UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI

Marco Antonio Marques

CAN AUTOMOTIVO
SISTEMA DE MONITORAMENTO

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como requisito parcial à obtenção do título de ***Mestre em Ciências em Engenharia Elétrica, na Área de Concentração Automação e Sistemas Elétricos Industriais***

Orientadora: Profa. Dra. Lucia Regina Horta Franco

Itajubá, abril de 2004

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI

Marco Antonio Marques

CAN AUTOMOTIVO
SISTEMA DE MONITORAMENTO

Dissertação aprovada por banca examinadora em 29 de Abril de 2004, conferindo ao autor o título de ***Mestre em Ciências em Engenharia Elétrica****, na* ***Área de Concentração Automação e Sistemas Industrias.***

Banca Examinadora:

Prof. Dr. Carlos Eduardo Cugnasca (USP)

Prof. Dr. Valberto Ferreira da Silva (UNIFEI)

Profa. Dra. Lucia Regina Horta Franco (Orientadora - UNIFEI)

DEDICATÓRIA:

Dedico este trabalho a minha esposa Miriam e meus filhos Michel e Miriane que tiveram paciência e me deram bastante incentivo; minha esposa Mirian que nunca faltou com suas palavras de carinho, apoio e principalmente amor. Meus pais e meus sogros que sempre incentivaram o meu trabalho e sempre me ajudaram.

AGRADECIMENTOS:

Agradecimentos às pessoas que de alguma forma auxiliaram a realização deste trabalho de dissertação através do apoio moral, principalmente a minha esposa Miriam e até mesmo apoio técnico principalmente de Everton Lívio Pedon Ribeiro e dentre os muitos colaboradores da Universidade Federal de Itajubá, Universidade de Mogi das Cruzes e a empresa Arcicar de Mogi das Cruzes destacam-se: os colegas Gustavo Gonçalves, Thiago Almeida, Ricardo Bannwart e Arcidio da Arcicar.

Agradecimento especial ao professor Wilton Ney do Amaral Pereira que com sua palavra de apoio e com sua ajuda foi possível iniciar o curso.

E à Profª Lúcia Franco, pessoa responsável pelo sucesso do trabalho nas questões técnicas e apoio moral um agradecimento muito especial.

SUMÁRIO

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema (hardware e software em tempo real) para comunicação com a rede CAN dos sistemas automotivos. Caracterizou-se, inicialmente, pelo estudo das redes de comunicação intraveiculares (proprietárias), buscando levantar os códigos utilizados para a troca de dados entre várias partes dos sistemas veiculares com o objetivo de se chegar aos códigos suficientes e necessários para a comunicação com seus sistemas de manutenção. Para isto, procurou-se identificar, através de monitoração, análises e testes, os códigos trocados entre diferentes unidades de controle de alguns veículos de diferentes fabricantes. Estas análises geraram uma base para desenvolvimento de programas de computadores para auxiliar o profissional a analisar e dar manutenção no sistema veicular de maneira mais clara e direta, otimizando tanto o seu tempo como seus recursos físicos e financeiros.

ABSTRACT

This work presents the development of a system (hardware and software in real time) for communication of the automotive systems. It started with the study of CAN protocols (intra-vehicles and proprietaries), in order to find out the codes used for the exchange of data among several parts of the transport systems. The goal is to discover the enough and necessary codes for the communication among vehicles and their maintenance systems. For this, this work tried to identify, through monitoring, analyses and tests, the codes exchanged among different units of control of some vehicles. These analyses generated a base for development of programs to help the professional to analyze and to give maintenance in the transport system in a clearer and direct way, optimizing as much his time as his physical and financial resources.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ROTINAS

LISTA DE ABREVIATURAS

ABS – Anti Lock Brakes - Freio hidráulico
ACK – Acknowledgment
CAN - Controller Area Network – Rede digital serial multiplexada
CRC – Cyclic Redundancy Check
CS – Chip Selec
CSMA – Carrier Sense Multiple Acess
CSMA/BA -Carrier Sense Multiple Access with Bitwise Arbitration
CTS – Sensor de temperatura
DLC – Data Length Code - CAN
DLC – Data Link Conector - OBD
DPLL – Digital Phase Lock Loop
EOF – End of Frame
GM – General Motors
ID – Identifier
IDE – Identifier Extension
INT – Interrupt
ISO - International Organization for Standardization
LabField – Laboratório de fieldbus
MCP2510 – Microchip 2510
MCU – CAN – Microcontroller Unit – Controller Area Network
MDI - Medium Dependent Interface
NBC – Body Computer
NCM – Motor
NQS – Painel de instrumentos
NRZ - Non Return to Zero
OBD – On Board Diagnostics
OBDII – Up date On Board Diagnostics standard effective in US after 1-1-1996
OSI - Open Systems Interconnection
PC – Personal Computer
PLL – Phase Lock Loop
PLS - Physical Signaling
PMA - Physical Medium Attachment
Point-to-Point – ponto a ponto
PWM – Pulse Width Modulation (J1850/SAE – FORD)
REC – Receiver Error Counter
RTR – Remote Transmission Request
SAE - Society of Automotive Engineers
SI – Serial Input
SJW – Synchronization Jump Width
SO – Serial Output
SOF – Start of Frame
SPI - Serial Peripheral Interface
SRR – Substitute Remote Request

TEC – Transmit Error Counter
TQ – Time de Quanta
Ve.N.I.C.E - Vehicle Net With Integrated Control
VPW – Variable Pulse Width Modulated (J1850/SAE – GM/Chryler)

1 CAPÍTULO 1 - INTRODUÇÃO

1.1 MOTIVAÇÃO DO TEMA

A tecnologia está presente em diversos setores industriais, comerciais ou residenciais. O que se deseja atingir geralmente é a perfeição nas realizações de tarefas, a exatidão e segurança com que um processo deverá ser executado e a confiabilidade nas suas repostas e atuações. As indústrias automobilísticas, marítimas e aeroespaciais são as que mais crescem mundialmente e com a ajuda de alguns adventos tecnológicos tem-se obtido produtos com mais qualidade, confiabilidade, funcionalidade e menor custo. No entanto esta tecnologia não está presente apenas no chão de fábrica, mas também em seus produtos finais. Sistemas automotivos (automóveis, ônibus, trens, etc), embarcações marítimas e aeronaves contêm um grau de sofisticação que os tornam "inteligentes". Para tanto, a quantidade de informações de naturezas diversas nestes sistemas é imensa e os hardwares que as processam e os meios físicos pelos quais estas informações circulam vem evoluindo com alto grau tecnológico.

Os comandos de um sistema de controle que necessitam de exatidão e pouco tempo de reação estão sujeitos a falhas humanas quando não automatizados e com isto alguns imprevistos podem ocorrer. Para possibilitar novas e complexas funções num sistema automotivo com confiabilidade é necessário um sistema inteligente computadorizado. Para garantir que este sistema possa atender às diversas situações possíveis simultaneamente e num tempo de resposta condizente com a aplicação automotiva, ele deve ser implementado de forma distribuído e integrado. E para isto é necessária a implementação de controladores que se comuniquem entre si, processem informações com rapidez e que possam atuar e controlar com segurança determinadas tarefas. As novas tecnologias de informação fazem uso de redes digitais de comunicação para estes fins.

Esse avanço tecnológico proporcionou inovações na comunicação de dados, como a implantação de barramento de campo (fieldbus) e equipamentos com inteligência própria. Cada vez mais se torna necessária a comunicação em tempo real e com uma qualidade de informação de grande confiabilidade.

A crescente necessidade de se implantar plantas de supervisão e controle de processos somados à sofisticação dos processos de manufatura criaram caminhos para a implementação de um grande número de pontos de coleta de dados e pontos de atuação do sistema de controle no processo. Num sistema convencional de controle seria necessária a interligação de cada um dos pontos até o centro de processamento do controle, sendo necessário uma grande quantidade de cabos e horas de instalação e manutenção.

A tecnologia Fieldbus surgiu com a necessidade da indústria em racionalizar melhor o material exigido e, também, em reduzir o tempo de instalação, reduzindo muito o custo do projeto e facilitando a manutenção. As grandes instalações, feitas com base na tecnologia convencional, consumiam muito tempo e material para interligar os sensores, botões, solenóides, etc. Neste tipo de instalação, a transmissão de informação (dados) entre dispositivos de entrada e saída é feita através de cabos multi-vias (Point-to-point wiring).

Atualmente as redes para sensores e atuadores têm um papel muito importante na automação de máquinas e sistemas, tanto nas áreas de processos, quanto na de manufatura. Na utilização de uma rede para sensores e atuadores espera-se obter uma série de vantagens econômicas e funcionais, tais como; a maior confiabilidade dos dados, compatibilidade dos módulos, modularidade, custo baixo para o seu planejamento, instalação, configuração e manutenção, através de um diagnóstico muito simplificado tornando o sistema mais flexível e de alta viabilidade econômica.

Diversos produtos com diferentes protocolos e tipos de Fieldbus surgiram no mercado. Um dos barramentos que tem sido mais utilizado, tanto por seu custo como por sua característica de evitar colisões para mensagens prioritárias durante a transmissão é o CAN (*Controller Area Network).*

O CAN, criado por Robert Bosch GmbH surgiu de modo a satisfazer a crescente necessidade, existente no mercado de automóvel, de segurança, conforto e comodidade. Essa rede é rápida, capaz de diagnosticar problemas e fazer o tratamento de erros, que o deixa altamente confiável em aplicações que envolvam vidas humanas, razão pela qual é empregado em automóveis e tem sido recomendado pelo SAE - Society of Automotive Engineers.

Devido ao protocolo aberto, além da área automotiva, o CAN é utilizado em outros ramos da indústria, como automação de plantas fabris. Alguns protocolos baseados em CAN são o CANOpen e DeviceNET.

Muitas empresas observaram esse nicho de mercado e apostaram no desenvolvimento de tecnologias para comunicação CAN, dentre elas a Motorola Industries Ltd. e a Microchip Technology Inc., que possuem circuitos integrados, assim como kits de treinamento e desenvolvimento de novas tecnologias para CAN. Para se ter uma idéia, já em 1998, aproximadamente 60 milhões de chips CAN foram fabricados [13] .

Num automóvel existem basicamente 4 aplicações principais que necessitam de comunicação em série, tendo cada uma, diferentes requisitos e objetivos: Controladores para o motor, transmissão, chassis e travamento das portas [3]. A Figura 1.1 mostra os possíveis elementos de uma rede automotiva.

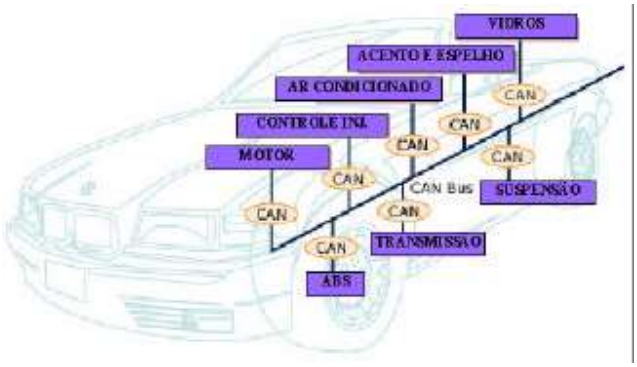


Figura 1.1-Rede CAN visto em um automóvel [4]

1.2 SINOPSE DO TRABALHO ESCRITO

O trabalho escrito é composto por 9 capítulos sendo que:

a) Capítulo 1- *Introdução*: Aborda aspectos gerias do protocolo CAN na industria e em sistemas automotivos;

b) Capítulo 2- *Rede Digital Serial*: Tutorial de redes multiplexadas e não multiplexadas definidas pelo padrão ISO e SAE;

c) Capítulo 3- *CAN*: Tutorial do protocolo CAN, onde são abordados aspectos técnicos dos protocolos e mecanismos da comunicação;

d) Capitulo 4 - *Controlador CAN*: Aborda características do controlador CAN MCP2510 da Microchip, funções de programação e temporização;

e) Capítulo 5 – *Material de Apoio*: Aborda sistemas de hardware e software utilizados para pesquisa e analise do funcionamento do protocolo CAN;
f) Capitulo 6 - *Interface CAN-PC*: Aborda o hardware padrão definido pela porta paralela do PC e que é responsável pelo intertravamento entre PC e o MCP2510;
g) Capitulo 7: *Implementação Hardware e Software*: Desenvolvimento de hardware e do software em VB para interfaceamento;
h) Capítulo 8: *Testes, Levantamento de Dados e Resultados:* Testes de comunicação entre PC simulando nó CAN, onde foram feitos testes de sincronismo, transmissão e recepção de dados; levantamentos de dados dos sistemas automotivos I (Palio – Fiat) e II (Fiesta – Ford);
i) Capitulo 9: *Conclusões e Trabalhos Futuros*: Conclusões sobre o trabalho realizado e trabalhos futuros possíveis.

1.3 OBJETIVOS

Este trabalho tem por objetivo final o desenvolvimento de um sistema de monitoramento, (hardware e software) através da rede digital CAN.
A rede CAN é hoje uma novíssima realidade no mercado para melhorar ou tornar mais inteligente um veículo motorizado, com equipamentos que se comunicam entre si, que processam informações com rapidez, e que podem atuar e controlar com segurança determinadas tarefas de um sistema. Em vista do grande interesse do mercado consumidor e de empresas por esta nova tecnologia far-se-á necessário o desenvolvimento de sistema de monitoramento capaz de auxiliar na análise das informações transmitidas através desta nova tecnologia.
O protocolo CAN está sendo empregado nos automóveis fabricados pelas indústrias automobilísticas, entre elas: CITROEN, AUDI, FORD, VOLKSWAGEN, FIAT, PEUGEOT, RENAULT e outras [6]. A complexidade da comunicação entre várias partes dos sistemas dos novos carros com projetos desenvolvidos no exterior faz com que os fabricantes nacionais não possuam o menor conhecimento sobre suas informações e seus funcionamentos. Modelos como Palio Weekend, Audi A3, Golf, Peugeot 307, Corsa, Fiesta [5], etc, já implementaram a rede fieldbus em sua eletrônica embarcada.

Mas apesar disto, nenhuma informação trafegada internamente é disponibilizada para os engenheiros nacionais.

A maior preocupação quanto a esta evolução está na manutenção destes veículos, que sem recursos humanos treinados e equipamentos especializados disponíveis no mercado estarão à mercê de oficinas mecânicas despreparadas ou de preços cada vez mais exorbitantes de concessionárias (que importam hoje poucos equipamentos destinados à manutenção destes sistemas projetados exclusivamente para suas montadoras), e que se manterão exclusivamente caso não apareçam outras alternativas. É preciso que pesquisas e desenvolvimentos nacionais sejam implementados urgentemente.

O Laboratório de Fieldbus da Universidade Federal de Itajubá (LabField), tem em suas instalações redes *fieldbuses* que utilizam este protocolo de comunicação, possibilitando realizar estudos e treinamentos. Contém também softwares que possibilitam fazer análises da informação transmitida pelo CAN as quais são enviadas por sensores.

Com ajuda de um kit de demonstração para as fases iniciais do projeto voltada ao estudo do protocolo, com posterior desenvolvimento e construção de placa dedicada à rede CAN e, ainda, com a implementação de softwares com base em programação visual para comunicação com dispositivos de interface CAN, foi possível compor um sistema totalmente padronizado muito mais adequado para todas as funcionalidades de um sistema de diagnóstico ideal. Além disto, este sistema deverá ter uma continuidade com futuros desenvolvimentos de nossas aplicações.

O barramento CAN ligado a vários módulos eletrônicos que serão responsáveis por captar, interpretar e analisar as informações relacionadas às respectivas partes controladas do automóvel é mostrado na Figura 1.2.

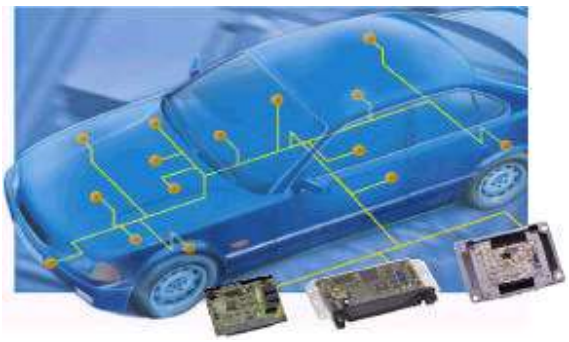


Figura 1.2-Rede CAN e seus módulos conectados [3]

Toda informação pode ser vista através um computador de bordo ou PC (Personal Computer) externo, local ou remoto, verificando assim se os módulos e peças do automóvel estão funcionando adequadamente.

Ao final deste trabalho foi possível desenvolver e disponibilizar um módulo integrado para interface entre o protocolo CAN e um canal para a comunicação com o PC de acordo com o diagrama de blocos do sistema proposto da Figura 1.3. Com ele pôde-se analisar os sinais de origens diversas que trafegam de um nó transmissor para um nó receptor. O sistema desenvolvido, a partir deste ponto citado como VisualCAN, pode capturar vários quadros de informações e armazenados em um banco de dados que por sua vez, pode ser analisado e verificado. Devido à natureza confidencial das informações o desenvolvimento exigiu uma inspeção dos dados e identificação da natureza dos quadros transmitidos.

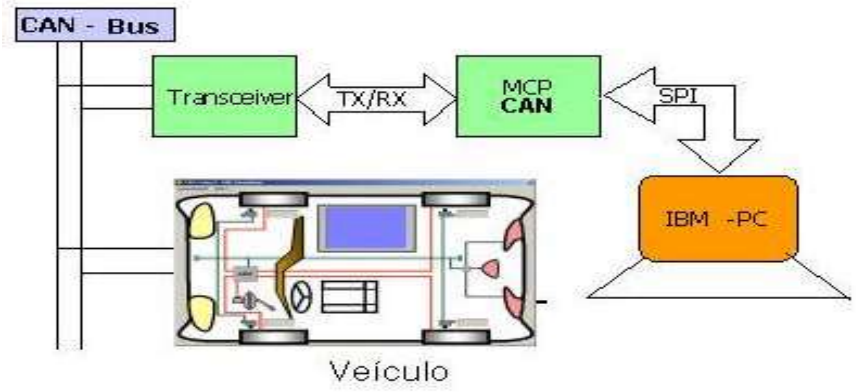


Figura 1.3-Diagrama de blocos do sistema proposto

2 CAPÍTULO 2 – REDE DIGITAL SERIAL

2.1 PROTOCOLO CAN

O CAN é uma rede de comunicação serial na qual trafegam dados com informações trocadas entre controladores distribuídos de forma modular. Cada módulo é responsável por funções específicas, mas colhem e/ou geram dados que muitas vezes devem e podem ser utilizados por outros módulos. Deste modo, cada módulo deve obtê-los, processá-los e transmiti-los através da rede de comunicação de dados de forma a compartilhá-los com todo o sistema de forma integrada. A flexibilidade dessa rede permite que seja aplicada a vários sistemas em que equipamentos precisem se comunicar ou onde existam sistemas microprocessados/ microcontrolados.

A complexidade dos sistemas de controle e a necessidade de trocar informação entre os módulos significavam cada vez mais cabeamento e linhas de controle dedicadas, implicando num maior volume de chicote do automóvel. Além do custo do cabeamento necessário para ligar todos os componentes, o tamanho físico do sistema e a complexidade que daí advém tornaram esta solução inconfortável. Os custos exagerados e o número crescente de ligações comprometiam seriamente a confiança, segurança e tolerância a falhas do sistema. Para se ter uma idéia de como esse cabeamento crescia proporcionalmente à quantidade de componentes eletrônicos que se incorporaram nos veículos, em 1921, nos primórdios da evolução dos veículos, tinha-se apenas 30 metros de fio e em 1940 - 60 metros, 1953 – 150 metros, 1980 – 800 metros e em 2000 tinha-se 2000 metros de fio, o que proporciona cada vez mais uma quantidade de fiação absurda [2].

2.2 SISTEMA NÃO MULTIPLEXADO - CONVENCIONAL

O sistema de controle convencional é realizado *point-to-point,* ou seja, cada módulo necessita ser conectado a todos os outros módulos, se houver necessidade de compartilhamento de informações. Uma mesma informação é enviada para blocos distintos através de meios distintos. Isso pode ser verificado conforme a Figura 2.1, o cabeamento é pesado.

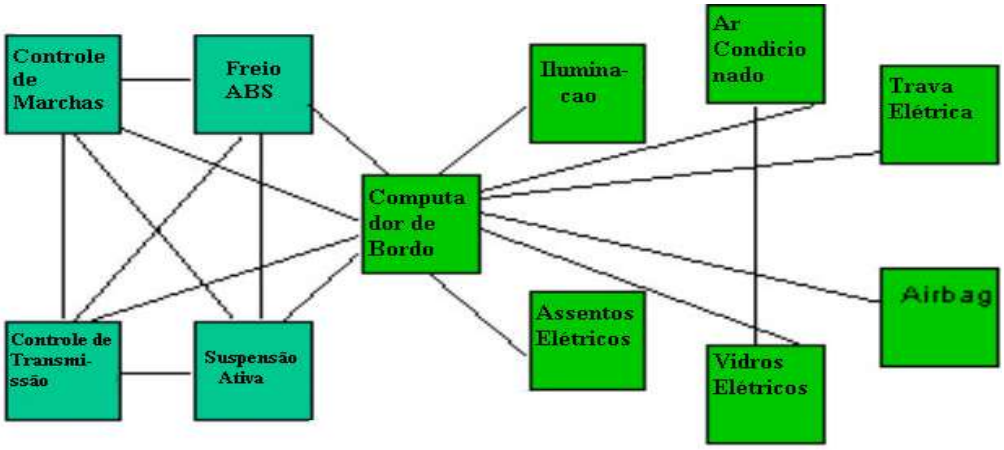


Figura 2.1 -Método Convencional: *Point-to-Point wiring* [12]

Atualmente todos os sistemas automotivos possuem uma eletrônica embarcada extremamente poderosa uma vez que o funcionamento de todas as partes deste sistema é composto de transdutores (sensores) que convertem qualquer grandeza física como a térmica, mecânica, química, etc em grandeza elétrica e transdutores (atuadores) que convertem a grandeza física elétrica em alguma grandeza física como mecânica, térmica, etc. Isto significa que uma mesma informação deverá ser transmitida independentemente para cada módulo que a necessitar.

Por estas razões as análises, manutenções, ajustes, etc, ficaram cada vez mais dependentes de equipamentos eletrônicos. Automóveis com estas características possuem conectores com dezenas de pinos para diagnósticos, onde estes estão incorporados na unidade de processamento central, e que através de códigos luminosos (piscantes) podem ser diagnosticados e testados. O Astra G99 da GM é um exemplo de sistema não multiplexado [3]. Analisando o diagrama funcional, mostrado na Figura 2.2, tem-se uma interligação entre todos os blocos de modo que a origem de uma mesma informação é transferida para diversos blocos por meios físicos independentes, caracterizando assim uma malha elétrica bastante grande. Por exemplo, uma informação de rotação é enviada para o setor de arrefecimento do motor e para o painel do carro por dois caminhos diferentes. Isto acontece também para outros vários tipos diferentes de informações.

Na eletrônica embarcada do Astra vem incorporado o protocolo de diagnóstico ISO 9141 executado a partir do conector DLC (data link conector) que será visto mais adiante.

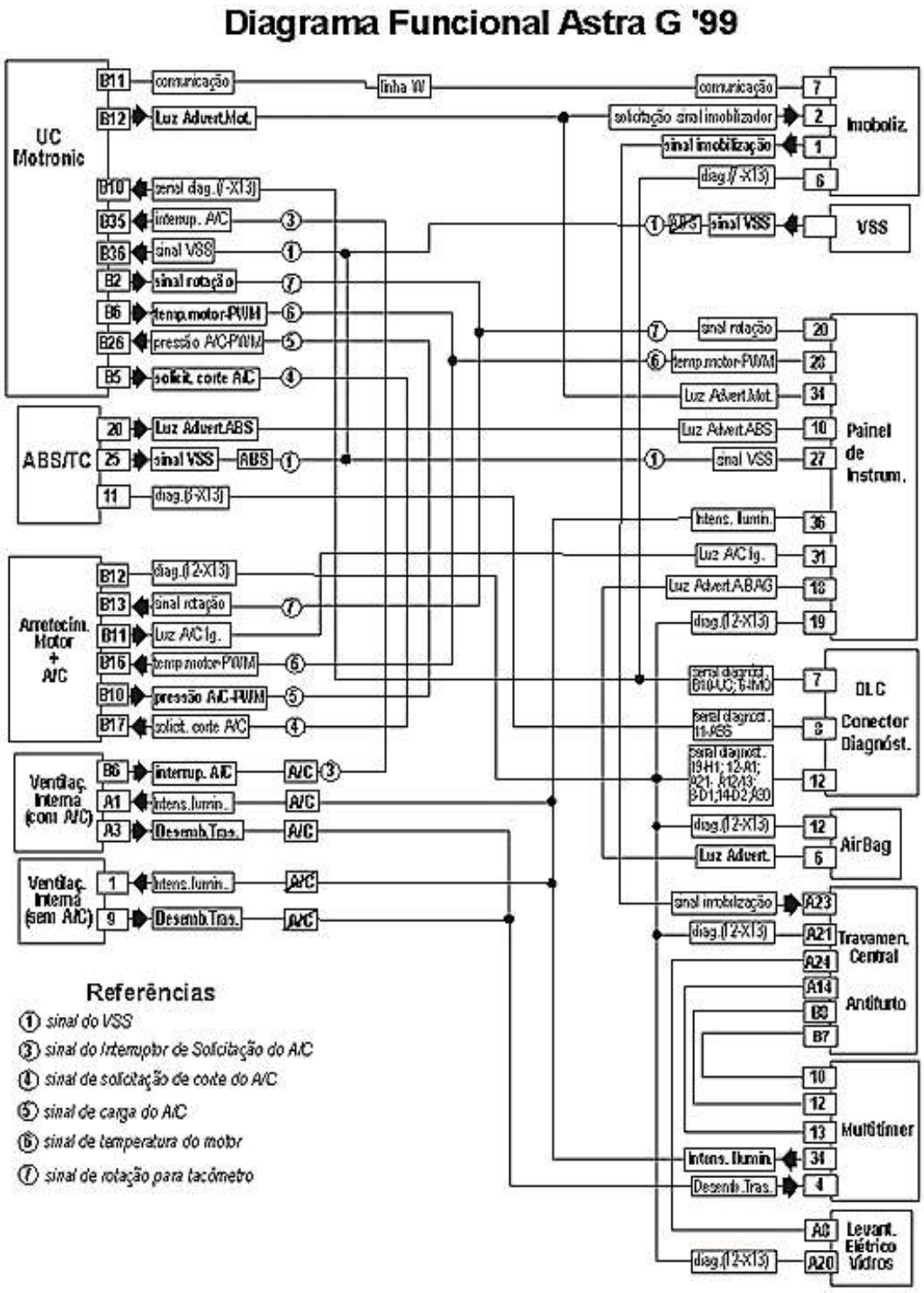


Figura 2.2-Diagrama funcional do Astra G99 - sistema não multiplexado [3]

2.3 SISTEMA MULTIPLEXADO

2.3.1 Introduzindo os protocolos para sistema multiplexado

A partir de 1994 foi implementada a rede multiplexada para emissão de diagnóstico da eletrônica embarcada OBDII "On Board Diagnostics", definida pela especificação SAE J1962, Figura 2.3. Ela possui várias opções de conexões para protocolos SAE e ISO (International Standards Organization) que dependem do país e do fabricante. Estas especificações estabelecem regras padrão apenas para diagnóstico automotivo.

