nici o cerere de întrerupere.
01	Status Change Interrupt	Controlerul CAN a actualizat (nu neapărat a modificat) registrul de stare. Această poate indica o modificare în registrul de stare a erorilor (EIE era setat și BOFF sau EWRN s-au modificat), sau a apărut un incident legat de transfer (SIE trebuie să fi fost setat) cum ar fi recepția sau transmiterea unui mesaj (este setat RXOK sau TXOK) sau a apărut o eroare pe magistrala CAN (LEC este actualizat). Microcontrolerul poate șterge RXOK, TXOK sau LEC, totuși, scrierea registrului de stare nu poate niciodată genera sau șterge o cerere de întrerupere. Pentru a actualiza codul INTID, trebuie ca registrul de stare SR să fie citit.
02	Message object 15 interrupt	Bitul INTPND din registrul de control al obiectului 15 a fost setat. Obiectul 15 (ultimul) are cea mai mare prioritate dintre toate obiectele.
2+N	Message N interrupt	Bitul INTPND din registrul de control al obiectului N a fost setat ($N=1\div 14$). De notat că un cod de întrerupere de la un mesaj apare în registrul de întreruperi IR numai dacă nu există o altă cerere mai prioritară.

Bitul INTPND al obiectului corespunzător trebuie să fie resetat pentru a permite obiectelor mai puțin prioritare să actualizeze sau să reseteze ulterior INTID.

7.3.7. Controlerul CAN în modurile de consum redus

Idle Mode

Controlerul CAN este complet funcțional. Atunci când apare o întrerupere de la controlerul CAN și aceasta este validată, microcontrolerul C515C repornește, revenind în modul de operare normală, după care trece la servirea întreruperii de la CAN.

Slow Down Mode

Controlerul CAN este pilotat cu un semnal de frecvență redusă: 1/32 din frecvența normală. De aceea, durata unui bit în acest mod este de 32 de ori mai mare decât în modul de operare normală. Acest mod poate fi combinat cu Idle Mode.

Power Down Mode

Dacă microcontrolerul C515C trece prin program în acest mod, semnalul de tact al sistemului este interrupt, ceea ce determină și oprirea modulului CAN. Orice transfer în curs este interrupt. Pentru a se asigura că modulul CAN nu este oprit în timp ce transmitea un nivel dominant pe magistrala CAN, microcontrolerul trebuie să seteze bitul INIT în CR înainte de a intra în acest mod. De asemenea, microcontrolerul poate verifica dacă există o transmisie în curs dacă citește biții TXRQ și NEWDAT în cadrul obiectelor, precum și TXOK în CR. După revenirea din acest mod controlerul CAN trebuie reconfigurat.

7.3.8. Exemplu de configurare a unui obiect de ieșire

Microcontrolerul dorește să configureze un obiect pentru transmisie. Se dorește transmiterea automată a mesajului drept răspuns la cadre de cerere, dar nu se dorește generarea întreruperilor pentru acest obiect.

Inițializare: Sunt stabilite identificatorul și tipul obiectului. CPUUPD este inițial setat, iar NEWDAT este resetat, întrucât octeții de date ai mesajului nu sunt încă inițializați.

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR0	1	0	0	1	0	1	0	1	F7n0h	UUh
	MSGVAL		TXIE		RXIE		INTPND			
	rw		rw		rw		rw			

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR1	0	1	0	1	1	0	0	1	F7n1h	UUh
	RMTPND		TXRQ		CPUUPD		NEWDAT			
	rw		rw		rw		rw			

După actualizarea datelor mesajului, bitul CPUUPD trebuie resetat, iar NEWDAT trebuie setat, pentru a indica existența unei date noi de la CPU. Dacă se dorește și transmisia imediată a mesajului, atunci se setează TXRQ (10), altfel, acesta este resetat (01) sau este lăsat nemodificat (11).

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR1	0	1	0	1	0	1	1	0	F7n1h	UUh
	RMTPND		TXRQ		CPUUPD		NEWDAT			
	rw		rw		rw		rw			

După primirea unui cadru de cerere, configurația devine automat:

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR1	1	0	1	0	0	1	0	1	F7n1h	UUh
	RMTPND		TXRQ		CPUUPD		NEWDAT			
	rw		rw		rw		rw			

7.3.9. Exemplu de configurare a unui obiect de intrare

Microcontrolerul dorește să configureze un obiect pentru recepție. El dorește să fie atenționat printr-o întrerupere de fiecare dată când primește date noi pentru acest obiect. Periodic, microcontrolerul transmite câte un cadru de cerere pentru a determina transmiterea datelor de către un alt nod.

Inițializare: Sunt stabilite identificatorul și tipul obiectului. CPUUPD și NEWDAT sunt reșetați.

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR0	1	0	0	1	1	0	0	1	F7n0h	UUh
	MSGVAL		TXIE		RXIE		INTPND			
	rw		rw		rw		rw			

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR1	0	1	0	1	0	1	0	1	F7n1h	UUh
	RMTPND		TXRQ		MSGLST		NEWDAT			
	rw		rw		rw		rw			

După primirea de noi date, configurația devine automat:

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR0	1	0	0	1	1	0	1	0	F7n0h	UUh
	MSGVAL		TXIE		RXIE		INTPND			
	rw		rw		rw		rw			

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR1	0	1	0	1	0	1	1	0	F7n1h	UUh
	RMTPND		TXRQ		MSGLST		NEWDAT			
	rw		rw		rw		rw			

Pentru a prelua datele recepționate de modulul CAN, microcontrolerul trebuie să reseteze INTPND și NEWDAT, să citească datele și apoi să verifice dacă NEWDAT a rămas resetat. Dacă nu, procedura se repetă.

Pentru a transmite un cadru de cerere, microcontrolerul trebuie doar să seteze bitul TXRQ.

	D7	D6	D5	D4	D3	D2	D1	D0		
MCR1	1	1	1	0	1	1	1	1	F7n1h	UUh
	RMTPND		TXRQ		MSGLST		NEWDAT			
	rw		rw		rw		rw			

Acest bit va fi șters de către controlerul CAN de îndată ce cadrul de cerere a fost transmis sau dacă datele au fost primite înainte ca modulul CAN să fi reușit transmitia cadrului de cerere.

Interfața cu aplicația

Controlerul CAN integrat în microcontrolerul C515C nu are implementat și nivelul fizic prin care să se conecteze la magistrala CAN. Acesta trebuie implementat în exterior, ținând cont de faptul că modulul CAN furnizează două linii: RXDC (P4.7) și TXDC (P4.6). Un nivel logic “0” este interpretat ca nivel dominant aplicat pe magistrala CAN, în timp ce un nivel logic “1” este asociat cu nivelul recesiv.