J1962 Pin	J1962 Pin Description
1	Discretionary* (GMLAN SW CAN Line)
2	+ line of SAE J1850
3	Discretionary* (GMLAN MS CAN H)
4	Chassis Ground
5	Signal Ground
6	Discretionary* (GMLAN HS CAN H)
7	K Line of ISO 9141-2
8	Discretionary*
9	Discretionary* (GM ALDL)
10	- line of SAE J1850
11	Discretionary* (GMLAN MS CAN L)
12	Discretionary*
13	Discretionary*
14	Discretionary* (GMLAN HS CAN L)
15	L line of ISO 9141-2
16	Un-switched Vehicle Battery Positive

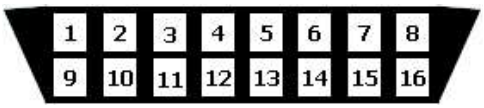


Figura 2.3 - Conector OBDII [1]

* A especificação SAE J1979 liberou estes pinos para serem utilizados pelos fabricantes automotivos.

Tabela 2.1 - Pinagem do OBDII / J1962 [1]

Os equipamentos "*scanners*" são ferramentas de varredura e capazes de enviar e receber mensagens definidas no padrão SAE J1979 (Tabela 2.1) para três tipos de barramento implementados (PWM (Pulse Width Modulation) – produtos Ford, VPW (Variable Pulse Width) – General Motor, ISO 9141-2 - Asian/European). Estes "*scanners*" necessitam ser inicializados para cada um dos três protocolos, onde códigos de diagnósticos são lidos [2].

A emissão de diagnóstico poderá ser realizada por equipamentos eletrônicos (*scanner,* por exemplo) que são ligados em terminais do conector DLC (data link conector) localizado em ponto estratégico do veículo, e por onde se processa a comunicação, utilizando o protocolo ISO 9141-2. O único requisito requerido do *scanner* é estabelecer a comunicação com a unidade desejada, através do terminal correspondente.

O novo conceito de rede CAN leva um único barramento elétrico, que percorre toda a extensão do veículo, onde dispositivos de sensoriamento e controle são a ele conectados, capazes de interagir uns com os outros, evitando um número excessivo de cabos entre as unidades de controle e os dispositivos sensores. Hoje em dia, a maioria dos fabricantes de automóveis utiliza a rede CAN.

Estas informações podem ser de temperatura, consumo, velocidade, pressão, derrapagem de pneus, frenagem brusca, acionamento de *air-bags*, sistema de alarme, iluminação, módulo danificado, etc., conforme Figura 2.4.

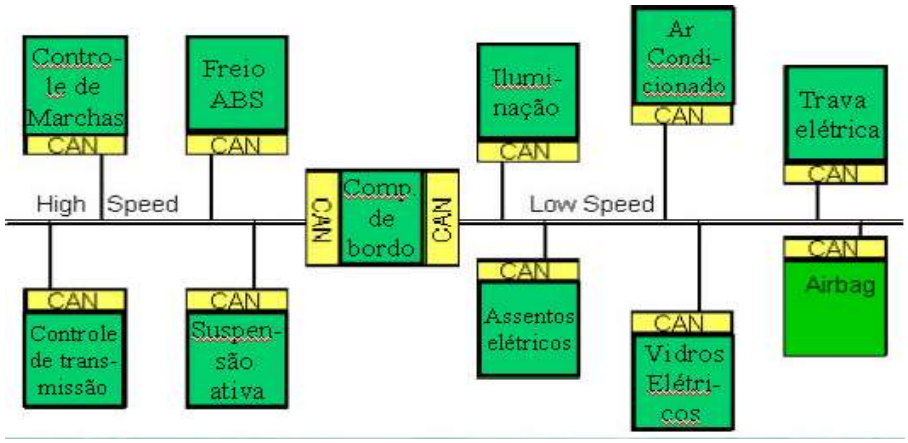


Figura 2.4-Novo conceito de rede CAN [12]

2.3.2 Arquiteturas para uso dos protocolos

Neste caso os fabricantes de veículos têm aplicado diversas soluções para disponibilizarem a informação de diagnóstico no conector:

1) As unidades de comando dos diversos subsistemas eletrônicos estão interligadas por uma rede interna CAN, através da qual trocam as informações necessárias ao seu funcionamento. Para diagnóstico, existe um conversor de protocolo entre a rede interna CAN e o conector de diagnóstico (DLC). Este conversor converte a informação de diagnóstico para o protocolo ISO 9141-2 . Somente assim pode-se ter acesso a estes

pontos de diagnóstico com o *scanner*. Em alguns casos, a rede está presente no conector de diagnóstico, mas somente para equipamentos de verificação utilizados na linha de produção. Como exemplo deste tipo de solução podem ser citados: Mercedes (carros de passeio e caminhões) e BMW até 2000. Em todos estes casos, a rede interna é CAN [3].

2) A rede interna multiplexada CAN é utilizada também para o diagnóstico via *scanner*, ou seja, o equipamento de teste se comunica com as unidades de comando através da rede interna utilizando o protocolo da mesma. Como exemplo podem ser citados os veículos das linhas GM USA e Ford USA [3].

3) Existem sistemas que possuem terminais para diagnóstico (ISO-OBDII) e uma linha específica da rede multiplexada no mesmo conector. Pode-se citar o sistema Ve.N.I.C.E, (Vehicle Net With Integrated Control) utilizado pela Fiat na linha Fire, que é um exemplo para este tipo de configuração. Analisando o diagrama funcional da Figura 2.5, percebe-se que as unidades de controle do motor, do ABS, do Air-Bag e do Fiat Code (imobilizador), disponibilizam um terminal específico para diagnóstico, no próprio conector. Para essas unidades, a comunicação é através de protocolo ISO. Não será necessário, portanto, que o equipamento de teste trate o protocolo interno (CAN). As informações de diagnóstico correspondentes às unidades apontadas acima, não trafegam pela rede CAN. O computador da carroceria (NBC-"body computer") e o painel de instrumentos (NQS) trocam informações através da rede multiplexada CAN (CANbus). Portanto, somente nestes casos o equipamento de teste deverá ter a capacidade de acessar a rede CAN **[3**].

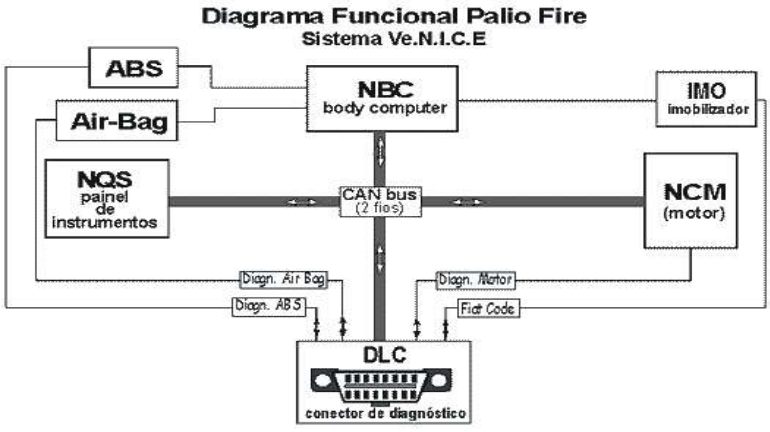


Figura 2.5-Sistema automotivo com OBDII e rede CAN [3]

Os enlaces entre equipamentos de um sistema com rede CAN não são mais ponto a ponto, pois não há necessidade de se ter cabos interconectando um mesmo sensor a várias unidades de controle e mesmo uma unidade de controle ligada diretamente às outras, como mostra a Figura 2.6. Como todos os equipamentos, sejam eles sensores ou unidades de controle, estão conectados eletricamente a um mesmo barramento, é possível que todos se comuniquem entre si por este barramento.

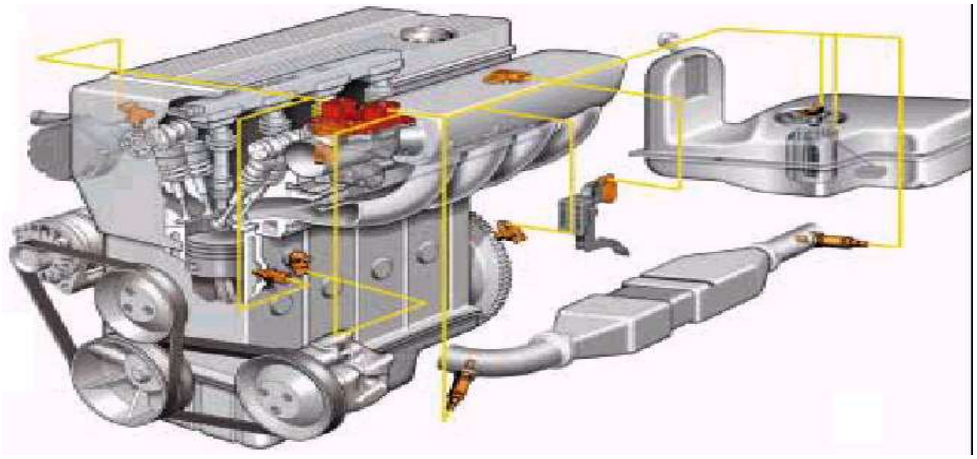


Figura 2.6-Interconexão dos módulos[2]

2.3.3 Nó no modo passivo ou ativo

Por essa razão, cada equipamento agora é denominado como "nó" e recebe um número de identificador, que já possui em sua concepção uma regra de prioridade que, junto a outros recursos, evita a colisão de dados no barramento. Este novo arranjo permite que os nós funcionem em dois modos: ou no modo Ativo, ou modo Passivo.

No modo Ativo, o nó é capaz de ouvir as informações que circulam no barramento, assim como podem interagir enviando mensagens.

No modo Passivo, o nó apenas ouve o barramento. Dessa forma, cada equipamento pode ser programado de forma a ser um simples atuador que trabalha no modo passivo, sendo que em certas situações de controle muda para o modo ativo e passa a ser um sensor ou até mesmo um controlador, o que permite entre outras características, a redundância entre unidades de controle.Vantagens da arquitetura com a rede CAN

Pode-se listar abaixo alguma das vantagens de se utilizar à transmissão pela rede CAN [13]:

- É um protocolo de comunicação padrão que se comunica como uma rede multiplexada, reduzindo bastante o tamanho da estrutura e elimina a instalação elétrica ponto a ponto, permitindo um sistema mais inteligente em aplicações que exijam redundâncias de mestres.
- Protocolo multimestre que utiliza o NON-Destructive Collision Resolution;
- Sofisticado mecanismo de detecção de erro e retransmissão dos dados, garantindo aproximadamente 100% de confiabilidade na transmissão de dados.
- Mais de 15 fabricantes de chips para a rede CAN a preços competitivos em virtude de ser um protocolo padrão;
- Redução do número e volume de chicotes, tendo um máximo comprimento de 40 m para uma transmissão de até 1 Mbit/s;
- Sistema de mensagem *Broad-cast* onde todo nó recebe todas as mensagens e participa da verificação de erros, ou seja, é feita uma varredura de erro antes delas serem aceitas;
- Utiliza *bitwise arbitration*, isto é, um dispositivo pode transmitir a qualquer momento quando o barramento estiver ocioso (CSMA). Em caso de colisão, o bit 0 no identificador é dominante, definindo assim a prioridade dos dispositivos;
- Redução do tempo de montagem;
- Melhor interação entre os módulos de controle;
- Possibilidade de diagnóstico
    - Para Produção
    - Para Assistência técnica

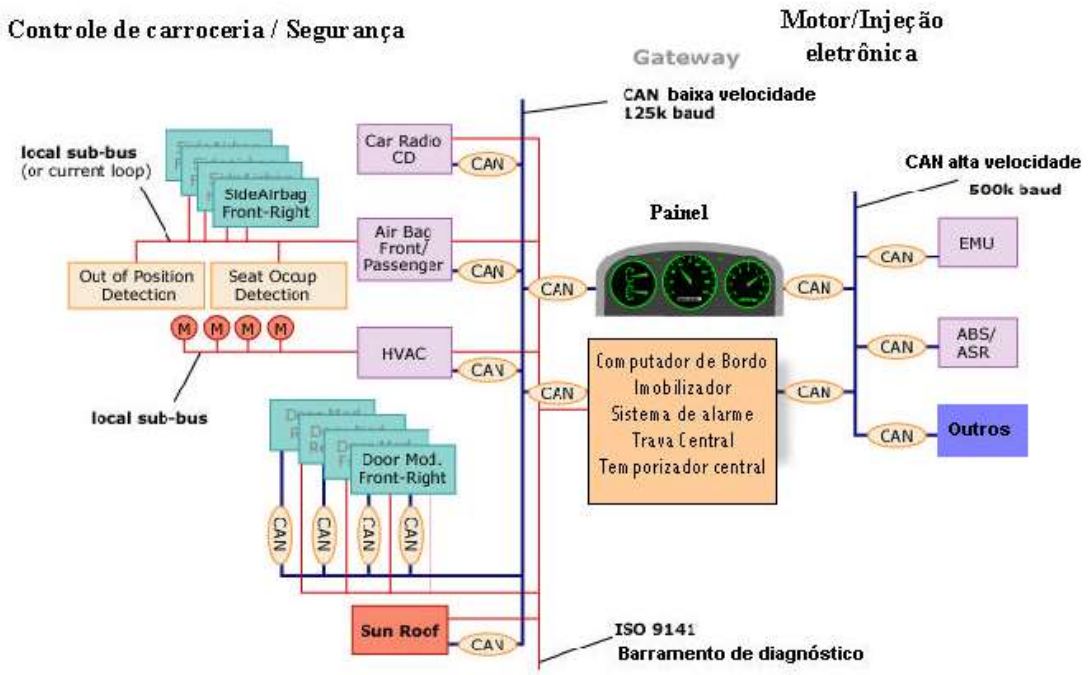


Figura 2.7-Rede CAN no Automóvel[4]

A Figura 2.7 mostra vários barramentos multiplexados, inclusive o CAN de alta e o de média velocidade, ligados a vários módulos eletrônicos, que serão responsáveis por captar, interpretar e analisar as informações relacionadas às respectivas partes controladas do automóvel. Toda informação pode ser vista através um computador de bordo ou PC externo local ou remoto, que verifica se os módulos e peças do automóvel estão funcionando adequadamente.

2.4 REVOLUÇÃO DO BARRAMENTO MULTIPLEXADO

A multiplexação de dados elétricos automotivos em barramento serial digital iniciou-se a partir de 1970. Gradativamente houve uma evolução que acarretou a normatização SAE em três classes A, B e C.

Classe A – Utilizado para comunicação de propósito geral, carroceria, áudio, etc e não utilizado para diagnóstico. Com uma baixa taxa de transferência, menor que 10Kb/s e que deve suportar transmissão de mensagem dirigida por evento. A

Tabela 2.4 mostra a maioria dos atributos de alguns protocolos da classe A.

NOME:	USUÁRIO:	USO:	MODELOS ANOS:	COMENTÁRIOS:
UART	GM	Muitos	19985 – 2005+	Sendo implementado
Sinebus	GM	Áudio	2000+	Rádio / Controle de direção
I²C	Renault	HVCA	2000+	Pouco utilizada
J1708/J1587/J1962	T&B	Geral	1985 – 2002+	Sendo implementado
ACP	Ford	Äudio	1985 – 2002+	
BEAN	Toyota	Geral	1995+	
UBP	Ford	Conexão	2000+	
LIN	Muitas empresas	Conector inteligente	2003+	Desenvolvido por um consórcio de empresas

Tabela 2.2- Alguns protocolos da classe A [11]

A Tabela 2.3 mostra a comparação dos atributos para os protocolos na classe A.

	NOME DO BARRAMENTO							
CARACTERÍSTICAS	UART	SINEBUS	E&C	I²C	SAE J1708	ACP	BEAN	LIN
ORIGEM	GM	DELCO	GM	PHILIPS	TMC-ATA	FORD	TOYOTA	MOTOROLA
APLICAÇÃO	GERAL & DIAGNÓSTICO	ÁUDIO	GERAL		CONTROL. & DIAGNÓSTICO	CONTROL. DE AUDIO	CONTROL. CARROCERIA & DIAGNÓSTICO	SENSORES INTELIGENTES
CODIFICAÇÃO DE BIT	NRZ	SAM	PWM	AM	NRZ	NRZ	NRZ	NRZ
MEIO FÍSICO	CABO	CABO	CABO	PAR TRANÇADO	PAR TRANÇADO	PAR TRANÇADO	CABO	CABO
MEIO DE ACESSO	MASTER/ SLAVE	MASTER/ SLAVE	CAPTURA	CAPTURA	MASTER/ SLAVE	MASTER/ SLAVE	CAPTURA	MASTER/ SLAVE
DETECÇÃO DE ERRO	8BIT CS	NONE	PARITY	ACK BIT	8 BIT CS	8 BIT CRC	8 BIT CRC	8 BIT CS
COMPRIMENTO DO "HEADER"	16 BITS	2BITS	11-12 BITS		16 BITS	12 – 24 BITS	25 BITS	2 BITS/BYTE
COMPRIMENTO DA INFORMAÇÃO	0 – 85 BYTES	10-18 BITS	1-8 BITS			6 – 12 BYTES	1 – 11 BYTES	8 BYTES
OVERHEAD	VARIÁVEL	75%	VARIÁVEL	45%	VARÁVEL	25%	28%	2 BYTES
VELOCIDADE DE TRANSFERÊNCIA	8192 B/S	66.6KB/S 200KB/S	1KB/S	1-100KB/S	9600 B/S	9600 B/S	10KB/S	1-20KB/S
COMPRIMENTO DO BARRAMENTO	NÃO ESPECIFICADO	10 METROS	20 METROS	NÃO ESPECIFICADO	NÃO ESPECIFICADO	40 METROS	NÃO ESPECIFICADO	40 METROS
MÁXIMO DE NÓS	10		10			20	20	16
NECESSIDADE DE MICROPROC. ?	SIM	NÀO	SIM		SIM	SIM	SIM	NÃO
CUSTO	BAIXO	BAIXO	BAIXO		MEDIO	BAIXO	BAIXO	BAIXO

Tabela 2.3- Comparação dos protocolos da classe A [11]

A maioria dos protocolos é implementada de forma simples e de baixo custo, muitas vezes não exigindo hardware microprocessado [11].

Classe B – Utilizado em uma vasta maioria de comunicações não criticas e não usadas em diagnóstico. Suporta mensagem orientada por evento e por período. Sua taxa de transferência de dados está entre 10Kb/s e aproximadamente 125Kb/s. O custo gira em torno de duas vezes em relação a classe A. A

Tabela 2.4 lista os protocolos utilizados na classe B.

NOME:	USUÁRIO:	USO:	MODELOS ANOS:	COMENTÁRIOS:
GMLAN (low)	GM	Varias partes	2002+	Apenas a GM, J2411 cabo simples CAN
GMLAN (mid)	GM	Informações	2002+	95.2Kb/s
Ford MSCAN	Ford	Varias	2004+	125Kb/s; J2284
DCX LSCAN	Chrysler	Varias	2004+	125Kb/s; ISO 11519
ISO 11898	Europa	Varias	1992+	Varias velocidades – 47.6Kb/s – 500Kb/s
J2284	GM, Ford, DC	Varias	2001+	500Kb/s ; baseado na ISO 11898
J 1939	T&B	Varias	Até 2002+	J1850
Intellibus	tbd			Industria de aviões

Tabela 2.4 – Comparação de alguns protocolos da Classe B [11]

A palavra padrão nesta classe é ainda CAN. Em particular, ISO 11898 com uma taxa 100Kb/s para aplicações em carros de passeio e J1939 com uma taxa de 250Kb/s para aplicações em caminhões e ônibus [11]. A Tabela 2.5 compara a maioria dos atributos dos diversos protocolos da Classe B.

	NOME DO BARRAMENTO					
CARACTERÍSTICAS	CAN 2.0 ISO 11898 ISO 11519-2 J2284	J1850 ISO 11519-4			SAE J1939	INTELLIBUS
ORIGEM	BOSCH/SAE/ISO	GM	FORD	CHRYSLER	TMC-ATA	BOENG/SAE
APLICAÇÃO	CONTROLEL & DIAGNÓSTICO	GERAL & DIAGNOSTICO	GERAL & DIAGNÓSTICO	GERAL & DIAGNÓSTICO	CONTROL. & DIAGNÓSTICO	CONTROLE & DIAGNÓTICO
CODIFICAÇÃO DE BIT	NRZ MSB FIRST	VPW MSB FIRST	PWM MSB FIRST	VPW MSB FIRST	NRZ MSB FIRST	MANCHESTER BI-PHASE
MEIO FÍSICO	PAR TRANÇADO	CABO	PAR TRANÇADO	CABO	PAR TRANÇADO	PAR TRANÇADO
MEIO DE ACESSO	CAPTURA	CAPTURA	CAPTURA	CAPTURA	CAPTURA	MASTER/ SLAVE
DETECÇÃO DE ERRO	CRC	CRC	CRC	CRC	CRC	CRC, PARIDADE
COMPRIMENTO DO “HEADER”	11 OU 29 BITS	32 BITS	32 BITS	8 BITS	29 BITS	16 – 48 BITS
COMPRIMENTO DA INFORMAÇÃO	0- 8 BYTES	0-8 BYTES	0 – 8 BYTES	0 – 10 BYTES	8 BYTES	0 – 32 BYTES
VELOCIDADE DE TRANSFERÊNCIA	10Kb/s até 1Mb/s	10.4 Kb/s	41.6Kb/s	10.4Kb/s	250Kb/s	12.5Kb/s
COMPRIMENTO DO BARRAMENTO	NÃO ESPECIFICADO 40 m (típico)	35 METROS 5m para SCANNER	35 METROS 5m para SCANNER	35 METROS 5m para SCANNER	40 METROS	30 METROS
MÁXIMO DE NÓS	32 (TIPICO)	32	32	32	30	64
NECESSIDADE DE MICROPROC. ?	SIM	NÃO	SIM		SIM	NÃO
CUSTO	MÉDIO	BAIXO	BAIXO	BAIXO	MÉDIO	MÉDIO

Tabela 2.5 – Comparação dos diversos protocolos na Classe B [11]

Classe C – Protocolos de alta velocidade com taxas de 125Kb/s a 1Mb/s utilizados para comunicação com sistemas como motor, freio ABS, etc. Portanto deverá suportar

transmissão de parâmetros periódicos em tempo real (poucos mili-segundos). Com um custo de 3 a 4 vezes por nó com relação à Classe A. A Tabela 2.6 mostra os protocolos mais usuais da Classe C.

NOME:	USUÁRIO:	USO:	MODELOS ANOS:	COMENTÁRIOS:
GMLAN (high)	GM	Todas partes	2002+	500Kb/s; J2284
HSCAN	Ford	Varias	2004+	500Kb/s; J2284
HSCAN	Chrysle	Varias	2004+	125Kb/s; J2284
ISO 11898	Europa	Maioria	1992+	Varias velocidades – 47.6Kb/s – 500Kb/s
J 1939	T&B	Maioria	1994+	250 Kb/s CAN

Tabela 2.6 – Protocolos da Classe C [11]

A maioria das aplicações em carros de passeio roda com protocolo ISO 11898 a 500Kb/s na rede Classe C. J1939 é utilizado tanto para Classe B e Classe C para aplicações em caminhões, ônibus, construção, agricultura, marítima e outras. A maior diferença das aplicações da Classe B da Classe C é o tipo de nó que estão interligados. A Tabela 2.7 mostra a comparação dos diversos protocolos utilizados na Classe C.

	NOME DO BARRAMENTO		
CARACTERÍSTICAS	CAN 2.0 ISO 11898 ISO 11519-2 J2284	SAE J1939	INTELLIBUS
ORIGEM	BOSCH/SAE/ISSO	TMC-ATA	BOENG/SAE
APLICAÇÃO	CONTROLEL & DIAGNÓSTICO	CONTROL. & DIAGNÓSTICO	CONTROLE & DIAGNÓTICO
CODIFICAÇÃO DE BIT	NRZ MSB FIRST	NRZ MSB FIRST	MANCHESTER BI-PHASE
MEIO FÍSICO	PAR TRANÇADO	PAR TRANÇADO	PAR TRANÇADO
MEIO DE ACESSO	CAPTURA	CAPTURA	MASTER/ SLAVE
DETECÇÃO DE ERRO	CRC	CRC	CRC, PARIDADE
COMPRIMENTO DO "HEADER"	11 OU 29 BITS	29 BITS	16 – 48 BITS
COMPRIMENTO DA INFORMAÇÃO	0- 8 BYTES	8 BYTES	0 – 32 BYTES
VELOCIDADE DE TRANSFERÊNCIA	10Kb/s até 1Mb/s	250Kb/s	12.5Kb/s
COMPRIMENTO DO BARRAMENTO	NÃO ESPECIFICADO 40 m (típico)	40 METROS	30 METROS
MÁXIMO DE NÓS	32 (TIPICO)	30	64
NECESSIDADE DE MICROPROC. ?	SIM	SIM	NÃO
CUSTO	MÉDIO	MÉDIO	MÉDIO

Tabela 2.7 –Comparação dos diversos protocolos da Classe C [11]

Protocolos de Emissão de Diagnósticos

São protocolos usados desde 1994 destinados a diagnósticos e ajustes nas diversas partes do carro chamados de OBD. A Tabela 2.8 mostra os protocolos mais utilizados e seus usuários.

NOME:	USUÁRIO:	USO:	MODELOS ANOS:	COMENTÁRIOS:
ISO 15765-4	EUROPA	E-OBD	2000+	E-ODB CAN
ISO 15765-4	TODOS	ODB-III	2007+	OBD HARMONIZADO
J 1850	GM, FORD, DC	ODB-II	1994+	NÃO ACEITO NA EUROPA
ISO 9141-2	EUROP, ASIA, ALGUMAS NOS E.U.A.	ODB-II	1994+	VELHO OBD-II UART
ISO 14230-4	MUITAS	OBDII, ODB-III	2000+	KEYWORD 2000

Tabela 2.8 – Protocolos mais utilizados para emissão de diagnóstico [11]

Dados referentes diagnósticos são transmitidos pelas centrais eletrônicas de modo serial utilizando as tecnologias de modulação pulsada (SAE J1850 ou ISO 11519-4) onde a GM utiliza para carros de passageiros e caminhonetes, o VPW e a Ford utiliza o PWM para comunicação padrão. Os produtos da Europa além dos importados da Ásia, utilizam o ISO 9141 [11]. A Tabela 2.9 mostra as comparações de diversos protocolos utilizados na emissão de diagnósticos. Nos Estados Unidos o CAN de alta velocidade é chamado de OBDIII.

	NOME DO BARRAMENTO					
CARACTERÍSTICAS	**ISO 15765**	**J1850 ISO 11519-4**			**ISO/DIS 9141 ISO/DIS 9141-2**	**KEYWORD XX (71, 72, ETC)**
ORIGEM	ISO	GM	FORD	CHRYSLER	MUNDIAL	VARIAS
APLICAÇÃO	EMISSÃO DE DIAGNÓSTICO	GERAL & DIAGNOSTICO	GERAL & DIAGNÓSTICO	GERAL & DIAGNÓSTICO	APENAS DIAGNÓTICO	DIAGNÓTICO
CODIFICAÇÃO DE BIT	NRZ	VPW MSB FIRST	PWM MSB FIRST	VPW MSB FIRST	NRZ (strt, 7D,P,stop) LSB first	NRZ
MEIO FÍSICO	PAR TRANÇADO	CABO	PAR TRANÇADO	CABO	PAR TRANÇADO	CABO
MEIO DE ACESSO	CAPTURA	CAPTURA	CAPTURA	CAPTURA	EQUIPAMENTO SCANNER/SLAVE	MASTER/ SLAVE
DETECÇÃO DE ERRO	CRC	CRC	CRC	CRC	PARIDADE (PAR)	CRC, PARIDADE
COMPRIMENTO DO "HEADER"		32 BITS	32 BITS	8 BITS	NÃO ESPECIFICADO	16 BITS
COMPRIMENTO DA INFORMAÇÃO		0-8 BYTES	0 – 8 BYTES	0 – 10 BYTES	NÃO ESPECIFICADO	0 – 85 BYTES
VELOCIDADE DE TRANSFERÊNCIA		10.4 Kb/s	41.6Kb/s	10.4Kb/s	< 10.4 Kb/s	5 b/s – 10.4Kb/s
COMPRIMENTO DO BARRAMENTO		35 METROS 5m para SCANNER	35 METROS 5m para SCANNER	35 METROS 5m para SCANNER	LIMITADO PELO TOTAL DE Z PARA TERRA	NÃO ESPECIFICADO
MÁXIMO DE NÓS		32	32	32	LIMITADO PELO TOTAL DE Z PARA TERRA	64
NECESSIDADE DE MICROPROC. ?		NÀO	SIM		SIM	SIM
CUSTO		BAIXO	BAIXO	BAIXO	BAIXO	BAIXO

Tabela 2.9 – Comparação dos protocolos de emissão de diagnósticos [11]

3 CAPÍTULO 3 - CAN

3.1 MENSAGENS DO CAN

Toda informação que trafega pelo barramento é definida por uma prioridade que a torna mais ou menos importante, de forma que a cada instante em caso de outra(s) informação(ões) necessitar(em) de transmissão, somente a mais urgente terá sucesso. A prioridade, conforme a norma CAN é estabelecida pelo campo de identificação da mensagem. Este campo pode ter 11 bits (padrão) ou 29 bits (estendido).

O protocolo CAN apresenta diversos padrões para suas mensagens de acordo com suas funções. As mensagens são enviadas em quadros (*frames*) e neles estão informações desde quem as enviou até o que está sendo enviado ou recebido. Os *frames* mais importantes são *Data Frame* e o *Remote Frame.*

3.2 FORMATO DAS MENSAGENS CAN

O CAN utiliza pequenas mensagens, com um máximo de dados de 64 bits. Não tem qualquer endereço explícito nas mensagens, sendo através do seu conteúdo que a estação receptora reconhece a mensagem. Este mecanismo denomina-se de endereçamento orientado ao conteúdo. Ele identifica a variável a ser transmitida e não o endereço a onde esta variável está.

Existem duas versões do CAN em uso até o momento: 1.0 e 2.0 (Figura 3.1)

A versão 2.0 tem duas variantes: a 2.0A (*Standard*) e a 2.0B (estendido).

O CAN 2.0A é completamente compatível com a versão 1.0, pois nas duas versões os identificadores têm 11 bits de comprimento. Enquanto que na versão 2.0B podem ter tanto identificadores de 11 bits como identificadores de 29 bits. De modo a manter a compatibilidade com as versões anteriores, o 2.0B pode ser tanto do tipo passivo como ativo.

Dispositivos ativos trabalham somente com dispositivos de 29 bits.

Dispositivos passivos trabalham com dispositivos de 11 e de 29 bits, mas a rede poderá identificar apenas os identificadores com os 11 bits, desprezando as diferenças dos outros 18 bits.

Os controladores CAN 1.0 geram erros sempre que recebem um *frame* com formato estendido.O formato do *frame* de mensagens Standard 2.0A é composto por sete campos

principais. Uma mensagem do tipo *Standard* começa com o campo início de *frame*, um bit dominante (0 lógico) que identifica o início do *frame*, seguido pelo campo de arbitragem, que contém o identificador de 11 bits. Logo após vem o bit RTR (pedido de transmissão remoto), que indica se é um *frame* de dados, se o bit for dominante, ou um *frame* de pedido (*frame* remoto), se o bit for recessivo.

Um *frame* remoto é emitido sempre que um nó necessita de informação de outro nó da rede, não contendo qualquer informação no campo de dados.

O campo de controle contém o bit IDE (identificador de extensão), que indica se o *frame* tem o formato *Standard* (se for dominante), ou o formato estendido (se for recessivo), seguido de um bit reservado para futuras extensões do formato. Os últimos 4 bits, DLC (tamanho dos dados), indicam o número de bytes no campo de dados.

O Campo de dados contém a informação propriamente dita e varia de tamanho entre 0 e 8 bytes. Ele é seguido pelo campo de CRC, que tem um tamanho de 15 bits, e é usado para detecção de erros. O campo de ACK é constituído por dois bits, o primeiro é transmitido com o valor recessivo, sendo subseqüentemente retransmitido com o valor dominante por todos os nós que receberam a mensagem corretamente, independentemente do resultado do teste de aceitação. O fim da mensagem é indicado pelo campo EOF que consiste em sete bits recessivos.

No final de cada frame são enviados 3 bits recessivos, designados intermissão, de modo a separar dois *frames* consecutivos. Se nenhuma estação quiser acessar ao *bus*, ele fica inativo.

O formato de *frame* estendido 2.0B, proporciona identificadores de 29 bits. Este novo identificador é constituído pelos 11 bits já existentes (identificador base), seguido de uma extensão de 18 bits (extensão do identificador).

A distinção entre os dois formatos é conseguida usando o bit IDE que será dominante no caso de o *frame* ter o formato *Standard* e recessivo se o *frame* for do tipo estendido. O bit RTR é substituído pelo bit SRR, sempre transmitido com o valor recessivo de modo a assegurar a prioridade dos *frames Standard* no caso da arbitragem entre os dois *frames* de formato diferentes e com o mesmo identificador base.

Ao contrário da versão 2.0A, na versão 2.0B o bit IDE é seguido pelo identificador de extensão de 18 bits, pelo bit RTR e por um bit reservado r1. Todos os campos seguintes são idênticos da versão 2.0A.

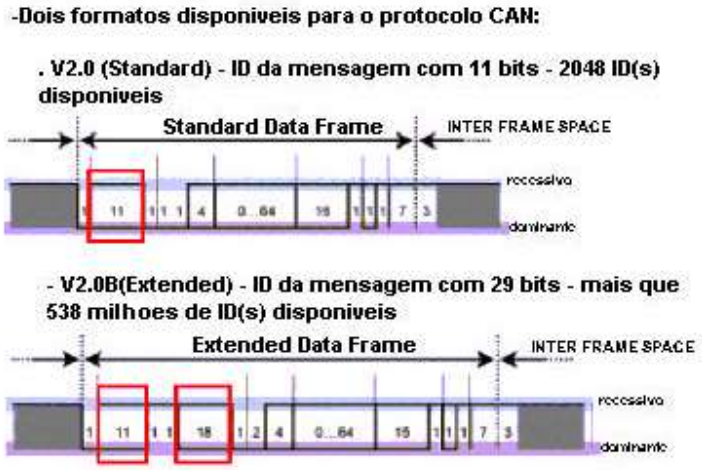


Figura 3.1-Versão do Protocolo CAN[13]

3.3 O FRAME DE DADOS (DATA FRAME)

O *Data Frame*, conforme a Figura 3.2, é um quadro de informação responsável por enviar dados que serão utilizados para atuar em algum módulo. Os dispositivos receptores dos dados realizarão determinada tarefa exigida após interpretar os dados recebidos.

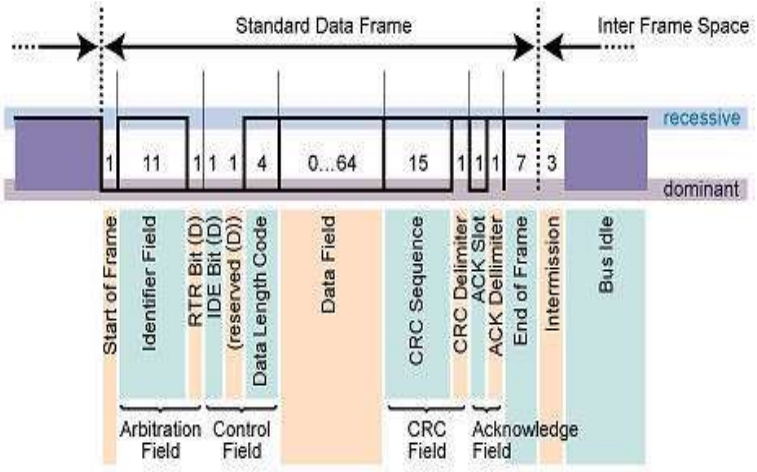


Figura 3.2-Formato de mensagem enviando dados[13]

Os vários campos do "*data frame*" indicam a posição da seqüência que será lida a mensagem por um dispositivo CAN.Em breves intervalos de tempo, o CAN-Bus de

dados transmite um protocolo de união de dados entre os nós. Para isso pode-se subdividir em 7 partes o *frame* de dados:

Start of Frame (SOF)

Arbitration Field,

Control Field,

Data field,

CRC Field,

ACK Field,

EOF.

- ***Start of Frame (SOF) = 1 bit (dominante):*** Indica o começo da mensagem. Depois de um *idle-time* (barramento inativo), na descida do edge, ele é usado para uma sincronização dos diferentes nós na rede. A alta sincronização é requerida sempre que existe um *edge* do recessivo para o dominante. Isso acontece também quando se tem uma suspensão na transmissão e o último bit do *interframe space.*
- ***Arbitration Field (11 bits padrão e 29 estendido):*** O comprimento do identificador no formato padrão é 11 bit e corresponde na base ID no formato padrão. O identificador é seguido pelo bit RTR. No *Data Frame* o bit RTR será dominante. Dentro do Remote Frame o bit RTR tem que ser recessivo. A Base ID é seguido pelo bit IDE (Identifier Extension); transmitido dominante no formato padrão (dentro do Control Field) e recessivo no Formato Estendido. Desta forma o formato padrão prevalece o formato Estendido no caso de colisão. Pela Figura 3.3 pode-se identificar a diferença do formato padrão do formato estendido.

CAMPO DE ARBITRAGEM (ARBITRATION FIELD) :

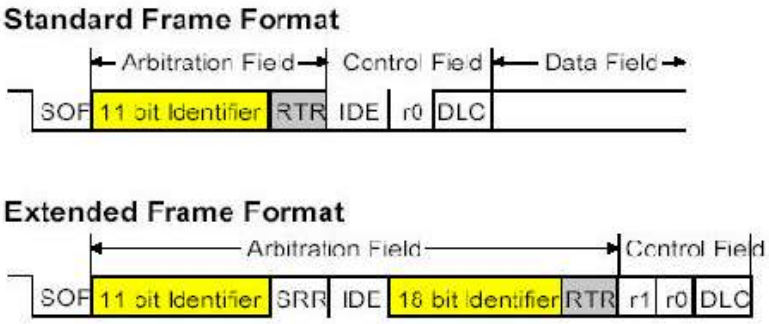


Figura 3.3-Campo de Arbitrariedade[13]

O formato Estendido inclui duas seções: Base ID com 11 bits e o Estendido ID com 18 bits. O bit SRR (*Substitute Remote Request*) substitui o bit RTR e é transmitido recessivo. Se o SRR é corrompido e transformado dominante, o recessivo irá ignorar este. Mas se o valor não é ignorado pelo *bit Stuffing* é arbitrário. Desde que o bit SRR seja recessivo antes do bit IDE, um bit recessivo não poderá decidir instantaneamente se ele é um bit recessivo RTR ou um SRR. Isto significa que apenas o bit IDE, decide se o frame é padrão ou um frame estendido. Em todas as implementações não serão requeridos todos os bits da extensão do identificador.

- ***Control Field = 6 Bits:*** O formato do *Control Field* é similar ao formato padrão e estendido. O *Control Field* no formato padrão exclui o *Data Length Code* (DLC), o bit IDE ao qual é transmitido dominante e o bit reservado r0 também é transmitido dominante. O *Control Field* no formato estendido exclui o DLC e dois bits reservados r1 e r0, conforme a Figura 3.4. Os bits reservados têm de ser enviados dominantes, mas aceitam receber bits dominantes e recessivos em todas combinações.

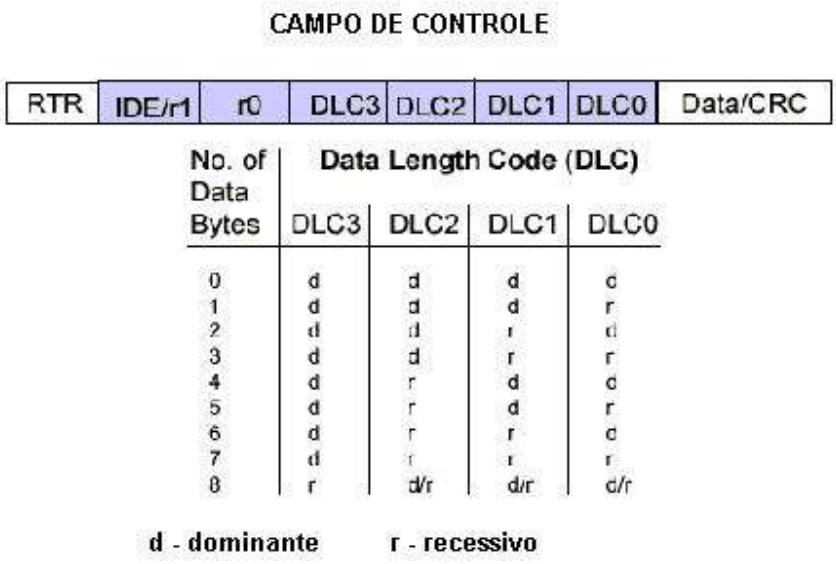


Figura 3.4-Campo de controle[13]

O formato Estendido inclui duas seções: Base ID com 11 bits e o Estendido

O número de bytes no campo de dados é identificado pelo *data length code (DLC)* no qual são quatro bits transmitidos pelo *Control Field.*O número máximo de dados no Data Frame é de 0 a 8 Bytes. DLCs na fileira de 0 a 7 indica comprimento de bytes de 0 a 7. Todos os outros DLCs indicam 8 bytes. Isto significa que os DLCs estendem-se de 9 para 15 que poderão ser usados para aplicações de propósitos específicos.

- ***Data Field = 0 to 64 Bits (0 to 8 Bytes):*** Em muitas aplicações ele é usado para transmitir dados, conforme a Figura 3.5.

Figura 3.5-Campo de dados[13]

- ***CRC Field = 16 Bits (incluindo CRC Delimiter = high (recessive)):*** O CRC contém uma seqüência de 15 bits e um recessivo do *CRC* limitador conforme a Figura 3.6. O frame checa se a seqüência é derivada de um código de redundância cíclica adaptado para *frames* com um contador menor que 127.

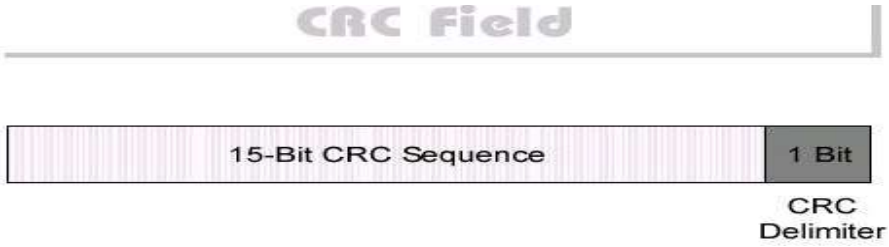


Figura 3.6-Campo CRC[13]

O bit recessivo (CRC delimiter) calcula o CRC do mesmo modo que um transmissor, como:

1. A mensagem é estimada como polinômio e é dividida pelo polinômio gerador: $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$
2. O resto da divisão do módulo 2 é uma seqüência CRC o qual é transmitido junto com a mensagem.
3. O receptor divide a mensagem inclusive a seqüência CRC pelo polinômio gerador.

- Um erro CRC é gerado se o resultado do cálculo não for o mesmo que foi recebido na seqüência CRC. Neste caso o receptor descarta a mensagem e transmite um *Error Frame* para requerer uma retransmissão. ***ACKNOWLEDGE FIELD = 2 Bits (incluindo ACK Delimiter = high (recessive)):*** O campo ACK é dois bits e contém o *ACK Slot* e o *ACK delimiter* conforme mostra a Figura 3.7. O transmissor do frame transmite ambos os bits do campo recessivo ACK. O receptor, o qual recebeu uma mensagem válida correta, anuncia isto para o transmissor enviando um bit dominante durante o *ACK Slot*. Se o transmissor detecta um acknowledge positivo, que é um dominante *ACK Slot,* o transmissor sabe que no mínimo uma mensagem dele esta correta.

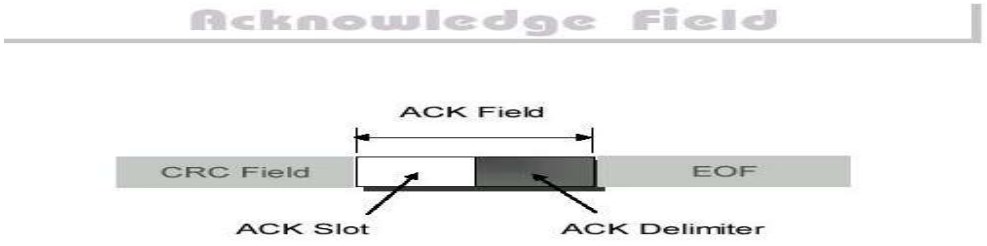


Figura 3.7-Campo do *Acknowledge*[13]

- ***End of Frame, EOF Field = 7 Bits = high (recessive):*** Cada *Data* e *Remote Frame* é delimitado por uma seqüência de flag de sete bits recessivos. Este EOF foi introduzido porque um *Error Frame* causava uma falha global CRC que deveria ser transmitida junto com o comprimento do *Data* ou *Remote Frame.*

3.4 REMOTE FRAME

O *Remote Frame*, conforme Figura 3.8, funciona como uma requisição de serviço ou pedido de informação de um ou vários dispositivos e/ou módulos de controle do sistema. Por isso ele não apresenta o campo de dados. O protocolo CAN pode ser utilizado com nenhum (sistema totalmente distribuído), um ou vários mestres controlando o sistema automotivo. Uma de suas maiores características é o uso de controle de acesso baseado no método publisher/subscriber. No entanto, pode também trabalhar com outros métodos.

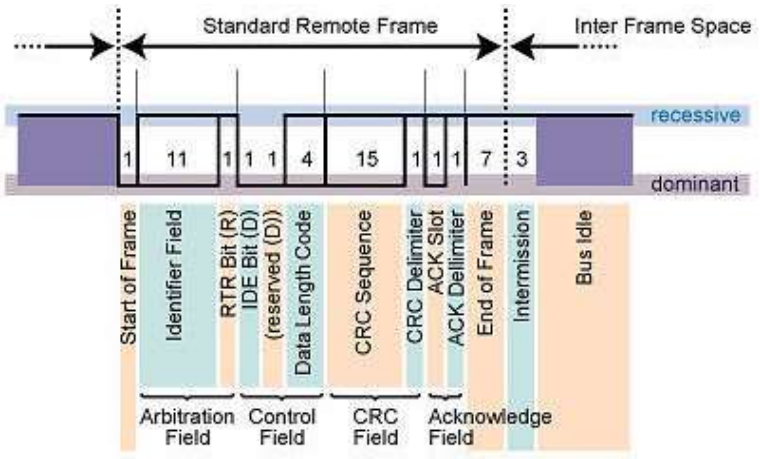


Figura 3.8-Mensagem de requisição de informação - *Remote Frame*[13]

3.5 CAN E AS 7 CAMADAS DO MODELO OSI DA ISO

A comunicação é a transferência de informações entre dois pontos. Para se estabelecer uma comunicação é preciso ter um transmissor, um receptor e um meio pelo qual os dados irão trafegar. Para uma grande variedade de equipamentos e fabricantes é preciso seguir alguma padronização para a troca de dados, e isto não significa estar padronizando uma rede *Fieldbus* como um todo, mas sim, de níveis mais básicos que envolvem uma

comunicação como, por exemplo, os níveis dos sinais elétricos em um cabo de interligação entre dois equipamentos.

No modelo de comunicação desenvolvido pela ISO o chamado "*Open Systems Interconnection"*, com a ajuda do qual pode-se ter uma boa idéia como um Fieldbus é estruturado. O modelo ISO é popular porque fornece uma explicação simples das relações entre o hardware complexo e os componentes de protocolo de uma rede. No modelo ISO, a mais baixa camada corresponde ao hardware, e as camadas sucessivas correspondem ao firmware ou software que usam o hardware [15].

Esse modelo foi idealizado para estruturar redes e aplicativos em computadores, mas analogicamente a rede Fieldbus, ele também efetua troca de dados e pode-se aproveitar alguns preceitos para melhor compreensão e divisão dos componentes que envolvem a troca de dados em sistema Fieldbus. De acordo com este modelo, os processamentos de uma comunicação devem ser estruturados em até sete camadas ou níveis.

Existem regras nas quais uma tarefa complicada como é a comunicação possa ser dividida em pequenas e gerenciáveis tarefas, e com isso é possível à troca do conteúdo de uma camada somente em caso de necessidade, sem alterar as demais. Não é necessário para um sistema de comunicação implementar as sete camadas do modelo, ou seja, podem-se deixar camadas vazias.

A rede CAN que define apenas três camadas do modelo OSI, a Física (*Physical Layer*), de enlace (*Data Link Layer*) e aplicação (*Application Layer*).

Pode-se verificar estas três camadas dentre as sete do modelo OSI na Figura 3.9.

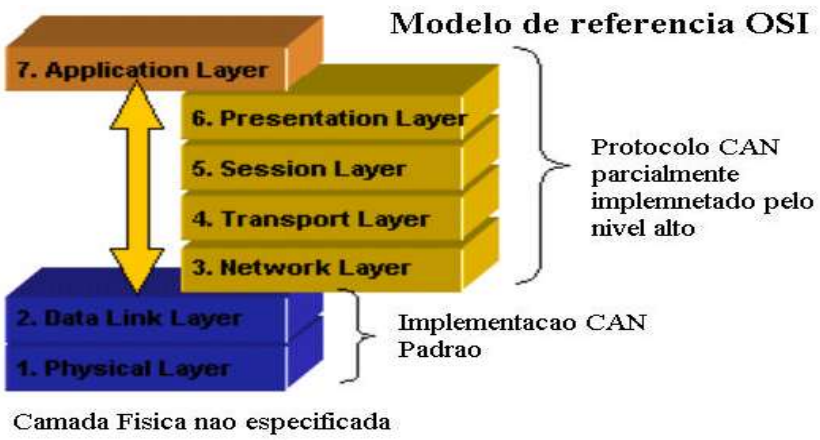


Figura 3.9 -Camadas OSI do CAN[13]

3.6 TRANSMISSÃO DE MENSAGENS CAN

Basicamente pode-se verificar que a camada física se encarrega por adequar o dispositivo aos diferentes meios de transmissão de dados. A camada de enlace passa para a camada física, além dos dados, também uma informação para a segurança dos mesmos, no momento adequado para acesso ao meio físico de acordo com a técnica de controle de acesso especificada para este sistema. A camada de aplicação, ao contrário das outras, disponibiliza serviços para o usuário. Deste modo a forma como os dados são transmitidos ou recebidos torna-se transparente para o usuário. Pode-se verificar a interação entre estas camadas pela Figura 3.10 :

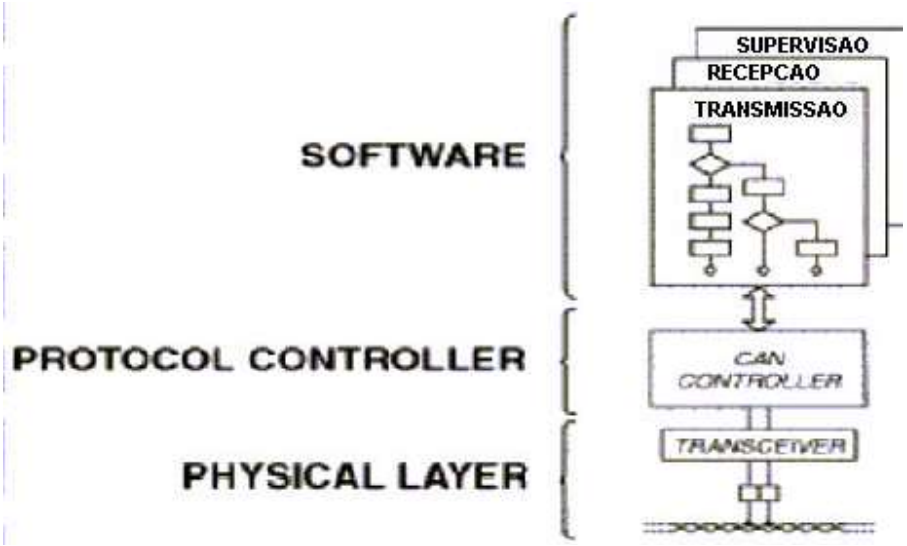


Figura 3.10-As 3 camadas OSI para o CAN[13]

3.7 ELEMENTOS QUE INTEGRAM O CAN-BUS DE DADOS

Um nó do CAN-bus é composto por um controlador CAN, um transceptor (*transceiver*), dois elementos final do barramento, um conector e dois cabos para a transmissão de dados.

Pode-se atribuir a cada nó funções tais como:

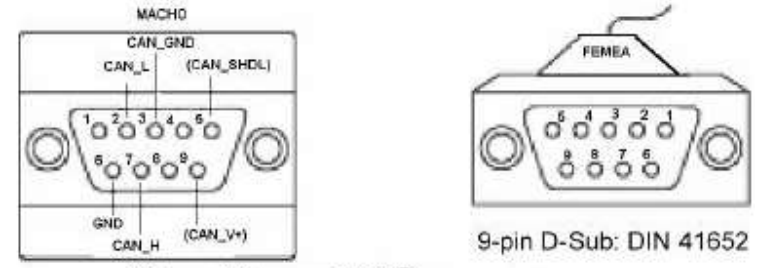
- O controlador CAN: Recebe do microprocessador, que em uma unidade de controle em um automóvel seria uma unidade de comando; os dados que acondicionados conforme o protocolo, serão transmitidos para o transceptor CAN.

Da mesma forma, recebe os dados do transceptor CAN, acondiciona-os e os passa ao microprocessador na unidade de comando.

- Transceptor CAN: é um transmissor e receptor. Transforma os dados do controlador CAN em sinais elétricos e transmite estas informações através dos cabos do CAN-Bus. Da mesma forma recebe os dados e os prepara para o controlador CAN.
- Elemento final do Bus: é uma resistência elétrica. Evita que os dados transmitidos sejam devolvidos em forma de eco dos cabos, criando informações falsas.
- Os cabos do Bus de Dados: Funcionam de forma bidirecional e servem para transmitir dados. Ao trabalhar com CAN-Bus não se define o nó destinatário dos dados. Os dados são transmitidos e todas as unidades os recebem e os analisam. A Figura 3.12 dá uma idéia de dois nós interligados pelo CAN-Bus.
- **Conectores CAN**

  Não existe qualquer norma que classifique os conectores CAN, assim fica a cargo das camadas mais altas do protocolo escolher o mais apropriado de acordo com as suas especificações. Os conectores mais usados são relacionados abaixo.

  - DSUB de 9 pólos, proposto pela CiA conforme Figura 3.11.
  - Mini-C de 5-pólos, usado pelo DeviceNet e pelo SDS
  - Conector alemão de 6-pólos, proposto pelo CANHUG.



PINO	SINAL	DESCRIÇÃO
1	-	Reserved
2	CAN_L	CAN_L bus line dominant low
3	CAN_GND	CAN Ground
4	-	Reserved
5	(CAN_SHLD)	Optional CAN Shield
6	GND	Optional Ground
7	CAN_H	CAN_H bus line dominant high
8	-	Reserved
9	(CAN_V+)	Opcional CAN - alimentaçao externa

Figura 3.11-Pinagem do conector DB-9 (CAN-BUS)[13]

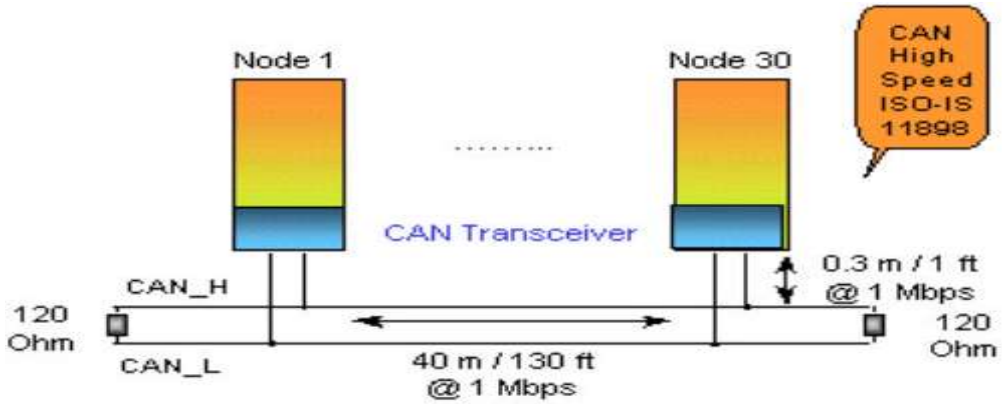


Figura 3.12-Elementos que interligam o Bus[13]

Camada Física

A camada física é responsável pela transferência de bits entre os diferentes nós da rede, e define os níveis elétricos, o esquema de sincronização a impedância do cabo e a codificação de acordo com o meio físico adotado conforme apresentados na Figura 3.13.

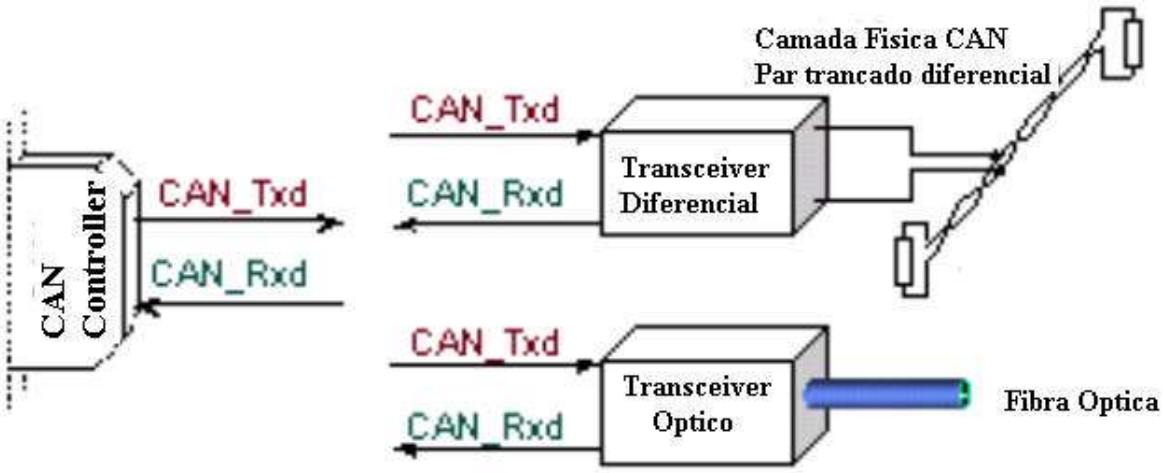


Figura 3.13-Comunicação CAN – Transceiver – CAN Controller[13]

Para transmitir estes dados é necessário que módulos controladores “conversem” entre si, isto é, entendam a informação que será trocada entre eles. Para esta compreensão entre módulos especifica-se um protocolo, no qual regras são definidas. A velocidade de transmissão é um de seus itens, importante para receber e transmitir uma resposta em *real-time*. Porém nem todos módulos durante a troca de informação necessitam de respostas com a mesma velocidade para todas as aplicações. Por isso o CAN apresenta diferentes taxas de transmissão de bits especificada pelo padrão ISO.

A velocidade máxima possível para o bus CAN dependerá do comprimento do barramento. Como se pode ver no gráfico da Figura 3.14, por exemplo a velocidade máxima de 1Mbit/segundo, limita a um cabo com o comprimento máximo de 40 metros. Isto acontece porque o esquema de arbitragem necessita que a onda se propague até ao nó mais remoto e volte antes de ser amostrada. Ele funciona de um modo *quasi-stationary*, isto é, na transmissão de cada bit é dado tempo suficiente ao sinal para se estabilizar antes de ser amostrado, amostragem essa que é feita quase simultaneamente por todas as estações.

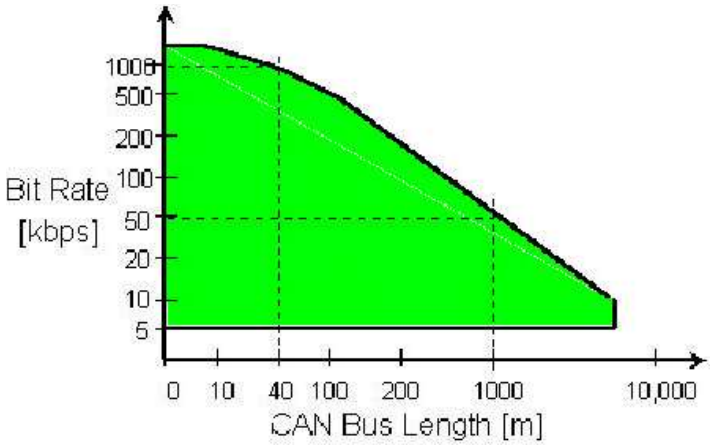


Figura 3.14-Velocidade de troca de dados em função do comprimento do Barramento[13]

O protocolo CAN é o um padrão definido internacionalmente pela ISO 11898 e ISO 16845 que garante a intercomunicação entre os circuitos integrados CAN.

De acordo com a norma ISO 11898-1, o protocolo CAN não define rigidamente uma camada física. Apenas define dois níveis lógicos que são representados por esta camada e que também irão possibilitar a transmissão e recepção dos sinais: um nível chamado de dominante (0) e o outro recessivo (1).

O ISO 11898 (com taxa de até 1 Mb/s) define o CAN como *high speed* e ISO 11519-1 (com taxa de até 125 Kb/s) como *low speed* [17].Esta diferença de velocidade de transmissão pode ser compreendida através da necessidade em duas situações diferentes:

1-Quando um motorista de um automóvel aciona o botão que abre o vidro do carro, a informação não é urgente, portanto poderá ser transmitida mais lentamente.

2- Entretanto, quando os freios do automóvel são solicitados, eles deverão ser acionados o mais rápidos possível para se evitar um acidente.

O barramento CAN contém duas linhas de transmissão por onde trafegam os dados. Uma das linhas é o CAN *high (CAN_H)* e a outra é o CAN *low (CAN_L)*. No modelo de camadas OSI, a definição do meio de transmissão faz parte da camada física. A representação dos níveis lógicos nas linhas CAN é feita de acordo com o nível de tensão que será colocado na linha. Um bit recessivo (nível lógico 1) é representado através das duas linhas do barramento com um nível de tensão de cerca de 2.5 V, de modo que a diferença de potencial entre CAN_H e CAN_L será de 0V. Um bit dominante (nível lógico 0) é representado colocando-se CAN_H em 3.5 V e CAN_L em 1.5 V. Isto resulta uma diferença de potencial para um bit dominante de cerca de 2V, veja o detalhe na Figura 3.15.

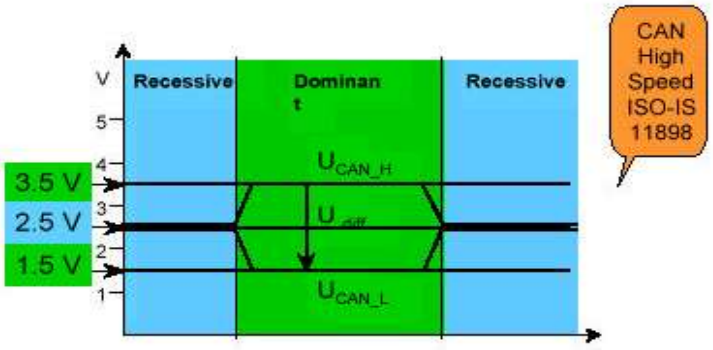


Figura 3.15-Níveis elétricos no barramento CAN[13]

O CAN realiza uma comunicação *broadcast* onde toda mensagem enviada ao barramento será identificada por apenas aquele periférico que tiver o mesmo ID presente na mensagem. Este ID servirá para priorizar uma informação. Esta prioridade será feita através de comparações bit a bit sendo que o nível baixo tem maior influencia sobre o nível alto. Por exemplo, um ID 0010 e um ID 0001, o segundo têm prioridade para exercer o seu serviço. A utilização de identificadores ao invés de endereços na rede facilita a configuração do sistema tornando-se mais flexível, podendo suportar ainda, a capacidade de receptores múltiplos e multi-mestre. A comunicação de dados é feita através do método CSMA/BA (*Carrier Sense Multiple Access with Bitwise Arbitration*), com *Non-Destructive Bitwise Arbitration.*

O barramento CAN utiliza a codificação NRZ (Non Return to Zero) com *bit-stuffing* (para assegurar o sincronismo), para comunicação de dados em um barramento diferencial a dois fios (geralmente par trançado).

No método NRZ o fluxo dos bits em uma mensagem CAN é codificado. Isto significa que durante o tempo total de bit o nível de bit é dominante ou recessivo, ou seja, não há em cada bit uma subida e descida do *edge* como acontece no modo Manchester, veja detalhe na Figura 3.16.

Figura 3.16-Método de codificação NRZ[13]

A camada física CAN pode ser dividida em três sub-camadas: PLS (*Physical Signaling*), PMA (*Physical Medium Attachment*) e MDI (*Medium Dependent Interface*).

A MDI trata dos níveis de cabeamento e conectores da rede. A PMA é representada pelos *tranceivers*.

A subcamada PLS compõe chips controladores CAN que codificam/decodificam os bits, definindo o tempo em que o bit permanece em nível alto ou baixo e se responsabiliza pela sincronização dos dados, detalhe na Figura 3.17.

Figura 3.17-As três subcamadas físicas[13]

Tempo de Bit (*Bit Timing*)

Como definido na norma ISO 11898, o tempo nominal de cada bit de uma mensagem é composto por 4 segmentos de tempo não sobrepostos, conforme a Figura 3.18:

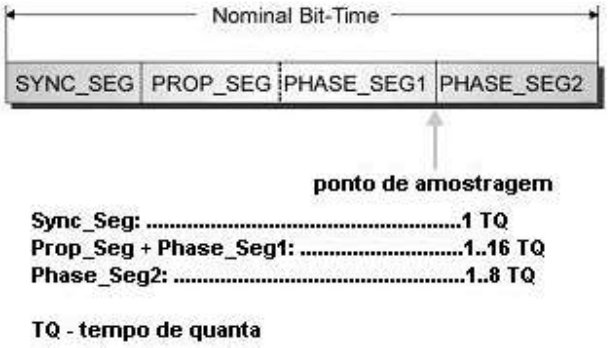


Figura 3.18-Os 4 segmentos de tempo[13]

Sync-seg é o segmento usado para fazer a sincronização dos nós do *bus*, é esperada uma transição de recessivo para dominante durante este segmento. O *Prop-seg* é um período de tempo usado para compensar o atraso resultante do meio físico da rede. O *Phase-seg1* é um segmento *buffer* que pode ser alongado durante a re-sincronização de modo a compensar o impulso do oscilador e a diferença de fase positiva entre os osciladores do transmissor e do receptor. O *Phase-seg2* é outro segmento de *buffer* que pode ser encurtado durante a re-sincronização de modo a compensar os erros de fase negativa e o impulso do oscilador.

O ponto de *sample* (amostragem) é sempre no final do segmento *phase-seg1*, e é neste momento que o bus é lido e interpretado o valor do bit corrente.

Quer seja a transmitir ou a receber, todos os nós têm o mesmo tempo de bit nominal, o tempo de cada bit é programado em cada nó da rede e é calculado em função do oscilador local de cada nó (valor que é introduzido no registro BRP do controlador de cada nó) e do número de time quanta por bit.

O registro BRP é um dos requisitos da norma ISO 11898, e tem que ser programada pelo utilizador com um valor inteiro entre 1 e 32.

Ao programarmos registros BRP (*baud rate prescaler*) individuais e o time quanta por bit com os valores corretos, podemos ter nós com osciladores a operar com freqüências diferentes, a funcionarem todos com o mesmo tempo de bit nominal.

Cada um dos 4 segmentos de tempo de um bit tem comprimento de um, ou mais, time quanta, pois a especificação da Bosch CAN2 refere:

- *Sync-seg* é sempre programado com um time quanta.
- *Prop-seg* é programado com um valor entre 1 e 8.

- *Phase-seg1* é programado com um valor entre 1 e 8.
- *Phase-seg2* é programado com o valor máximo entre *Phase-seg1* e o tempo de processamento da informação.

Sincronização

Quando qualquer nó recebe um frame de dados, é necessário que o receptor se sincronize com o emissor.

Como não existe nenhum impulso de *clock* explícito que um o CAN possa usar como referência, são utilizados dois mecanismos para manter a sincronização:

Sincronização Dura - Ocorre dentro de cada controlador quando este está em modo de recepção, e é detectada uma transição, de recessivo para dominante, falhada no campo de SOF.

Re-sincronização

De modo a compensar o impulso do oscilador e as diferenças de fase entre o transmissor e o receptor, a re-sincronização automaticamente aumenta ou diminui o tamanho do tempo de bit, dependendo de onde ocorre à transição recessivo - dominante. O tamanho máximo que o tempo de bit pode ser alargado ou encurtado e é definido pelo tempo do quanta, também conhecido como largura do salto de sincronização, SJW conforme Figura 3.19.

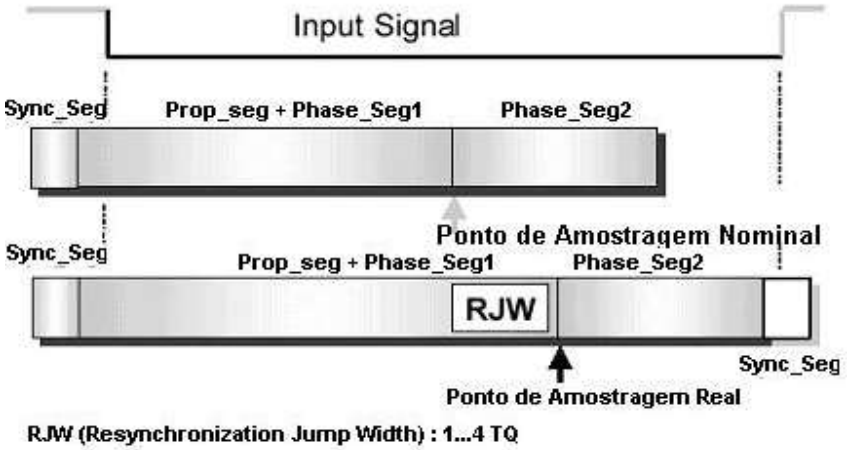


Figura 3.19-Re-sincronização[13]

4 CAPÍTULO 4 – CONTROLADOR CAN

4.1 ESTUDO DO MCP2510

O circuito integrado MCP2510 da Microchip é próprio para interface entre um PC ou microcontrolador e um barramento CAN. Pode operar nos modos ativo e passivo e é capaz de transmitir e receber mensagens de identificadores padrão e estendido. Possui filtros para seus dois registros de recepção capazes de selecionar apenas as mensagens desejadas, evitando superdimensionamento do microcontrolador. Possui três registros de transmissão acionada por hardware, através de pinos próprios, ou por software, através de programação SPI (*Serial Peripheral Interface*) , operando a velocidades de até 5Mb/s.

Qualquer mensagem detectada no barramento CAN é checada para erros e então comparada com a programação dos filtros para decidir se a mensagem deve ser movida para um dos dois *buffers* de recepção ou não.

Pinagem

O MCP2510 é fabricado nas versões de encapsulamento PDIP, SOIC e TSSOP, possuindo de 18 a 20 pinos. A Figura 4.1 mostra um encapsulamento PDIP, com 18 pinos. A alimentação é feita pelos pinos V_{DD}(18), entre 3 e 7 $volts_{dc}$, e V_{SS}(9), pino de referência. Possui um pino para sincronizar um pedido de reset, RESET(17), sob operação normal em nível alto. O CI precisa de um sinal de sincronismo de clock através do pino OSC1(8), de 1 a 25MHz, e faz uma ligação interna com o pino OSC2(7) para disponibilizar o mesmo clock a outro dispositivo. A freqüência do sinal de clock do pino CLKOUT(3) é programável, com base num divisor de 1, 2, 4 ou 8 da freqüência de oscilação em OSC1(8). Os pinos TXCAN(1) e RXCAN(2) são conectados ao barramento CAN, por intermédio de um *transceiver*. Os pinos TXnRTS(4, 5 e 6) têm duas funções possíveis programáveis: entradas digitais ou requisições de transmissão imediatas via hardware. Os pinos RXnBF(10 e 11) também possuem duas funções possíveis programáveis: sejam como saídas digitais, ou como indicadores de recepção de mensagens nos respectivos registros de recepção. Há ainda um pino INT(12), de interrupção mascarável programável para indicar de imediato qualquer mudança de

estado que se deseje. Finalmente, os pinos SCK(13), SI(14), SO(15) e CS(16) fazem parte do módulo de comunicação SPI e serão analisados na implementação.

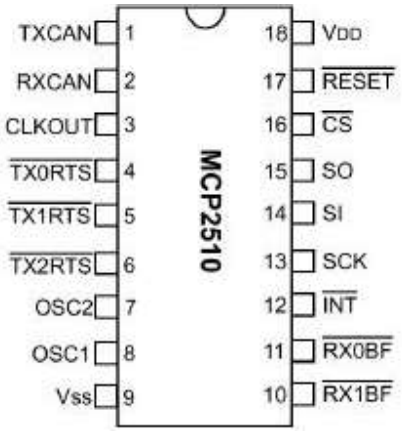


Figura 4.1-Encapsulamento do MCP2510[7]

Para que o MCP2510 seja ligado ao barramento CAN ainda é necessário que exista um *transceiver* CAN entre eles. Um exemplo disponível no mercado é o PCA82C251 da Philips Semiconductors Inc., cujo circuito interno é mostrada na Figura 4.2. Os pinos TXCAN(1) e RXCAN(2) do MCP2510 são ligados aos pinos TXD(1) e RXD(4) do PCA82C251 respectivamente. Dependendo da ligação do pino Rs(8), estabelecem-se três modos de operação neste CI: *high-speed* (conectado diretamente a terra), *low-speed* (conectado à terra por meio de um resistor) e inativo (conectado a Vcc(3)). O pino Vref(5) provê uma tensão de referência de 0.5 volts, e não é utilizado. A alimentação é feita através dos pinos Vcc(3), entre 4.5 a 5.5 volts, e pelo pino GND(2), referência à terra. Por fim, os pinos CANH(7) e CANL(6) são conectados ao barramento CAN.

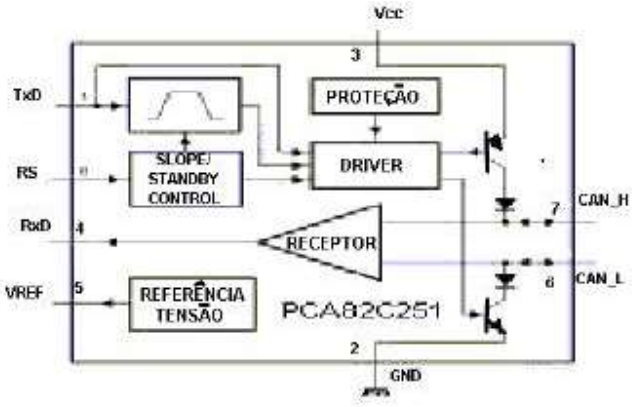


Figura 4.2-Diagrama elétrico interno do PCA82C251 (*Transceiver* CAN)[8]

Uma ligação típica deste CI em um barramento CAN é mostrada pela Figura 4.3.

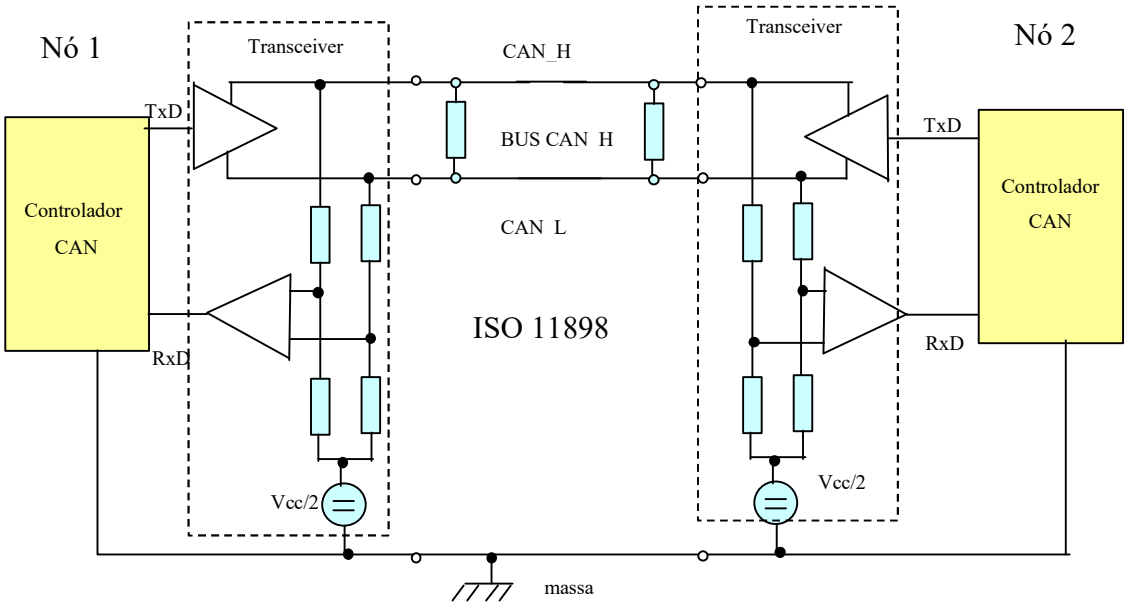


Figura 4.3-Conexão de MCP2510 com controlador CAN [8]

Transmissão de Mensagens

Como visto, a rede CAN define apenas três camadas do modelo OSI, a Física (*Physical Layer*), de Enlace (*Data Link Layer*) e Aplicação (*Application Layer*). Basicamente pode-se verificar que a camada física é especificada de acordo com os diferentes meios de transmissão de dados. Uma seqüência de bits representa os dados que estão sendo transmitidos pelo canal. A camada de enlace passa para a camada física, além dos dados, também uma informação para a segurança dos mesmos, além de assegurar também que o receptor pode aceitar o pacote de dados por completo e passá-los adiante no momento adequado.

A camada de aplicação, ao contrário das outras, disponibiliza serviços para o usuário. E Desta forma, para os usuários a forma como os dados são transmitidos ou recebidos torna-se transparente.Com o MCP2510 é possível simplificar aplicações que requeiram uma interface com o barramento CAN. O protocolo CAN habilita todas as funções para receber e transmitir mensagens no barramento. As mensagens são transmitidas primeiramente carregando o buffer apropriado de mensagem e registros de controle. A transmissão é iniciada usando os bits de registros de controle, através da interface SPI, ou

usando os seus pinos de transmissão. Os estados (status) e os erros podem ser checados lendo os registros apropriados.

Qualquer mensagem detectada no barramento CAN é checada para erros e então emparelhada contra o uso de filtros definidos para ver se deveria ser movido para um dos dois *buffers* de recepção.

O MCU-CAN (Microcontroller Unit) é conectado ao dispositivo pela interface SPI. A escrita e leitura de todos os registros são feitas usando o padrão SPI através de comandos de leitura e escrita. Os pinos de interrupção são utilizados para permitir grande flexibilidade no sistema. Existem pinos de interrupção de múltiplos propósitos, como também, pinos de interrupção específicos para cada um dos registros recepção, que podem ser usados para indicar quando uma mensagem é válida em um dos *buffers* de recepção. O uso dos pinos de interrupção específica é opcional e o propósito geral dos pinos de interrupção, como também as posições dos registros (acessado via interface SPI), podem ser usados para determinar quando uma mensagem válida foi recebida.

Existem também três pinos avaliados para inicializar imediatamente uma transmissão que tenha sido carregada em um dos três registros de transmissão. O uso destes pinos é opcional e a inicialização de uma transmissão de mensagem pode ser feita utilizando os registros de controle via interface SPI.

Buffers de Transmissão

O MCP2510 possui 3 *buffers* de transmissão. Cada um deles ocupa 14 Bytes de SRAM e são mapeados na memória do dispositivo. O primeiro Byte é um registro de controle associado com o *buffer* de mensagem. A informação no registro determina a condição sobre a qual a mensagem será transmitida e indica o status na transmissão da mensagem. Cinco Bytes são usados para manter os identificadores (padronizados e estendidos) e outras informações da mensagem. Os últimos oito Bytes são para os oito possíveis bytes de dados da mensagem a ser transmitida.

Para o MCU-CAN ter acesso a escrita no *buffer* de transmissão o bit do registro de controle do buffer n de transmissão TXBnCTRL.TXREQ (TXREQ: bit 3 do registro TXBnCTRL) deve estar zerado (*clear)* [7], indicando que o buffer de mensagem está pronto para ser transmitido. Com isto, o registro do buffer de transmissão n do

identificador alto padrão TXBnSIDH (registro que indica os 8 bits mais significativos do identificador a ser transmitido pelo buffer n), o registro do buffer de transmissão n do identificador baixo padrão TXBnSIDL (registro que indica os 3 bits menos significativos do identificador a ser transmitido pelo buffer n) e registro de comprimento de dados do buffer n de transmissão TXBnDLC (registro que armazena a quantidade de dados a ser transmitido pelo buffer n de transmissão) devem ser carregados. Se os bytes de dados estão presentes na mensagem, os registros TXBnDm (registros onde serão armazenados cada byte m do buffer n para transmissão) devem ser carregados.

Antes de enviar uma mensagem, o MCU-CAN deve inicializar o CANINTE.TXNIE para habilitar ou desabilitar uma interrupção quando uma mensagem é enviada (TXNIE: bits 2,3 e 4 do registro CANINTE, quando 1 lógico habilita a interrupção, torna o buffer N pronto para receber um dado para transmissão, quando 0 a interrupção estará desabilitado). O MCU-CAN deve também inicializar o bit de prioridade TXBnCTRL.TXP (TXP<1:0>: bit 0 e 1 que estabelecem 4 níveis de prioridade para o buffer de transmissão [7]).

Prioridade de transmissão

Esta é uma prioridade dentro do MCP2510 para se realizar uma transmissão de mensagem pendente, ou seja, está relacionada com a prioridade de transmissão de mensagem no CAN.

Antes de enviar o SOF, a prioridade de todos os *buffers* que estão empilhados para uma transmissão são comparadas. O *buffer* de transmissão de maior prioridade enviará seu dado primeiro. Por exemplo, se o buffer de transmissão 0 tem uma maior prioridade do que buffer de transmissão 1, o buffer 0 irá enviar sua mensagem primeiro. Se dois *buffers* apresentam a mesma prioridade de envio, o buffer com o número maior, enviará sua mensagem primeiro. Por exemplo, se um buffer de transmissão 1 tem a mesma prioridade de envio que o buffer 0, a mensagem do buffer 1 irá ser enviada primeiro.

Existem quatro níveis de prioridade de transmissão. Se TXBnCTRL.TXP<1:0> para um buffer de mensagem particular for ajustado para 11, aquele buffer tem a maior prioridade possível. Se TXBnCTRL<1:0> para um buffer de mensagem particular for 00, aquele buffer tem a mais baixa prioridade possível.

Inicializando a Transmissão

Para inicializar uma transmissão de mensagem, o bit TXBnCTRL.TXREQ deve ser ligado para cada buffer a ser transmitido. Isto pode ser feito pelo "*writing"* através do registro via interface SPI ou ligando o pino TXnRTS para um buffer de transmissão particular a ser transmitido. Se a transmissão é iniciada via interface SPI o bit TXREQ pode ser ligado ao mesmo tempo em que os bits de prioridade TXP.

Quando o bit TXBnCTRL.TXREQ é ligado, os bits TXBnCTRL.ABTF (Message Aborted Flag : bit de leitura apenas, que indica resultado da transmissão, 0 indica sucesso e 1 indica abordagem na transmissão), TXBnCTRL.MLOA (Message Lost Arbitration: bit que indica situação da arbitragem, se 0 não perdeu arbitragem se 1 perda) e TXBnCTRL.TXERR (Transmission Error Detected: bit que indica se houve erro no barramento, se for 1 um erro no barramento ocorreu na transmissão e se for 0 nenhum erro de barramento ocorreu) serão zerados (*clear).* A transmissão irá começar quando o dispositivo detectar que o barramento está disponível. O dispositivo irá começar a transmissão com a prioridade da mensagem que estiver pronta.

Quando a transmissão foi completada com sucesso, o bit TXBnCTRL.TXREQ estará zerado (*clear)*, o bit CANINTF.TXnIF estará ligado (TxnIF bit2 2,3 e 4 que sinalizam através dos pinos 4, 5 e 6 do MCU_CAN sucesso na transmissão), e um interruptor será produzido se o bit CANINTE.TXnIE for ligado.

Se uma transmissão de mensagem falha, o TXBnCTRL.TXREQ irá permanecer ligado indicando que a mensagem ainda está pendente para a transmissão, e uma das seguintes condições de flags estará ligada. Se uma mensagem começou a ser transmitida mas encontra condição de erro, os bits TXBnCTRL.TXERR e os bits CANINTF.MERRF(Message Error Interrupt Flag: bit 7 do registro CANINTF) serão ligados e uma interrupção será gerada no pino INT se o bit CANINTE.MERRE(bit 7 do registro CANINTE) estiver ligado. Se a mensagem perder para uma mensagem de prioridade maior, o bit TXBnCTRL.MLOA será ligado.

Pinos TXnRTS

Os pinos TXnRTS são pinos de entrada que podem ser configurados como entradas de requisição para envio, os quais provê de um meio secundário de inicializar uma

transmissão de mensagem: ou de algum dos buffers de transmissão ou como um padrão digital de entradas.

A configuração e controle destes pinos são realizados usando o registro de estado e controle TXRTSCTRL . O registro TXRTSCTRL pode ser ainda modificado quando o MCP2510 estiver no modo de configuração. Se configurado para operar como pedido de envio de dados, o pino é mapeado no respectivo bit TXBnCTRL.TXREQ para buffer de transmissão. O bit TXREQ é mapeado pela queda do *edge* do pino TXnRTS. Os pinos TXnRTS são designados para permitir que eles sejam ligados pelos pinos RXnBF para iniciar automaticamente uma transmissão de mensagem quando o pino RXnBF estiver baixo. Os pinos TXnRTS têm um *pullup* nos resistores de 100K Ohm (nominal).

Pode-se observar no fluxograma da Figura 4.4 o esquemático da transmissão de mensagens.

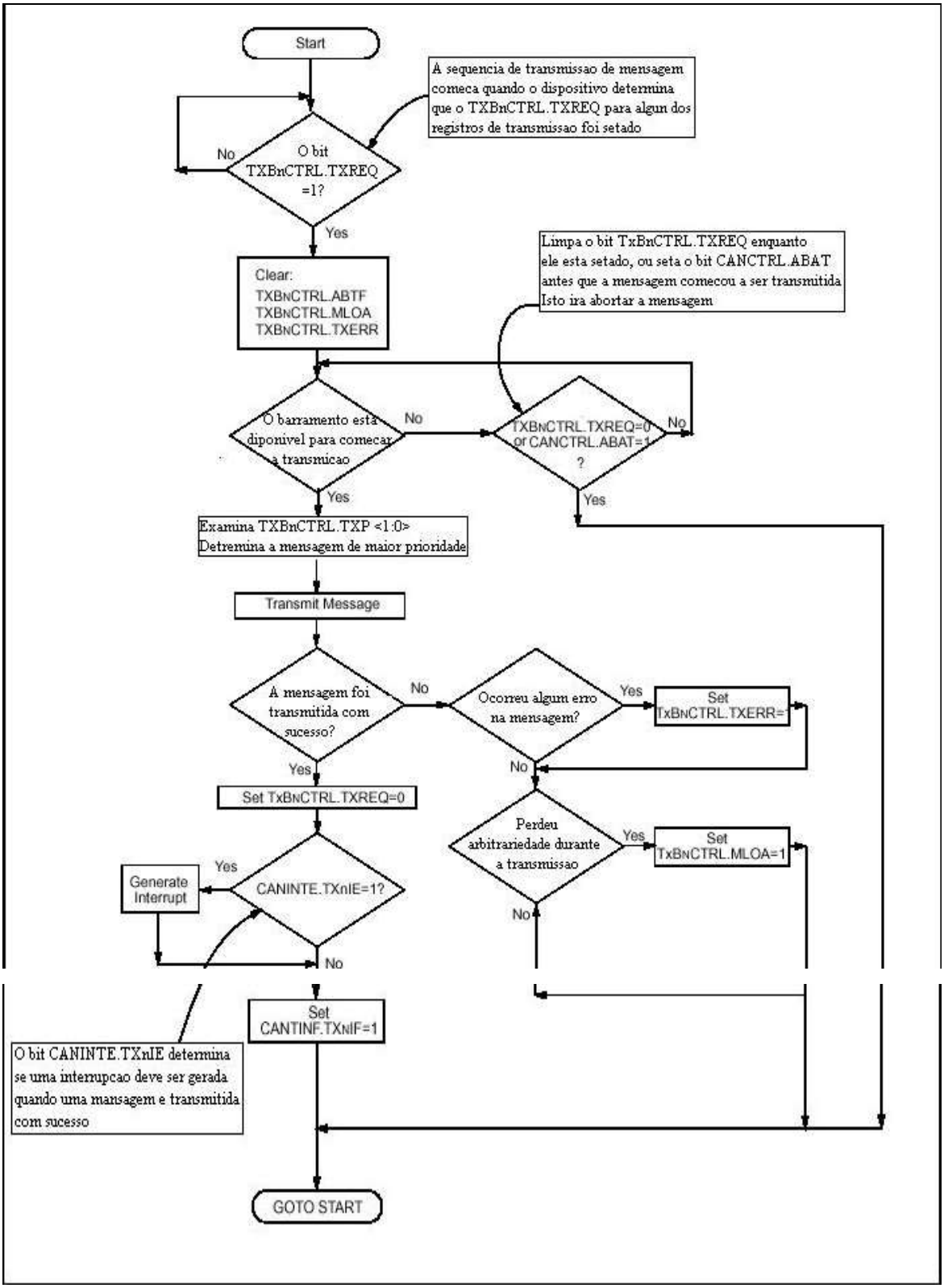


Figura 4.4-Fluxograma da transmissão do MCP2510[7]

Abortando a Transmissão

O MCU-CAN pode pedir para abortar uma transmissão de mensagem em um *buffer* de mensagem pelo bit TXBnCTRL.TXREQ. Também, todas as mensagens pendentes podem ser requisitadas para serem abortadas ligando o bit CANCTRL.ABAT(Abort All Pending Transmission: bit 4 do registro de controle CANCTRL). Se o bit CANCTRL.ABAT é ligado para abortar todas as mensagens pendentes, o usuário deve limpar (*reset*) este bit (tipicamente depois que o usuário verificar que todos os bits TXREQ foram zerados) para continuar a transmissão da mensagem. O *flag* CANCTRL.ABT será ligado se a abordagem foi requerida via bit CANCTRL.ABAT.

Apenas as mensagens que ainda não começaram a ser transmitida podem ser abortadas. Uma vez que uma mensagem começou a transmissão, não será possível ao usuário re-ligar (*reset*) o bit TXBnCTRL.TXREQ. Depois que uma transmissão de mensagem perde arbitrariedade, a mensagem irá ser retransmitida independentemente do pedido de abordagem.

4.2 REGISTROS INTERNOS

O MCP2510 é um dispositivo incapaz de tomar decisões por si só, dependendo de um microcontrolador que possa gerir as informações que vem e vão pelo barramento CAN e são armazenadas no MCP2510 através de seus registros de memória. O MCP2510 possui registros de controle, de status e de dados, totalizando um mapeamento de 128 bytes de memória.

As funções dos registros de controle são: definir o modo de operação, a freqüência de trabalho do *clock*, habilitar interrupções, configurar informações de sincronismo, estado dos pinos de saída, e como as mensagens de transmissão e recepção deverão ser manipuladas pelo MCP2510.

Os registros de *status* sinalizam códigos de interrupção, modo de operação, erros internos e do barramento CAN, recebimentos de requisições remotas, qual filtro aceitou a mensagem recebida e em que *buffer* ela foi armazenada, tanto para verificação por recenseamento, como por atendimento à chamada de interrupção do microcontrolador.

Fazem parte dos registros de dados os dois *buffers* de recepção e seus respectivos filtros e máscaras, e os três *buffers* de transmissão.

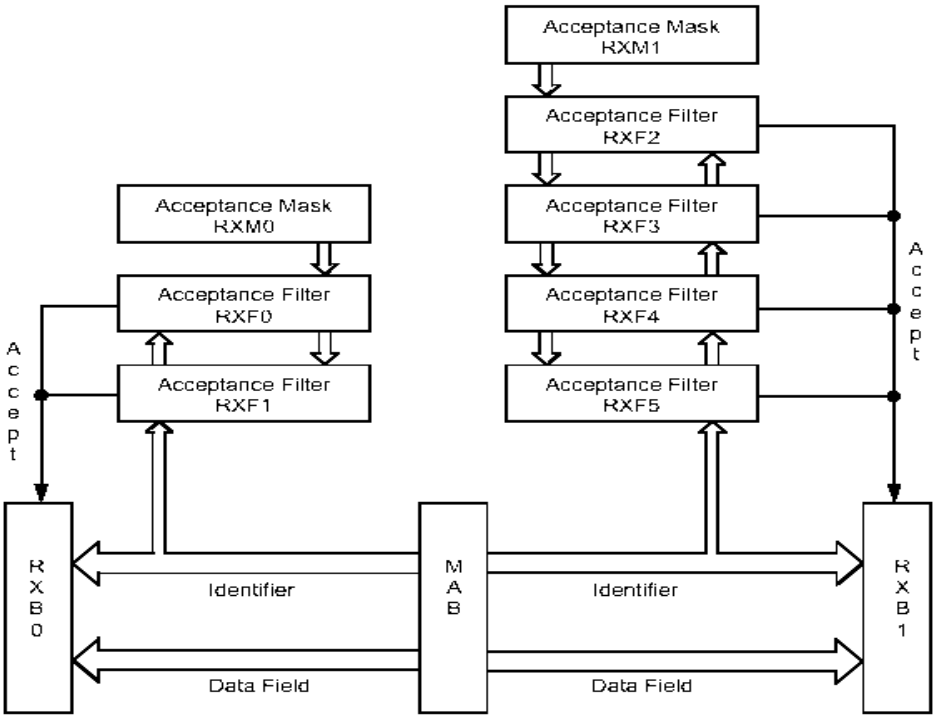


Figura 4.5-Recepção de Mensagens[7]

Uma mensagem recebida pelo MCP2510 só é armazenada em um dos *buffers* de recepção se seu identificador for aceito por um de seus respectivos filtros e máscaras, como mostra a Figura 4.5.

4.3 MODOS DE OPERAÇÃO

O MCP2510 tem cinco modos de operação: *Configuration, Normal, Sleep, Listen-Only, Loopback*.

O modo *Configuration* é automaticamente selecionado após ligar ou reiniciar o equipamento, e é apenas neste modo que é permitida a modificação do conteúdo de máscaras, filtros e de alguns registros de controle.

No modo *Sleep*, o MCP2510 pára de oscilar internamente, minimizando o consumo de corrente. A interface SPI, porém, continua funcionando, permitindo a leitura dos registros internos. Pode-se habilitar nesse caso uma função *wake-up*, de forma a retirar o MCP2510 do modo *Sleep*, caso seja detectada uma perturbação pela comunicação de um dado no barramento CAN, restabelecendo o modo Normal.

O modo *Listen-Only* provê um meio para que o MCP2510 receba todas as mensagens do barramento, incluindo as mensagens de erro. Este modo pode ser usado para monitorar as aplicações do barramento ou para detectar o *baud rate* quando houver "*hot plugging*", ou seja, trafico de dados pelo barramento. Os contadores de erro são reiniciados e desativados nesse estado.

O modo *Loopback* permite que o MCP2510 fique isolado do barramento CAN, operando num modo "silencioso". Neste caso, mensagens transmitidas são imediatamente recebidas pelo próprio MCP2510. É um modo, portanto, para o auto teste do dispositivo.

No modo Normal, o MCP2510 pode monitorar todas as mensagens no barramento, gera os bits de reconhecimento, janelas de erro e é o único modo capaz de transmitir mensagens para todo o barramento CAN.

4.4 PROTOCOLO DE COMUNICAÇÃO SPI

É através da interface SPI a única forma do microcontrolador acessar os registros internos do MCP2510, seja para leitura, escrita, recenseamento ou reposta a uma chamada de interrupção. O MCP2510 utiliza o protocolo de comunicação SPI mestre/escravo, *full-duplex*, síncrono, enviando dados via pino SO(15) e recebendo pelo pino SI(14). O pino CS(16) deve sempre estar em nível baixo para possibilitar o uso desta interface, e no pino SCK(13) deve ser gerado um clock de sincronismo pelo microcontrolador, que é o mestre da comunicação.

Na Figura 4.6 é possível verificar uma esquematização simplificada do funcionamento do MCP2510 e também de como é a interconexão com o barramento CAN e sua interface de comunicação serial (SPI).

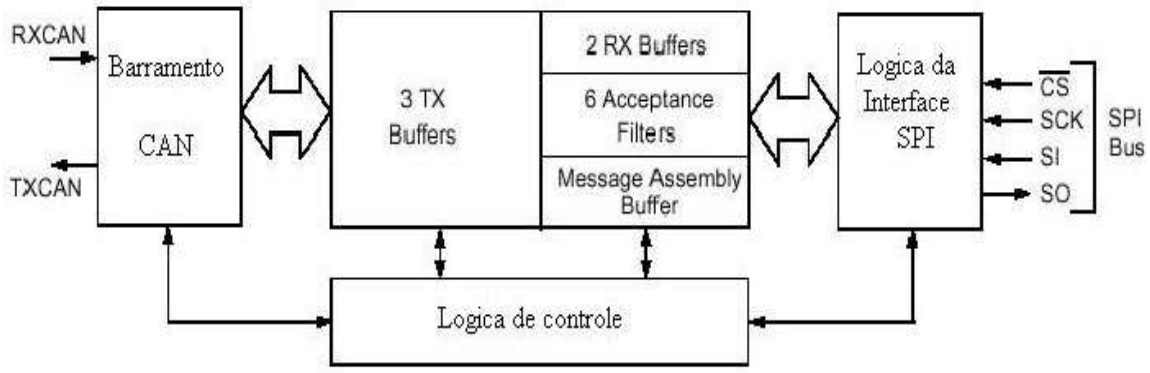


Figura 4.6-Diagrama simplificado do funcionamento do MCP2510[7]

Existem seis instruções pré-definidas no MCP2510 para acessar seus registros: *Read*, para ler um byte de qualquer posição da memória; *Write*, para escrever um byte em qualquer posição da memória; *Request-to-Send*, que é uma requisição de transmissão por software; *Read-Status*, que informa o *status* de alguns bits de sinalização de transmissão e recepção; *Bit-Modify*, que permite alterar bits individualmente de posições específicas da memória e *Reset*, que efetua um reinício do dispositivo por software.

Com o intuito de realizar a programação do sistema através do conjunto de códigos de operação, foi possível desenvolver uma tabela de funções do MCP2510 com a possibilidade de listar todos registros afetados, retratando operações fundamentais do núcleo do MCP2510 como modo de operação, manipulação de códigos de interrupção, transmissão e recepção de dados.

4.5 TEMPORIZAÇÃO DE BIT

Análise do Funcionamento

Para analisar a necessidade da temporização de bit, foi necessário relembrar detalhes do funcionamento do barramento CAN ressaltadas a seguir. O barramento CAN possui duas linhas de sinal elétrico cujo valor de tensão diferencial resulta num bit dominante “0” ou recessivo “1” num dado instante de tempo, visto por todos os nós conectados ao barramento. O conjunto organizado de vários bits recessivos e dominantes dá origem a uma mensagem entendida por um ou mais nós. Estas duas linhas de sinal elétrico, têm comportamento diferencial e são na verdade apenas uma linha de sinal e outra de referência, que não está necessariamente ligada fisicamente a uma referência. Dessa forma, pode-se ver claramente que, com apenas uma linha de sinal, não pode haver sincronia na transmissão e recepção dos bits. O barramento CAN é, portanto, um barramento de comunicação assíncrono. Isso força todos os nós a terem uma mesma taxa nominal de bits, ou baud rate, de forma que os nós receptores recuperem o mesmo clock do nó transmissor. Como não há uma rampa de clock no barramento para avisar quando um bit está disponível para ser lido, essa sincronização é feita durante uma transmissão normal de dados no barramento, de forma que os nós receptores devem possuir a capacidade de compor a mesma mensagem que foi transmitida lendo seus bits nos tempos corretos.

Deve-se perceber também que outro agravante é o fato de que os nós possuem *clocks* próprios, próprios de seus respectivos osciladores. Com isso, cada nó precisa calcular uma fração de sua freqüência de oscilação para se ajustar ao baud rate nominal do barramento.

Para isso, o MCP2510 usa uma lógica interna chamada Laço de Travamento de Fase Digital (DPLL). Ele se incumbe de se sincronizar ao barramento e fornecer um temporizador para leitura configurado com o baud rate nominal. Sua função é quebrar o tempo de um bit em vários pequenos tempos eqüiduráveis chamados de Quanta de Tempo (T_Q). No MCP2510, para cada tempo de um bit, existem quatro segmentos de tempo não sobreposto, cada um com o tamanho múltiplo de T_Q, veja detalhe na Figura 4.7:

- Segmento de Sincronização
- Segmento de Tempo de Propagação
- Segmento de Buffer de Fase 1
- Segmento de Buffer de Fase 2

O tempo nominal de um bit no MCP2510 pode ser programado e varia entre 8 T_Q e 25 T_Q, dispondo uma taxa de bit nominal de até 1Mb/s no barramento CAN.

O Quanta de Tempo é uma unidade fixa que depende apenas do baud rate e da freqüência do oscilador. Sendo o oscilador um dispositivo de hardware, a única variável disponível ao usuário para se alterar o Quanta de Tempo é o baud rate, que pode variar entre 1 e 64. Sua equação é dada por:

$$T_Q = 2 \cdot (\text{Baud Rate} + 1) \cdot T_{OSC}$$

onde Baud Rate é representado pelos bits CNF1.BRP<5:0>.

Por exemplo:

Se F_{OSC} == 16 MHz, CNF1.BRP<5:0> == 00h e Tempo de Bit Nominal == 8 T_Q, então T_Q = 125 ns e Taxa de Bit Nominal = 1 Mb/s.

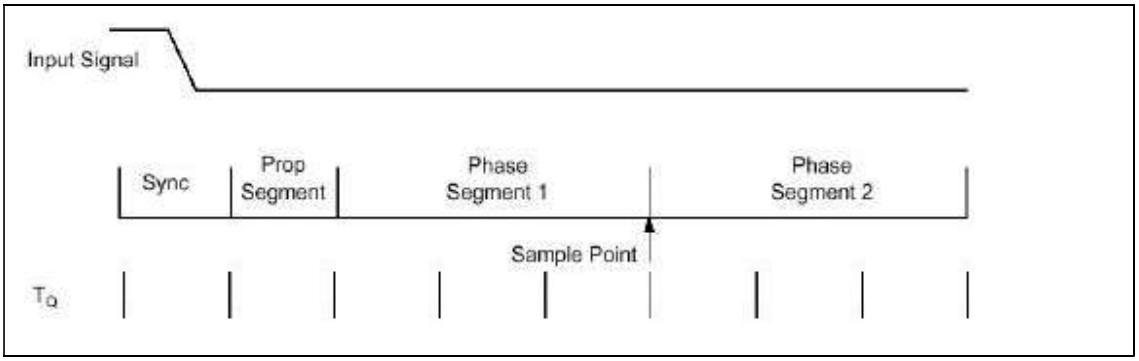


Figura 4.7-Segmentação do Tempo de um Bit[7]

Segmento de Sincronização (seg_sinc)

Esta parte do tempo de bit serve para sincronizar todos os nós do barramento. A rampa do sinal de entrada é esperada para acontecer durante este segmento, cuja duração é de 1 T_Q.

Segmento de Propagação (seg_prop)

Serve para compensar o tempo de atraso físico da rede elétrica, que consiste na soma do tempo de atraso no barramento e do tempo de propagação interna em cada um dos nós. É calculado como a soma dos tempos arredondando para cima da viagem do sinal do transmissor ao receptor (duas vezes o tempo em se tratando do barramento CAN), da demora do comparador de entrada e da demora do driver de saída. Seu tamanho pode ser programado através dos bits CNF2.PRSEG<2:0> variando de 1 T_Q a 8 T_Q.

Os tempos individuais de atraso são:

- Duas vezes o tempo de atraso físico no barramento CAN de ponta a ponta (T_{BUS})
- Duas vezes o tempo de atraso do comparador de entrada (T_{COMP})
- Duas vezes o tempo de atraso do driver de saída (T_{DRIVE})
- Uma vez o tempo de propagação da entrada até a saída do controlador CAN (T_{CAN}), definido como no máximo 1 T_Q + atraso em ns
- $T_{PROPAGAÇÃO} = 2 \cdot \left(T_{BUS} + T_{COMP} + T_{DRIVE} \right) + T_{CAN}$
- $SEG\_PROP = T_{PROPAGAÇÃO} / T_Q$

Segmentos de Buffer de Fase (SEG_FASE)

Segmentos de buffer de fase são utilizados para localizarem, de forma ótima o ponto de amostragem do bit recebido dentro dos critérios do tempo do bit nominal. O ponto de amostragem ocorre entre os dois segmentos de buffer de fase. Cada um dos segmentos poderá ser esticado ou encurtado em tempo pelo processo de re-sincronização, garantindo a funcionalidade do DPLL. O segmento de buffer de fase 1 pode ser programado para um tamanho de 1 T_Q a 8 T_Q. O segmento de buffer de fase 2 provê um tempo de atraso antes da próxima transição do dado transmitido, também podendo ser programado entre 1 T_Q a 8 T_Q, embora devido aos requerimentos do Tempo de Processamento da Informação (IPT) seu tamanho mínimo deva ser de 2 T_Q.

Ponto de Amostragem

É o instante de tempo no qual a informação contida no barramento será lida, ocorrendo no final do segmento de buffer de fase 1. Se a temporização de bit é muito lenta contendo muitos T_Q, pode-se especificar múltiplas leituras do barramento no ponto de amostragem. Existem duas opções de amostragem: Simples, com uma única leitura no ponto de amostragem; a melhor de três, com três leituras consecutivas do barramento onde o resultado do bit é aquele que mais está presente entre as três leituras.

Tempo de Processamento da Informação

É o segmento de tempo que se inicia no ponto de amostragem, reservado para o cálculo do nível do bit subseqüente. As especificações CAN sugerem que sua duração seja menor ou igual a 2 T_Q. No MCP2510 sua duração é de 2 T_Q, justificando a duração do segmento de buffer de fase 2 ser de no mínimo 2 T_Q.

Sincronização

Para compensar o deslocamento de fase entre cada um dos nós do barramento, o controlador CAN precisa se sincronizar a rampa de sinal recebido. A sincronização é o processo no qual a função DPLL é implementada. Quando uma rampa no dado transmitido é detectada, a lógica compara a posição da rampa com o tempo esperado (SEG_SINC). O circuito ajusta os tamanhos necessários do segmento de buffer de fase 1 e 2 , mecanismos estes usados na sincronização.

A sincronização forçada acontece apenas quando uma rampa gerada por uma transição de recessivo para dominante ocorre durante a condição de barramento inativo, indicando o início da transmissão. Após a sincronização forçada, os contadores de tempo de bit são reiniciados junto com o segmento de sincronização. A sincronização forçada força à rampa ocorrida a se alinhar com o segmento de sincronização do tempo de bit reiniciado. Segundo as regras de sincronização, se uma sincronização forçada ocorre, não haverá re-sincronização durante esse tempo de bit.

Numa re-sincronização, o segmento de buffer de fase 1 deve ser alongado (Figura 4.8) ou o segmento de buffer de fase 2 deve ser encurtado (Figura 4.9). A quantidade de alongamento ou encurtamento não é superior ao valor dado pela Largura de Salto de Sincronização (SJW). O valor de SJW será acrescentado ao segmento de buffer de fase 1 ou decrementado do segmento de buffer de fase 2. O SJW representa o laço de filtragem do DPLL. O SJW pode ser programado com valores entre 1 T_Q a 4 T_Q.

As informações de temporização só podem ser fornecidas na passagem de estado recessivo para dominante. A propriedade de que apenas um número fixo de sucessivos bits, tem a mesma polaridade (*bit stuffing*), garante o processo de re-sincronização durante a transmissão de um pacote de mensagens.

O erro de fase de uma rampa é dado pela posição da rampa em relação ao segmento de sincronização, medido em T_Q, e é definido como:

- E = 0 se a rampa se encontrar exatamente sobre o segmento de sincronização
- E > 0 se a rampa se encontrar antes do ponto de amostragem
- E < 0 se a rampa se encontrar depois do ponto de amostragem do bit anterior

Se a magnitude do erro for menor ou igual ao valor programado em SJW, o efeito da re-sincronização é o mesmo que o da sincronização forçada.

Se a magnitude do erro for maior que o valor em SJW, e o erro de fase for positivo, o segmento de buffer de fase 1 é alongado para um valor igual ao contido em SJW.

Se a magnitude do erro for maior que o valor em SJW, e o erro for negativo, o segmento de buffer de fase 2 é encurtado para o valor igual ao contido em SJW.

As seguintes regras de sincronização devem ser seguidas:

- Somente um tipo de sincronização é permitido durante um tempo de bit
- Uma rampa será usada para sincronização somente se o valor detectado no ponto de amostragem anterior for diferente do valor imediato do barramento após a rampa
- Todas as outras rampas recessivas para dominantes obedecendo às condições anteriores serão usadas na re-sincronização, com a exceção do nó transmissor transmitindo um bit dominante que não executará a re-sincronização sendo uma rampa recessiva para dominante com erro de fase positivo.

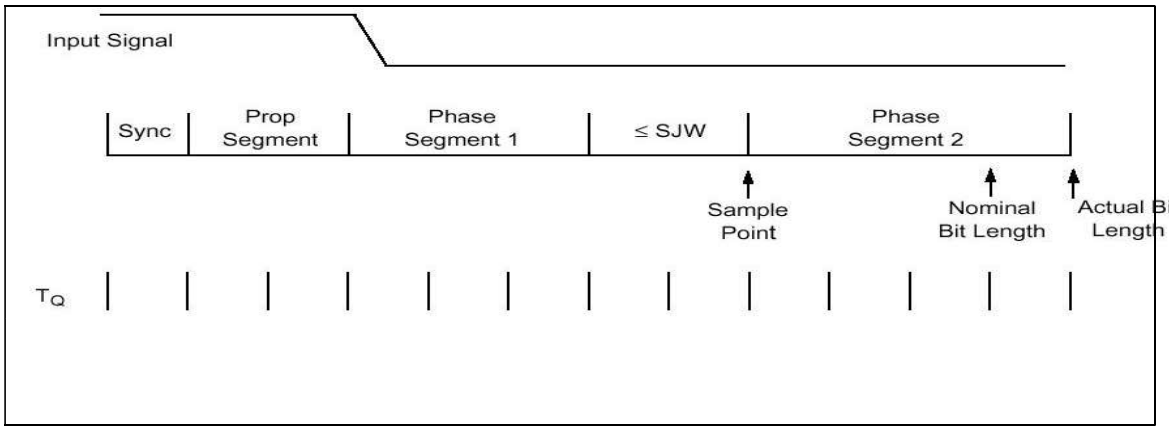


Figura 4.8 - Alongando o segmento de buffer de fase 1[7]

Programando os Tempos dos Segmentos

Há alguns requisitos para a programação dos tempos dos segmentos:

- SEG_PROP + SEG_FASE1 >= SEG_FASE2
- SEG_PROP + SEG_FASE1 >= T_{ATRASO}
- SEG_FASE2 > SJW

Por exemplo, assumindo que se deseja um barramento CAN de 125 kHz de baud rate com F_{OSC} == 20 MHz:

T_{OSC} = 50 ns, escolhe-se BRP = 04h fazendo T_Q = 500 ns. Para obter 125 kHz, o tempo de bit deve ser 16 T_Q.

Tipicamente, a amostragem do bit deve ser localizada entre 60% e 70% do tempo de bit, dependendo dos parâmetros do sistema, e, também, o T_{ATRASO} está comumente entre 1 T_Q e 2 T_Q.

Dessa forma, os segmentos ficam: SEG_SINC = 1 T_Q; SEG_PROP = 2 T_Q; se fizermos SEG_FASE1 = 7 T_Q, então o ponto de amostragem ficará em 10 T_Q após a transição e deixará SEG_FASE2 = 6 T_Q.

Uma vez que SEG_FASE == 6 T_Q, pelas regras, SJW deve ser no máximo 4 T_Q. Normalmente, um valor grande de SJW só é útil quando a geração de clock nos vários nós é incerta e instável, como se fossem utilizados capacitores cerâmicos. Dessa forma, um SJW de 1 T_Q é suficiente.

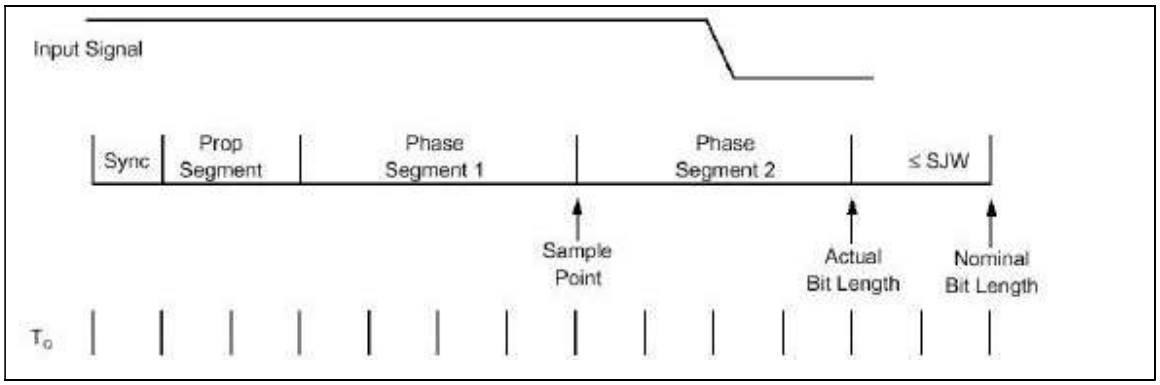


Figura 4.9-Encurtando o segmento de buffer de fase 2 [7]

Tolerância do Oscilador

Os requisitos para a temporização de bit permitem a utilização de capacitores cerâmicos em aplicações, com taxas de transmissão de até 125 kb/s. Já para uma velocidade máxima de barramento com protocolo CAN, um oscilador de quartzo é necessário. Uma variação máxima de nó para nó de 1.7% é permitida no oscilador.

4.6 TRATAMENTO DA SPI

A comunicação SPI possibilita que 8 bits de dados sejam transmitidos e recebidos simultaneamente. Por ela que o microcomputador através da porta paralela, comunica-se com o MCP2510, que por sua vez transmite as informações, e recebe ordens dele para executar alguma tarefa. O PC fica responsável por fornecer um *clock* de sincronismo ao MCP2510, de até 5 MHz.

Através da comunicação SPI, o PC configura o MCP2510 para a transmissão dos dados no barramento CAN, bem como modifica as máscaras e filtros de recepção para aceitarem apenas mensagens específicas que circulem pelo barramento CAN. Além disto, faz o recenseamento dos registros de *status* do MCP2510, controla a habilitação de suas

interrupções para que possam avisar-lhe do armazenamento de novas mensagens e então atendê-las. E faz tudo isso de maneira semi-automática, bastando apenas programar a estrutura das instruções SPI que devem ser comunicadas, bem como tratar os dados que completam essa estrutura tanto no envio como na recepção pelo PC. No nó PC tem-se uma estrutura de software desenvolvida para a comunicação SPI, que gerencia o envio de instruções ao MCP2510 e armazena dados provenientes deste, avisando ao programa da mudança de estado dos registros por meio de interrupções de software.

5 CAPÍTULO 5 - MATERIAL DE APOIO

5.1 HARDWARE DEDICADO

Para apoio ao desenvolvimento do trabalho foram utilizados recursos em hardware e software destinados à análise de funcionamento e desenvolvimento de programas para geração de interfaces gráficas. Entre eles estão recursos proprietários como o kit de desenvolvimento da Microchip, a linguagem de programação Visual Basic e o hardware padrão a partir do computador pessoal PC, utilizando a porta paralela para interfaceamento com hardware desenvolvido.

5.2 HARD DE DESENVOLVIMENTO

O *kit* de desenvolvimento MCP2510 da Microchip visto na Figura 5.1 possui uma estrutura de hardware que possibilita o estudo do componente MCP2510, desde sua organização interna, seu comportamento em meio a um barramento CAN simulado e até mesmo a capacidade de interação com um barramento CAN real. A placa está dividida em duas partes, onde cada parte corresponde a um nó CAN independente, mas ligados entre si por um barramento CAN .

Figura 5.1-Kit de desenvolvimento CAN[9]

O chamado nó 0, ou nó do PC, possui um MCP2510 que tem seus pinos SPI ligados a uma porta paralela típica do PC. Junto com o *kit*, vem um software para PC que simula os comandos de um microcontrolador sobre o MCP2510 do nó 0. Com isso é possível analisar os registros internos do MCP2510, bem como configurá-los para interagirem com o barramento, enviando mensagens pré-definidas em intervalos pré-determinados de tempo, ou até mesmo para filtrarem apenas mensagens pertinentes.

Do lado do chamado nó 1, ou nó do PIC, como o próprio nome diz, existem além do MCP2510, soquetes para PICs, de forma a implementarem aplicações personalizadas. Os soquetes para PIC possuem ligações com uma porta serial de PC típica, possibilitando a comunicação USART entre um PIC e um PC.

Existe ainda uma abertura para que a placa seja conectada a um barramento CAN real externo, disponibilizando as linhas CANH e CANL das saídas dos *transceivers* dos nós anteriores.

A placa vem de fábrica com um PIC16F876 no soquete do nó 1, com software já gravado, compatível com o PIC16F73. Através de um software para PC que o acompanha, todos os recursos que a placa dispõe podem ser aproveitados para o aprendizado. Neste projeto, o objetivo foi construir um nó PC capaz de se comunicar com o nó automotivo através da rede CAN. Utilizou-se o *kit* de desenvolvimento MCP2510 para análise dos sinais, tempos e outras informações para aprimoramento da interface a ser desenvolvida. Obtendo sucesso nessa empreitada, o passo final seria de elaborar um software próprio, em ambiente gráfico no PC para monitoramento veicular.

Pode-se então verificar que este kit de demonstração e implementação de softwares para comunicação com dispositivos de interface CAN, é composto por:

Nó CAN de número 0 para diagnóstico de barramento CAN e nó Can de número 1 reservado para implementação de novo dispositivo, porta paralela para comunicação do nó 0 com PC, porta serial RS232-C Standard para comunicação do nó 1 com o dispositivo a ser implementado, cabo Paralelo DB25 Standard Centronics para conexão entre as portas paralelas do PC (LPT) e do kit didático, possibilidade de conexão do kit com um barramento CAN externo, software próprio de diagnóstico da linha CAN através do nó 0, fonte própria de 9V com cabo e adaptador.

O chamado nó 0, ou nó do PC, possui um MCP2510 que tem seus pinos SPI ligados a uma porta paralela típica do PC. Junto com o *kit*, vem um software para PC que simula os comandos de um microcontrolador sobre o MCP2510 do nó 0. Com isso é possível analisar os registros internos do MCP2510, bem como configurá-los para interagir com o barramento, enviando mensagens pré-definidas em intervalos pré-determinados de tempo, ou até mesmo filtrar apenas mensagens pertinentes.

Do lado do chamado nó 1, ou nó do PIC, como o próprio nome diz, existem além do MCP2510, soquetes para PIC, de forma a implementar uma aplicação personalizada. Os soquetes para PIC possuem ligações com uma porta serial de PC típica, capaz de possibilitar a comunicação USART entre um PIC e um PC.

Existe ainda uma abertura para que a placa seja conectada a um barramento CAN real externo, disponibilizando as linhas CANH e CANL das saídas dos *transceivers* dos nós anteriores.

Neste projeto, o objetivo foi construir um nó PC capaz de comunicar-se com o nó automotivo através da rede CAN. Utilizou-se o *kit* de desenvolvimento MCP2510 para análise dos sinais, tempos e outras informações para aprimoramento entre interface a ser desenvolvida. Obtendo sucesso nessa empreitada, o passo final seria de elaborar um software próprio, em ambiente gráfico no PC para monitoramento veicular.

Pode-se então verificar que neste kit de demonstração e implementação de softwares para comunicação com dispositivos de interface CAN, inclui Hardware com 2 nós CAN:

Nó 0 para diagnóstico de barramento CAN e nó 1 reservado para implementação de novo dispositivo, Porta Paralela para comunicação do nó 0 com PC. Porta Serial RS232-C Standard para comunicação do nó 1 com o dispositivo a ser implementado, cabo Paralelo DB25 Standard Centronics para conexão entre as portas paralelas do PC (LPT) e do kit didático, possibilidade de conexão do kit com um barramento CAN externo, software próprio de diagnóstico da linha CAN através do nó 0, fonte própria de 9V com cabo e adaptador.

Para configurar o kit deve-se seguir uma seqüência de passos e logo após executar testes. O software *firmware* do kit, CANKing, utiliza ambos os nós CAN para o funcionamento básico da comunicação CAN. Cada nó pode ser configurado para transmitir e receber mensagens que poderão ser locadas em posições definidas pela tela.

5.3 SOFTWARE DE APOIO

Das três *telas* disponíveis fornecidos pelo fabricante do kit de desenvolvimento utilizou-se a tela "*Register Template*". Esta tela será detalhada um pouco mais a seguir.

Template Register

O "*Register Template"* é uma interface gráfica de baixo nível que permite o controle dos níveis dos bits dos registros do MCP2510. Esta *tela* pode ser usada para a tornar familiarizado com o MCP2510, sendo possível ajustar as máscaras e filtros, temporizações de bit, registros de configuração e outras funções associadas com a configuração do MCP2510.

Das várias telas disponíveis no CANKing o *Template Register* é o que é utilizado para análise como apoio à construção da interface de monitoramento veicular, pois este possibilita controle manual de todos os registros, podendo transmitir, receber e fornecer dados físicos para sincronismo entre os nós que estão se comunicando. A Figura 5.2 mostra o *Template Registro*, onde se tem várias janelas relacionadas a seguir:

OBS: Alguns procedimentos de configuração podem ser executados para garantir uma operação correta antes de se continuar, como a verificação do endereço da porta paralela e a freqüência do clock.

Physical Layer Window

Os três registros CNF usados para toda temporização de bit do CAN podem ser configurados nesta janela.

OBS: Os registros CNF apenas podem ser modificados no modo de configuração.

Configuration Windows

Os registros TXRTSCTRL, BFPCTRL, CANINTF, CANINTE e CANCTRL são todos modificados nesta janela. Estes são os registros de controle e de *flag*.

Se uma mensagem for recebida no *template Register,* os *flags* do *buffer* de recepção no CANINTF (RX0IF e RX1IF) podem ser limpos manualmente para receber mensagens adicionais.

Transmit Windows

A janela de Transmissão controla o conteúdo dos *buffers* para os registros de transmissão, incluindo o registro de controle (TXBnCTRL), o identificador de registro e os registros de dados.

Receive Windows

Esta janela contém todo o conteúdo dos *buffers* para os *buffers* de recepção, incluindo registro de controle (RXnCTRL), o identificador de registro, e os registros de dados.

Modificando a Freqüência do Oscilador por Software

A freqüência do oscilador deve ser ajustada para relacionar a oscilação da placa com a precisão da taxa de transmissão de bits. Para ajustar, selecione " Options > MCP2510..." e ajuste a propriedade do oscilador para uma freqüência do cristal em uso que no caso é de 16000 kHz.

Salvando Configuração:

Salvando as configurações como projeto garante que os novos ajustes sejam efetuados. Para salvar deve-se optar por "File > Save" e logo após deve-se nomear o projeto.
Como descrito anteriormente, o "Template Register" é uma tela de baixo nível que permite a visualização dos bits dos registros internos do MCP2510 do nó do PC, bem como permite a sua alteração. Este software pode ser usado para se familiarizar com o MCP2510, permitindo a alteração dos filtros e máscaras, temporizações de bit, configuração dos registros e outras funções associadas com a configuração do MCP2510.

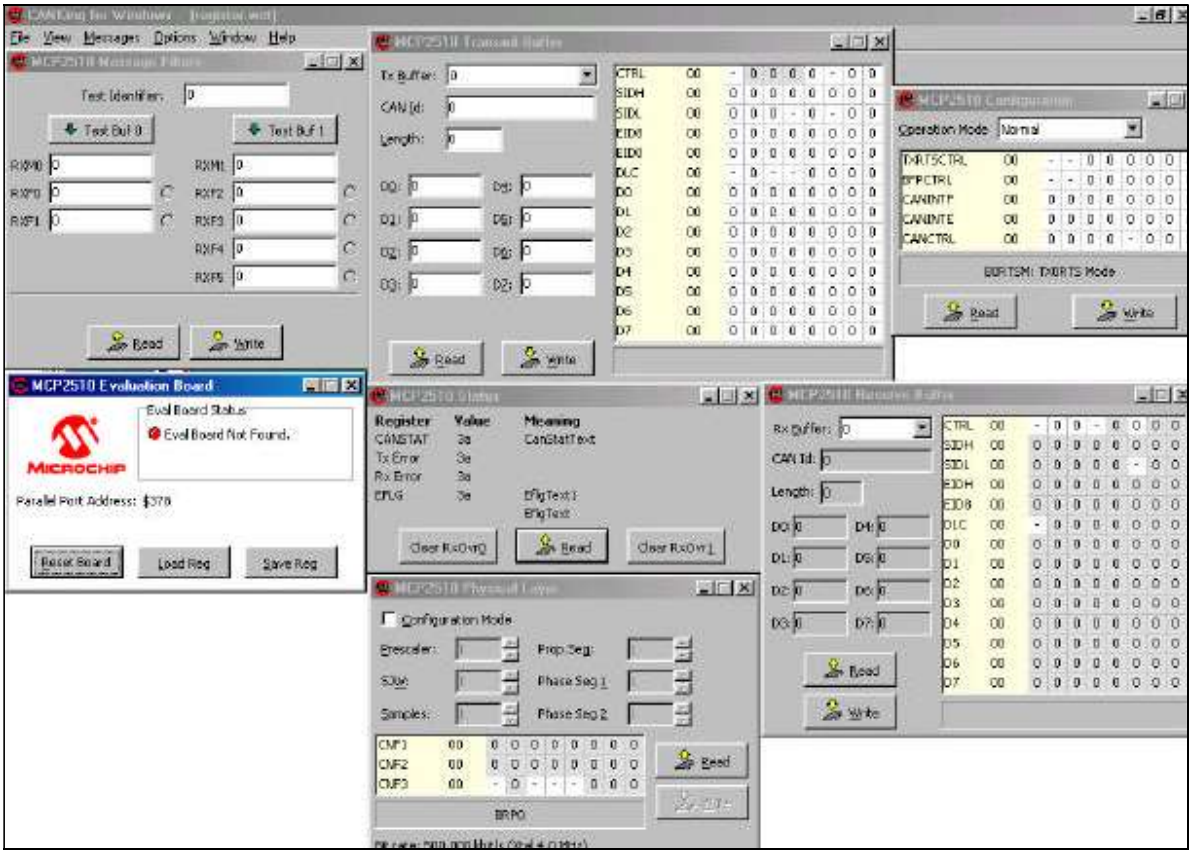


Figura 5.2-Template Register[9]

Com a utilização do kit foi possível entender como os registradores de configuração trabalham e como os demais registradores se interagem. Pode-se entender o mecanismo da transmissão e da recepção a partir do momento em que o kit foi interligado ao hardware desenvolvido.

Com base na interface gráfica do “Template Register” pode-se projetar as interfaces gráficas para o trabalho final.

6 CAPÍTULO 6 – INTERFACE CAN -PC

6.1 PORTA PARALELA

Uma vez que se utiliza a conexão entre o MCP2510 e o PC através da sua porta paralela, os próximos itens abordarão suas principais características e funcionamento, assim como a programação utilizada no projeto desenvolvido.

Interface Paralela

A porta paralela é uma interface de comunicação entre o computador e um periférico. Quando a IBM criou seu primeiro PC (Personal Computer) ou Computador Pessoal, a idéia era conectar a essa porta uma impressora, mas atualmente, são vários os periféricos que utilizam esta porta para enviar e receber dados do/para o computador (exemplos: *Scanners*, Câmeras de vídeo, Unidade de disco removível e outros).

A Porta Paralela está ligada diretamente à placa mãe e constitui um hardware padrão do PC. Cuidado deve ser tomado ao se conectarem circuitos eletrônicos a essa porta, pois, uma descarga elétrica ou um componente com a polaridade invertida, poderá causar danos irreparáveis ao computador.

6.2 MODELO DE PORTA PARALELA

Transmissão unidirecional: A porta paralela SPP (Standard Parallel Port) pode chegar à taxa de transmissão de dados a 150KBits/s. Comunica-se com a CPU utilizando um barramento de dados de 8 bits. Para a transmissão de dados entre periféricos são usados 4 bits por vez.

Transmissão bidirecional: A porta avançada EPP (Enhanced Parallel Port) chega a atingir uma taxa de transferência de 2 MB/s. Para atingir essa velocidade, será necessário um cabo especial. Comunica-se com a CPU utilizando um barramento de dados de 32 bits. Para a transmissão de dados entre periféricos são usados 8 bits por vez.

A porta avançada ECP (Enhanced Capabilities Port) tem as mesmas características que a EPP, porém, utiliza DMA (acesso direto à memória), sem a necessidade do uso do processador, para a transferência de dados. Utiliza também um buffer FIFO de 16 bytes.

Endereço da Porta Paralela

O PC nomeia as Portas Paralelas, chamando-as de LPT1, LPT2, LPT3 etc, mas, a porta física padrão do PC é a LPT1, e seus endereços são: 378h (para enviar ou receber um

byte de dados pela porta LPT1), 378+1h (para verificar o estado da porta LPT1) e 378+2h (para controlar a LPT1. Às vezes, a LPT2pode estar disponível, e seus endereços serão: 278h, 278+1h e 278+2h, com as mesmas funções dos endereços da porta LPT1 respectivamente. Veja resumo na Tabela 6.1.

Nome da Porta	Endereço de memória	Endereço da Porta		Descrição
LPT1	0000:0408	378 hexadecimal	888 decimal	Endereço base
LPT2	0000:040A	278 hexadecimal	632 decimal	Endereço base

Tabela 6.1-Portas paralelas padrão no PC[10]

6.3 REGISTRADORES

Endereço Base

Define os endereços de entrada e saída (I/O) do computador que estão relacionados a comunicação paralela do computador LPT1 e LPT2 (impressora). Estes endereços também identificam os registrador de dados. Com base neste endereço tem-se: o endereço base+1 e endereço base+2 que endereçam o Registro de Status e o Registro de Controle, conforme Tabela 6.2.

Nome	Endereços LPT1	Endereços LPT2	Descrição
Registro de Dados	378h	278h	Envia ou recebe byte do dispositivo
Registro de Status	379h	279h	Ler o Status do dispositivo
Registro de Controle	37Ah	27Ah	Envia dados de controle para o dispositivo

Tabela 6.2-Detalhes e funções diversas das LPT(s) [10]

Conector DB25

O acesso físico à porta paralela é feito pelo DB25, e é através deste, que o cabo paralelo se conecta ao computador para poder enviar e receber dados. No DB25, um pino está em nível lógico 0 quando a tensão elétrica no mesmo estiver entre 0 a 0,4V. Um pino se encontrará em nível lógico 1 quando a tensão elétrica no mesmo estiver acima de 3.1 e até 5V.

A Figura 6.1 mostra o conector padrão DB25, com 25 pinos, onde cada pino foi designado como mostra a Figura 6.2.

Figura 6.1-Foto do conector DB25 macho

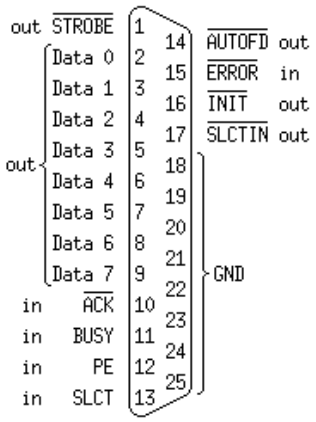


Figura 6.2-Identificação dos pinos no DB25 e sentido do sinal [10]

Para estabelecer a comunicação entre o PC e a MCU utilizam-se os três registros de operação da porta paralela. O objetivo é transportar a informação do computador para fora e vice-versa. Para tanto, utiliza-se a porta paralela do PC, a qual possibilita a transferência da informação de um modo paralelo, ou seja, o byte será transportado ao mesmo instante pelo barramento de dados do computador para o dispositivo ou vice-versa utilizando a funcionalidade dos pinos, os quais serão endereçados pelos registradores.

O motivo do uso da porta paralela é a facilidade da conexão entre esta e a interface, através de cabo sem a necessidade de acesso ao hardware do PC. A porta paralela permite a entrada de 9 bits ou a saída de 12 bits com um mínimo de hardware ligado a ela para tarefas simples. A porta é composta de 5 linhas de estado (status), 4 de controle (control) e 8 de dados, padronizadas pela IEEE 1284 a partir de 1994 que define 5 modos de operação:

a)Modo Compatível ou "Modo Centronics";

b)Modo "Nibble" (SPP);

c)Modo "Byte" (SPP);

d)EPP (Enhanced Parallel Port) ;

e)ECP (Extended Capabilities Mode).

O objetivo no início era desenvolver dispositivo compatível entre eles e com o que era utilizado, ou seja, o modo SPP (Standart Parallel Port). Os modos compatíveis Nibble e Byte usam apenas cartões com hardware de porta paralela padrão original para comunicação existente.s Após isto os modos ECP e EPP foram desenvolvidos para hardware adicional que podem trabalhar com velocidades maiores, mas com compatibilidade com o SPP.

O modo compatível ou "Modo Centronics" é o mais comum e o mais utilizado, podendo apenas enviar dados no sentido direto, ou seja, do computador para o dispositivo (ex. impressora) com uma velocidade típica de 50 KB/s. Para receber dados ou seja, no sentido reverso, os modos Nibble(4 bits) e Byte(8bits) podem ser utilizados. Estes modos utilizam a característica bidirecional (encontrada em algumas placas paralelas) para permitir que a informação do dispositivo seja acessada pelo PC. Os modos ECP e EPP utilizam um hardware específico que trabalha nos dois sentidos (bidirecional) em sincronismo físico (handshaking).

6.4 CONFIGURAÇÃO DO BARRAMENTO DA PORTA PARALELA

A Tabela 6.3 mostra detalhes sobre todos os pinos da porta paralela, identifica todos os registros, especifica como o sinal deverá ser colocado em cada pino de entrada da porta e detalha a polaridade dos sinais de controle e seus estados. Estas informações são cruciais para desenvolvimento da interface física desenvolvida no trabalho de dissertação e são mais detalhadas abaixo.

A saída da porta paralela é normalmente nível lógico TTL (0 lógico menor ou igual a 0,5V e 1 lógico maior ou igual a 4,3V) com uma fonte de corrente (*source*) e carga (*sink*) em torno de 12mA.

A nomenclatura com a utilização do "n" na frente de alguns sinais de entrada serve para indicar que estes são ativos em 0 lógico (*low*). Por exemplo: se o sinal recebido pela entrada nError estiver baixo a impressora estará com problemas. A situação de hardware invertido significa que o sinal é invertido pelo hardware da placa paralela. Por exemplo:

na linha Busy (linha de entrada - pino 11), se for aplicado +5V (1 lógico) o valor que será atribuído no bit 7 do Registro de Status será 0 lógico.

Pino (DB-25)	Sinal SPP	Direção (E/S)	Registro	Situação do hardware invertido
1	NStrobe	E/S	CONTROL	Sim
2	DATA 0	OUT	DATA	
3	DATA 1	OUT	DATA	
4	DATA 2	OUT	DATA	
5	DATA 3	OUT	DATA	
6	DATA 4	OUT	DATA	
7	DATA 5	OUT	DATA	
8	DATA 6	OUT	DATA	
9	DATA 7	OUT	DATA	
10	Nack	IN	STATUS	
11	Busy	IN	STATUS	Sim
12	Paper-Out/ Paper-End	IN	STATUS	
13	Select	IN	STATUS	
14	NAuto-Linefeed	IN/OUT	CONTROL	Sim
15	nError/nFault	IN	STATUS	
16	NInitialize	IN/OUT	CONTROL	
17	nSelect-Printer/ nSelect-In	IN/OUT	CONTROL	Sim
18 – 25	Ground	GND		

Tabela 6.3-Detalhes da relação entre registros e hardware da LPT[10]

Endereçamento das Portas Paralelas

Como já mencionado acima, a porta paralela possui três endereços normalmente usados como base de endereçamento, que estão listados na Tabela 6.4. O endereço base 0X3BC

foi originalmente introduzido para portas paralelas na placa de vídeo. O endereço 0X3BC deixou de ser utilizado quando as portas foram removidas das placas de vídeo. No entanto, estes endereços reaparecem nas "motherboards", os quais podem ser configurados pela BIOS. A porta paralela LPT1 é normalmente definida no endereço base 0X378, enquanto que a LPT2 é definida no endereço base 0X278. Portanto, os endereços de I/O do PC, 378H & 278H, são sempre definidos como endereços bases das Portas Paralelas, apesar de ser possível encontrar alguma variação de máquina para máquina.

Endereço	Observações
3BCH - 3BFH	Usado para Porta Paralela as quais estão incorporadas na placa de vídeo.Não suporta modo ECP
378H – 37FH	Endereço usual para LPT1
278H – 27FH	Endereço usual para LPT 2

Tabela 6.4-Endereçamento das portas paralelas[10]

O endereço base, usualmente é chamado de Porta de Dados ou Registro de Dados, e é simplesmente usado para saída de informação na Porta Paralela através dos pinos (pinos 2-9). Este registro é usado como porta de escrita apenas no modo padrão SPP.

O endereçamento para cada registro de dados, controle e estado é baseado no endereço base, assim temos que para a LPT1 o endereço base é 378H (LPT1) endereço para acesso do registro de estado (status) é o 379H e o endereço para acesso do registro de controle é o 37AH .

6.5 REGISTRO DE DADOS (DATA REGISTER)

O registro de dados padrão, detalhado na Tabela 6.5, é utilizado no projeto, para integrar o microcontrolador CAN com o PC. Os sinais de controle e dados enviados para o microcontrolador foram utilizados apenas de escrita. Portanto esta porta não foi utilizada para leitura, ficando esta operação reservada a outro registro da porta paralela.

Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write (OBS-1)	Bit 7	Data 7
			Bit 6	Data 6

			Bit 5	Data 5
			Bit 4	Data 4
			Bit 3	Data 3
			Bit 2	Data 2
			Bit 1	Data 1
			Bit 0	Data 0

Tabela 6.5-Registro de dados[10]

OBS: A porta paralela é bidirecional, portanto pode-se definir as operações de escrita e leitura no registro de dados. O programa fonte da Rotina 6.1, em Visual Basic, foi desenvolvida para acessar o registro de dados, necessário para funções da SPI do microcontrolador CAN

```
Sub DataPortWrite(BaseAddress%, Value%)

    ‘Escreve byte através da variável inteira ”Value” na porta paralela de endereço definido na
    ‘ variável interira “ BaseAddress”
    'rotinas desenvolvidas para apoio ao projeto do CAN Automotivo
    Out BaseAddress, Value

End Sub
```

Rotina 6.1-Linguagem fonte em Visual Base para sub rotina de escrita no registro de dados

Desenvolvida em Visual Base para escrita do dado no microcontrolador CAN através da SPI (SI).

Onde a variável BaseAddress recebe o valor do endereço base que neste caso 0X378 (LPT1).

6.6 REGISTRO DE ESTADO (STATUS REGISTER)

A porta ou registro de Status (endereço base+1) é uma porta só de leitura. Se forem feitas tentativas para escrita neste endereço elas serão ignoradas (detalhe na Tabela 6.6). A porta de Status é mascarada pelas 5 linhas de entrada (Pinos 10,11,12,13 e 15), um IRQ status register e dois bits reservados. Note que o Bit 7 (Busy) é ativo em entrada baixa (0 lógico), ou seja, invertido. Exemplo: se o bit 7 estiver em 0 lógico, significa que há +5V

no pino 11. No projeto desenvolvido esta questão foi resolvida com uma instrução lógica colocada na Rotina 6.2 em Visual Basic. Utilizou-se uma operação lógica OU EXCLUSIVA para ajustar o bit 7 do registro de estado.

Offset	Name	Read/Write	Bit No.	Propriedade
Base + 1	Porta de Status	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

Tabela 6.6-Porta de Status[10]

```
Function StatusPortRead%(BaseAddress%)
   'rotinas criadas para apoio ao projeto do CAN Automotivo

      'Lê o registro de estado da porta a paralela
   'Endereço base (BaseAddress) por exemplo 378H=LPT1
   'O hardware do PC fornece o bit 7 invertido para registrador de estado,
   'assim o valor lido deve ser invertido para iguala-se ao valor do pino do conector.
   StatusPortRead = (Inp(BaseAddress + 1) Xor &H80)
   ‘Endereço do porta de estado (BaseAdrres +1) para o exemplo 379H (LPT1)
   ‘Operador OU Exclusivo entre o byte lido e o valor 80H garante a inversão do bit 7 da porta de estado.

End Function
```

Rotina 6.2-Fonte da rotina de leitura do registro de estado da porta paralela

Rotina desenvolvida em Visual Basic para leitura do dado vindo da saída do microcontrolador CAN através da SPI (SO).

6.7 REGISTRO DE CONTROLE (CONTROL REGISTER)

A porta de controle (endereço base +2) é uma porta de escrita apenas. Utiliza-se esta porta para o sincronismo através do Strobe, Auto Linefeed, Initialize e Select Printer. Todos são invertidos exceto, Initialize, conforme detalhado na Tabela 6.7.

Offset	Name	Read/Write	Bit No.	Propriedade
Base + 2	Porta de Controle	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable Bi-Directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

Tabela 6.7-Porta de Controle[10]

7 CAPÍTULO 7 – IMPLEMENTAÇÃO

7.1 HARDWARE E SOFTWARE

O objetivo do trabalho foi construir um sistema destinado ao monitoramento automotivo através da rede digital interna CAN. Para tanto, o trabalho iniciou-se com a pesquisa, desenvolvimento e implementação da interface eletrônica e de softwares para comunicação com base no protocolo CAN.

A partir dos recursos desenvolvidos foram realizados testes de simulação entre PC(s) e monitoramentos em sistemas automotivos de fabricantes diferentes que utilizam tecnologia de rede embarcada CAN.

Foram realizados estudos, análises e testes sobre as possíveis características que possibilitariam a integração entre o microcontrolador CAN (MCU-CAN) e o microcomputador pessoal (PC) através da porta paralela via interface SPI (Serial Peripheral Interface) do MCU-CAN. Também foram analisadas as possíveis tarefas para cada módulo, seus parâmetros e formas de alteração, as informações requeridas pelos diversos módulos para executar uma atividade, as informações disponibilizadas por eles e o que cada módulo poderia executar interiormente.

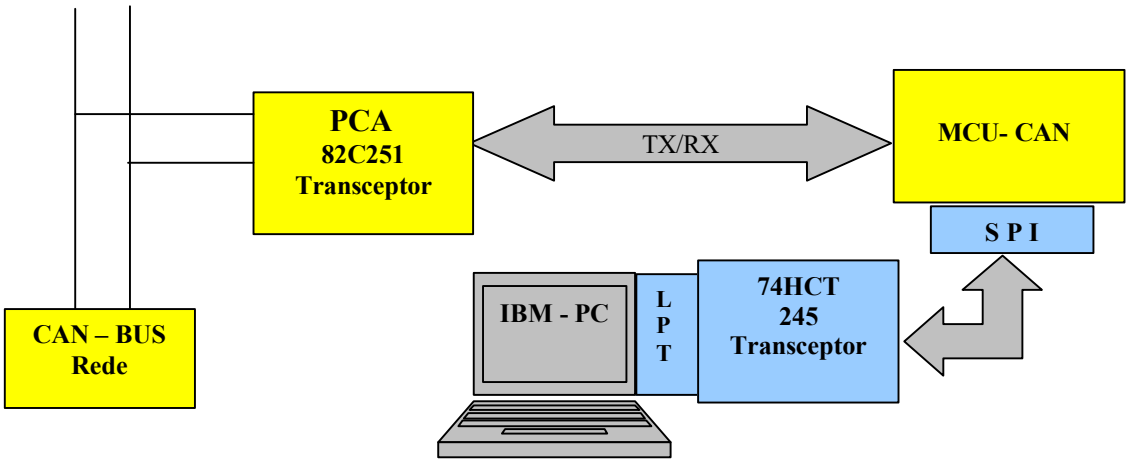


Figura 7.1-Esquemático da interconexão entre os módulos

Além disto, também foram estudadas as formas de interconectar estes módulos (PC-MCU) através da interface SPI, com as quais foi possível desenvolver uma interface

visual amigável com o usuário, via PC. A Figura 7.1 mostra a primeira proposta que foi sugerida na interconexão necessária dos módulos à rede embarcada (CAN-BUS).

Foram montados dois protótipos (circuitos eletrônicos), com freqüências de clocks diferentes. Um deles com clock de 16MHz e o outro de 20MHz para os testes preliminares e ajustes de temporização da taxa de transmissão e recepção (*baud rate*), fundamentais para o início dos testes e monitoramentos dos sistemas automotivos. A Figura 7.2 mostra o aspecto do hardware montado em placa universal com conectores para interface paralela e para barramento CAN.

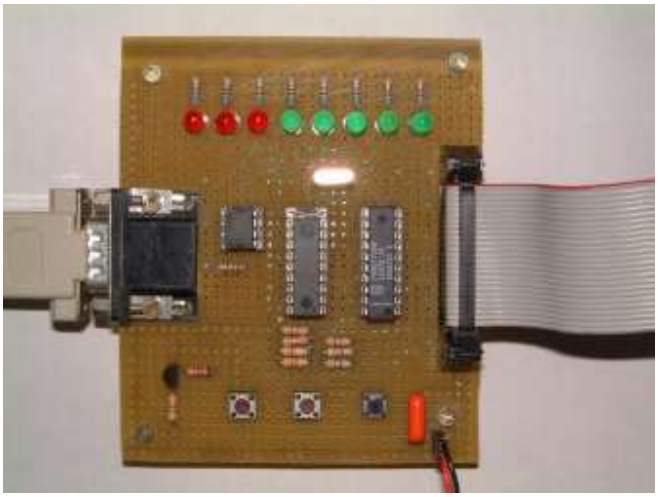


Figura 7.2-Interface (hardware)

7.2 INTERFACE GRÁFICA DE PROGRAMAÇÃO DO MCU-CAN

Para agilizar o processo da comunicação do MCU-CAN com o PC, foi desenvolvida a interface gráfica que é base do trabalho em termos de comunicação entre o microcomputador e o microcontrolador CAN, em linguagem de programação visual VB (Visual Basic).

A interface gráfica mostrada na Figura 7.3 possibilita o acesso a todos os registradores internos e elementos da unidade microcontrolador CAN. Através desta interface é possível realizar operações de escrita, de leitura de diversas informações diversas do MCU-CAN, de leitura de estado; fazer a verificação do estado lógico de bits do registro de estado (status) após a ocorrência de um evento relacionado ao MCU-CAN; modificar bits de registradores específicos, através de uma máscara; e habilitar buffer de

transmissão e ou recepção. Esta interface foi desenvolvida para pesquisa, pois possibilita o estudo da comunicação e do processamento da informação. Através da interface de teste foi possível consolidar uma ótima relação entre os nós, possibilitando ainda o monitoramento dos diversos sinais utilizados para intertravamento do PC com o MCU-CAN.

Nesta interface gráfica foi acrescentados um menu com opção de abertura de outras interfaces gráficas, tais como de "Recepção" e "Transmissão" (detalhadas posteriormente). Ainda através desta interface pode-se selecionar a porta paralela desejada e outros detalhes na programação do MCU-CAN. Portanto, trata-se de uma interface poderosa de alto nível a qual possibilita uma programação dinâmica do controlador CAN, explorando todo recurso de hardware interno do MCU-CAN.

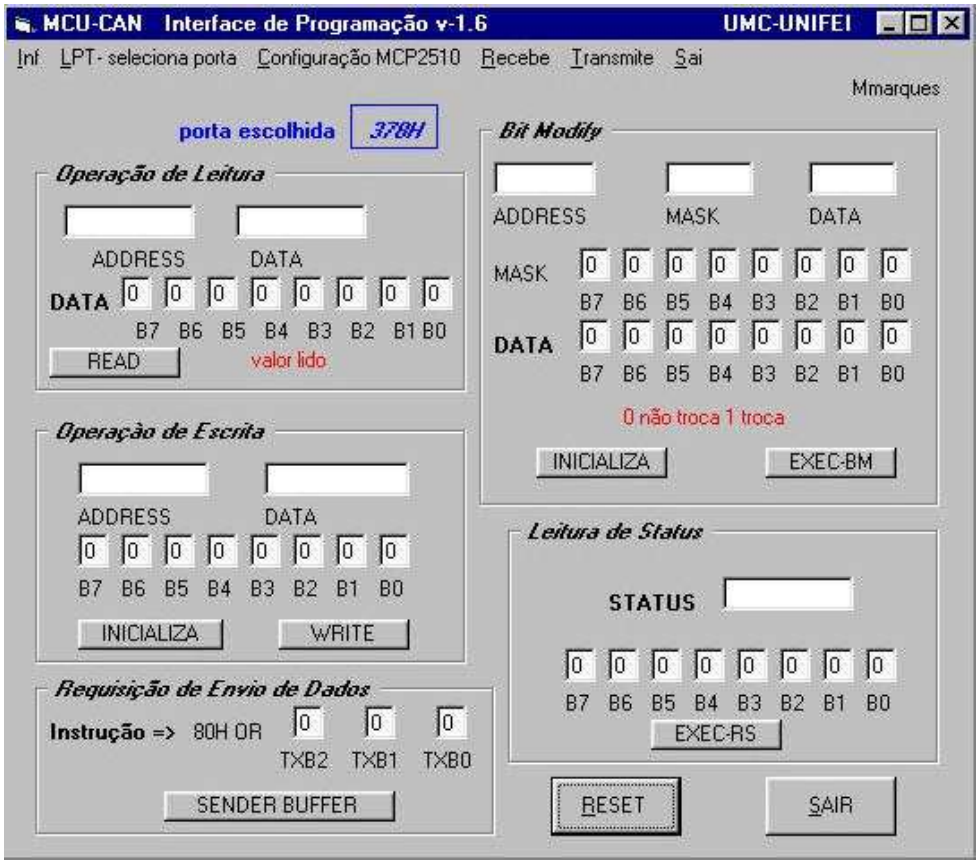


Figura 7.3-Interface de Programação do MCU-CAN

7.3 INTERFACE SPI – PC

A interface entre a LPT do PC e a SPI do MCU-CAN foi feita de acordo com as informações disponibilizadas pela Microchip, por dados fornecidos pelo manual do componente (tabela de temporização, tensão e corrente).

O circuito eletrônico da interface física é mostrado no esquema geral da Figura 7.5, onde tem-se o hardware de conexão entre a porta do PC com os sinais necessários para cada ligação do intertravamento entre o PC e o MCU-CAN para handshaking ou intertravamento lógico. O detalhe do sinal elétrico do clock gerado pelo programa desenvolvido e fornecido para o pino 13 (SCK) do MCU-CAN é mostrado na Figura 7.4.

Figura 7.4 - Oscilograma do sinal de sincronização no MCU-CAN

As informações necessárias para confecção do circuito eletrônico estão presentes no "Stand-Alone CAN Controller with SPI Interface" da Microchip. A partir das informações do fabricante, somadas aos testes realizados em bancada pôde-se compreender como a interface SPI do controlador comunica-se com as portas do PC e como é possível ajustar as temporizações exatas dos diversos sinais gerados, tanto pelo controlador como pelo PC, para realização da transferência de dados do PC para o controlador CAN e vice-versa. Para efetivar a comunicação restou, a partir deste ponto, o desenvolvimento de programas de computador.

7.3.1 Relação do Hardware Padrão da LPT com o Hardware do MCU

As conexões entre pinos da LPT com MCU-CAN foram feitas com um transceptor 74HCT245 para barramento bidirecional de 8 bits que funciona como driver entre o PC e MCU-CAN. Foram utilizados os 7 pinos da porta de dados (saídas digitais acessadas pelo registro de dados da LPT -pinos 2,3,4,5,6,7,e,8) e 4 pinos da porta de estado (entradas digitais acessadas através do registro de "status" da interface paralela – pinos 10,11,12,e,13) do barramento DB25 do PC para a programação da SPI do MCU_CAN. Assim, como resultado, têm-se os sinais do PC que irão ser aplicados ao controlador CAN. Todas ligações entre PC-LPT e o MCU-CAN é representada por EPC_XXX e as ligações entre o transceptor e o MCU-CAN e representada por PC_XXX. Detalhes desses sinais estão presentes no esquema da Figura 7.6 .

A funcionalidade do intertravamento entre o PC o MCP-CAN esta relacionada na Tabela 7.1, onde tem-se todos os sinais necessários para a comunicação do protocolo SPI , direção e outros detalhes.

Estabelecidos os sinais entre a porta paralela do PC através do drive (74HCT245), interliga-se com o microcontrolador CAN 2510. Alguns componentes foram necessários para adequar o intertravamento lógico a nível TTL ao transceptor para o acesso ao barramento CAN. A Figura 7.7 mostra o hardware desenvolvido com base nas informações fornecidas pelo fabricante do microcontrolador para permitir a comunicação no projeto. Foram disponibilizados 7 leds indicadores de ações que podem sinalizar o funcionamento da interface. O uso de 3 chaves ligadas aos pinos do MCU, normalmente abertas permite testar ações de transmissão por hardware. Para o clock utiliza-se um sinal de 20MHz para controlador CAN. A partir deste sinal será gerado o tempo de quanta (TQ) e que contempla todas as necessidades da interface e é adequada para aplicação automotiva.

A partir deste ponto o sinal serial de entrada RX e o sinal serial de saída TX poderão ser acoplados a camada física, ou seja, no barramento CAN. Os sinais RX e TX são aplicados ao transceptor da Philips 82C251, onde o resistor de 120 ohm foi utilizado como elemento terminal para rede CAN como é mostrado pelo o circuito da Figura 7.8.

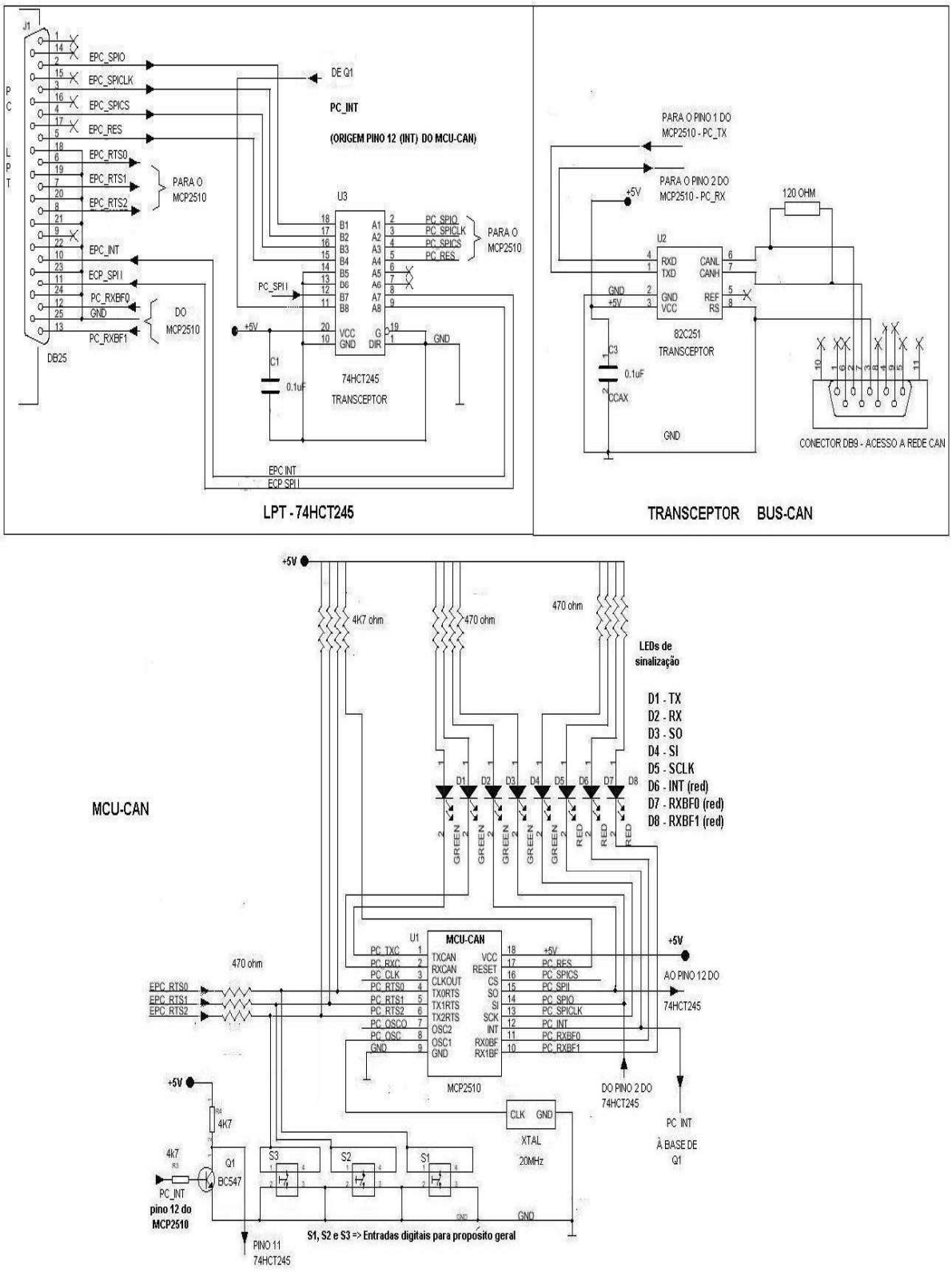


Figura 7.5-Esquema de conexão desenvolvido e aplicado para interfaceamento SPI –LPT [9]

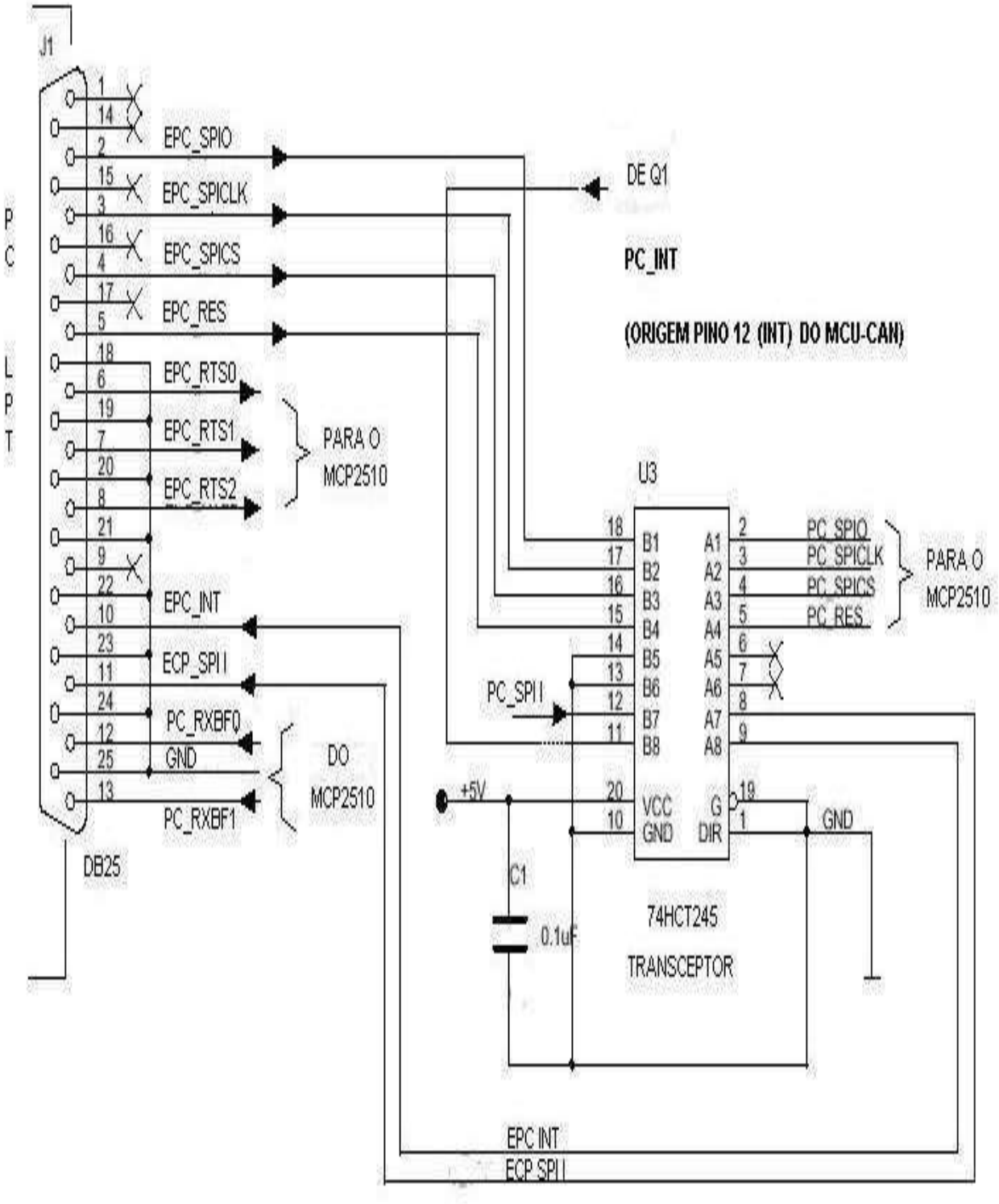


Figura 7.6- Detalhe de ligação da porta LPT com o transceptor bidirecional 74HCT245N[9]

Sinal	Direção	Função	Observação
EPC_SPIO / PC_SPIO	LPT➔ MCU	Entrada de dado serial para MCP2510 (SI) através do registro de dados da LPT.	Através do buffer Bit **D0** (data-LPT)
EPC_SPICLK / PC_SPICLK	LPT➔ MCU	Sinal de clock para o MCP2510	Através do buffer Bit **D1** (data-LPT)
EPC_SPICS / PC_SPI_CS	LPT➔ MCU	Sinal para habilitar o MCP2510	Através do buffer Bit **D2** (data-LPT)
EPC_RES / PC_RES	LPT➔ MCU	Sinal para reset do MCP2510	Através do buffer Bit **D3** (data-LPT)
EPC_SPII / PC_SPII	MCU ➔LPT	Saída de dado serial do MCP2510 (SO) para o registro de estado da LPT	Através do buffer (Status) **BUSY** -(pin 11-LPT)
ECP_INT / PC_INT	MCU➔LPT	Habilita interrupção por hardware	Através do buffer (Status) **ACK** (pin 10-LPT)
EPC_RTS0, EPC_RTS1 e EPC_RTS2 PC_RTS0, PC_RTS1 e PC_RTS2	LPT ➔MCU	Habilita cada um dos buffers de transmissão (TX0RTS, TX1RTS e TX2RTS) por hardware ou entrada digital.	Diretamente do PC para o MCP2510 (Data) Bit **D4** (data –LPT) Bit **D5**(data – LPT) Bit **D6**(data – LPT)
PC_RXBF0 E PC_RXBF1	MCP➔LPT	Sinaliza interrupção de recepção para cada buffer e pode ser utilizado como saídas digitais.	Diretamente do MCP2510 para o PC (Status) **Paper end** (pin 12) **Select in** (pin 13)

Tabela 7.1-Funcionalidade dos vários sinais utilizados na interface PC

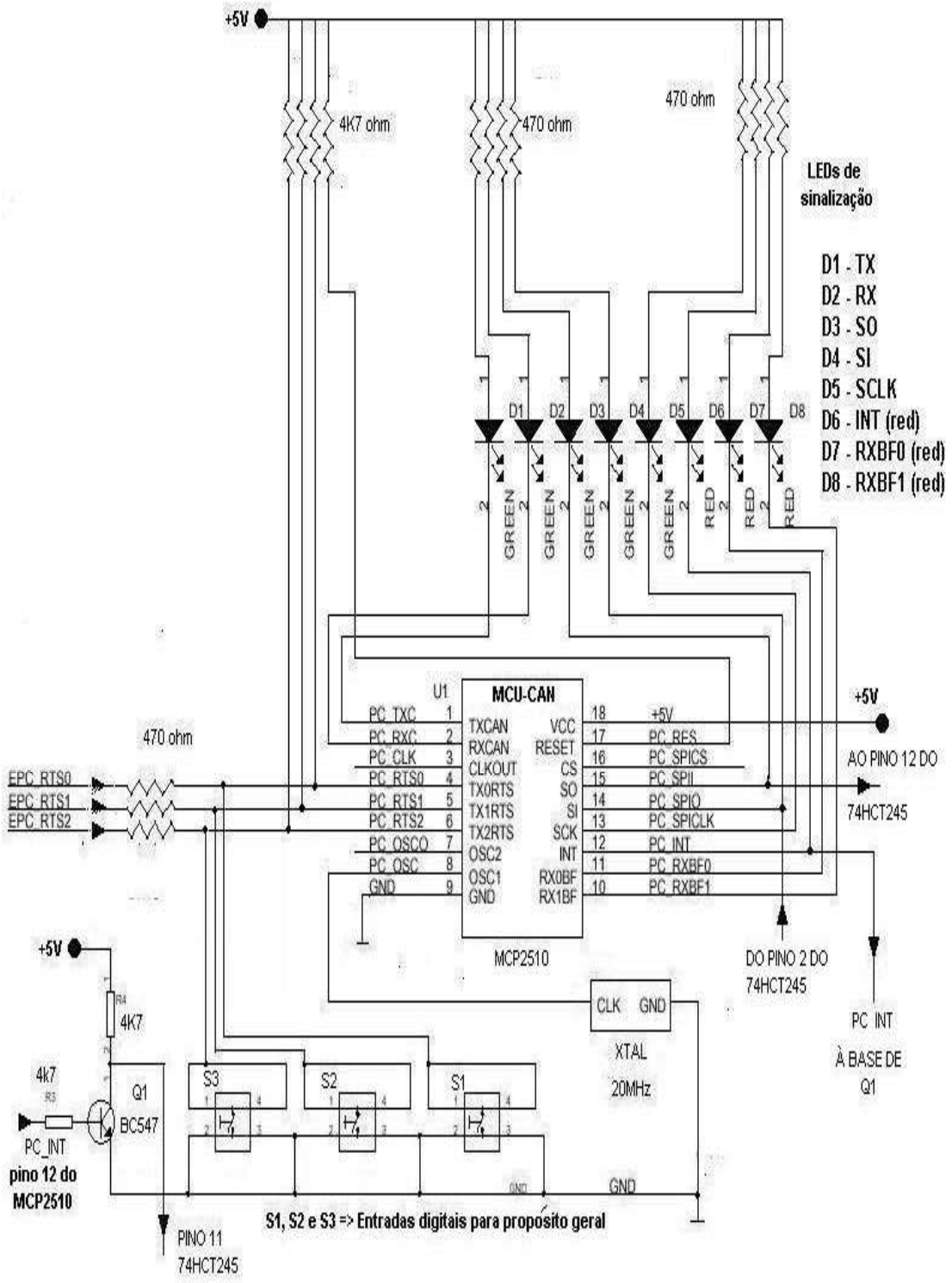


Figura 7.7-Circuito de ligação do processador CAN[9]

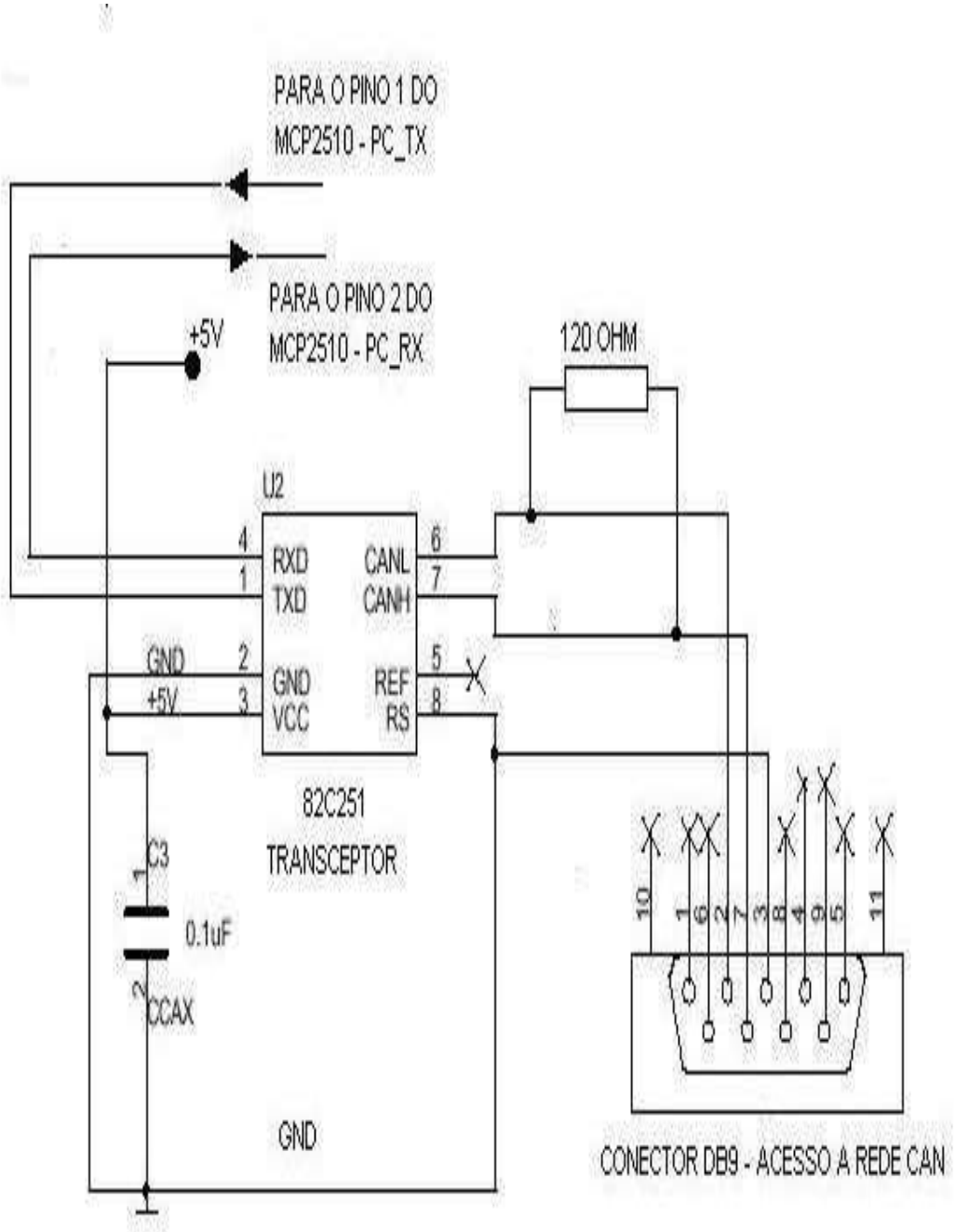


Figura 7.8-Circuito de ligação do transceptor CAN[9]

7.4 CONEXÃO ENTRE A INTERFACE E O CARRO

O acesso à rede CAN embarcada é feito pelo conector DB9 do lado da interface que tem como pinos padrão 2 (CAN-L) e 7 (CAN-H) definidos pela norma SAE J1972. O padrão que determina a estrutura de ligação do OBDII (On Board Data) poderá ser encontrado em algum local estratégico definido pelo fabricante (veja detalhe da Figura 7.9).

Algumas empresas automotivas fornecem manuais ou informações via internet, que orienta o profissional de manutenção a localizar o conector DLC (data link conector) é o caso do "Informativo – Tabela de Aplicações Rasther MPI" fornecido pela empresa Tecnomotor [2].

Figura 7.9-Localização do OBDII

Figura 7.10-Detalhe do OBDII

O chicote CAN foi construído e adaptado para ser utilizado no sistema de monitoramento desenvolvido (detalhe do cabo utilizado na Figura 7.11) a partir de informações do cabo utilizado por equipamentos *scanner*, uma vez que a conexão física obedece o padrão.

Figura 7.11-Detalhe do chicote CAN[6]

O cabo, normalmente utilizado para diagnóstico ISO é composto por um cabo de 2 metros de comprimento com conector macho SAE J1962 [18] do lado do nó do carro e possui 16 pinos. Somente os pinos 6 (CAN-L) e 14 (CAN-H) são utilizados. No outro extremo, lado do PC, utiliza-se um conector tipo fêmea DB9 de 9 pinos dos quais só se utilizam os pinos 2 (CAN-L) e 7 (CAN-H). Os detalhes das funções de cada pino são mencionados na tabela do capitulo 1 (Tabela 1:1) do capitulo 1 onde são descritas as funções de cada pino do lado do carro e todas as possibilidades de conexão com equipamentos compatíveis com ISO 9141, J1850 e CAN.

A conexão mostrada na Figura 7.12 foi feita no sistema automotivo da Fiat modelo Palio Weekend (Fire) já com o cabo construído. O detalhe do conector tipo macho para OBDII é mostrado na Figura 7.13.

Figura 7.12-Detalhe na conexão

Figura 7.13-Detalhe do conector J1962

7.5 SINAIS E TEMPORIZAÇÃO DA INTERFACE

A proposta para o desenvolvimento do hardware visa um processador CAN que possa ser ligado diretamente através da porta Serial Peripheral Interface (SPI) com a porta paralela do PC, utilizando o modo ECP (bi-direcional) da porta paralela. As rotinas para programar o MCU-CAN, resumem-se basicamente nas instruções de escrita e leitura de dados através da interface SPI. Comandos e informações são enviados pelo pino SO do MCU-CAN para a porta paralela através do pino 11 (BUSY) acessado pelo registro de estado que recebe o dado serial. Através de rotinas de programação, o dado é novamente convertido em informação paralela e processado no PC. Cada bit da informação é lido a cada subida do período do sinal de clock (gerado por um programa para o pino SCK do MCU-CAN). Este sinal é gerado por uma rotina de software desenvolvida no PC (pino 3 do registro de dados). As informações na linha de entrada SI do MCU-CAN são direcionadas para o PC através da porta LPT pelo pino 2 (D0 do registro de dado), na descida do sinal de clock (pino SCK). O pino CS do MCU-CAN deve ser mantido baixo durante qualquer uma das operações. A Tabela 7.2 mostra os bytes de todas as instruções para o MCU-CAN.

Instruction Name	Instruction Format	Description
RESET	1100 0000	Resets internal registers to default state, set configuration mode
READ	0000 0011	Read data from register beginning at selected address
WRITE	0000 0010	Write data to register beginning at selected address
RTS (Request To Send)	1000 0nnn	Sets TXBnCTRL.TXREQ bit for one or more transmit buffers 1000 0nnn Request to send for TXB2 — Request to send for TXB0 Request to send for TXB1
Read Status	1010 0000	Polling command that outputs status bits for transmit/receive functions
Bit Modify	0000 0101	Bit modify selected registers

Tabela 7.2-Conjunto de instruções SPI[7]

O diagrama de temporização para entrada (SI) é mostrado na Figura 7.14 onde pode-se ver a fase dos demais sinais, ou seja, “chip select –CS” e o sinal de “clock – SCK. O mesmo deve ser analisado para as temporizações para a saída (SO) e sinais CS e SCK, mostrados na Figura 7.15.

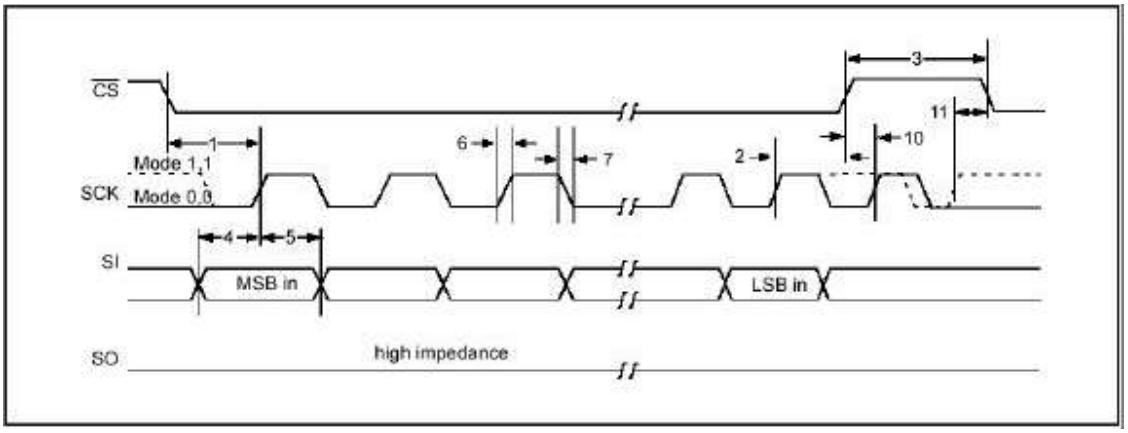


Figura 7.14-Tempos da entrada SPI[7]

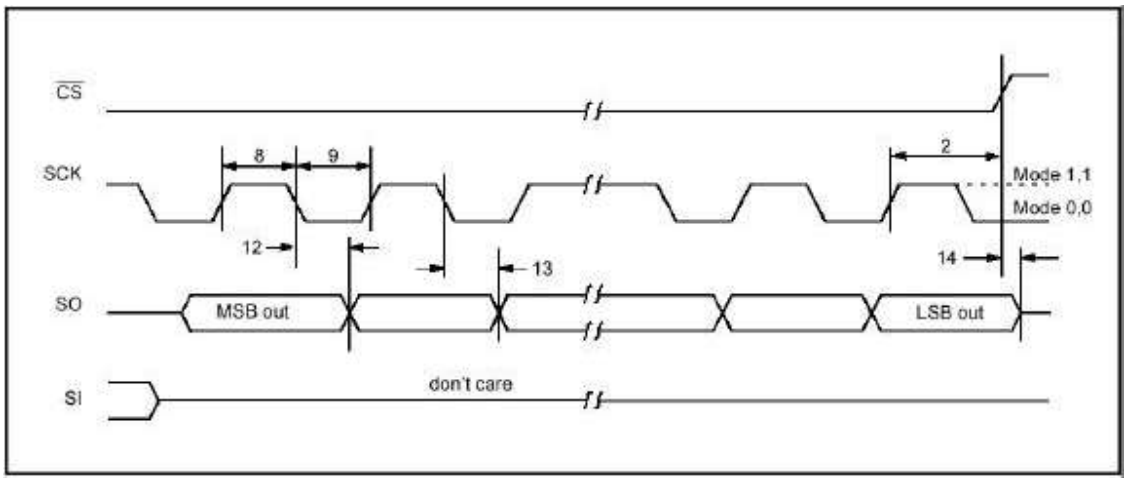


Figura 7.15-Tempos da saída SPI[7]

O sinal de clock para o microcontrolador CAN (SCK) é gerado por uma rotina de programação. Assim sendo, a geração do período do clock é baseada no mapa de tempo fornecido pela Microchip através da Tabela 7.3, onde o valor da metade do período é maior que 115ns. Nas rotinas de escrita e leitura do MCU-CAN é introduzido um laço "for..next" associado a uma função *time_a (valor)* onde poderá ser ajustar o período de clock para o MCU-CAN.

SPI Interface AC Characteristics			Industrial (I): TAMB = -40°C to +85°C VDD = 3.0V to 5.5V Extended (E): TAMB = -40°C to +125°C VDD = 4.5V to 5.5V			
Param. No.	Sym	Characteristic	Min	Max	Units	Conditions
	FCLK	Clock Frequency	— — —	5 4 2.5	MHz MHz MHz	VDD = 4.5V to 5.5V VDD = 4.5V to 5.5V (E temp) VDD = 3.0V to 4.5V
1	TCSS	$\overline{CS}$ Setup Time	100	—	ns	
2	TCSH	$\overline{CS}$ Hold Time	100 115 180	— — —	ns ns ns	VDD = 4.5V to 5.5V VDD = 4.5V to 5.5V (E temp) VDD = 3.0V to 4.5V
3	TCSD	$\overline{CS}$ Disable Time	100 100 280	— — —	ns ns ns	VDD = 4.5V to 5.5V VDD = 4.5V to 5.5V (E temp) VDD = 3.0V to 4.5V
4	TSU	Data Setup Time	20 20 30	— — —	ns ns ns	VDD = 4.5V to 5.5V VDD = 4.5V to 5.5V (E temp) VDD = 3.0V to 4.5V
5	THD	Data Hold Time	20 20 50	— — —	ns ns ns	VDD = 4.5V to 5.5V VDD = 4.5V to 5.5V (E temp) VDD = 3.0V to 4.5V
6	TR	CLK Rise Time	—	2	µs	**Note**
7	TF	CLK Fall Time	—	2	µs	**Note**
8	THI	Clock High Time	90 115 180	— — —	ns ns ns	VDD = 4.5V to 5.5V VDD = 4.5V to 5.5V (E temp) VDD = 3.0V to 4.5V
9	TLO	Clock Low Time	90 115 180	— — —	ns ns ns	VDD = 4.5V to 5.5V VDD = 4.5V to 5.5V (E temp) VDD = 3.0V to 4.5V
10	TCLD	Clock Delay Time	50	—	ns	
11	TCLE	Clock Enable Time	50	—	ns	
12	TV	Output Valid from Clock Low	— — —	90 115 180	ns ns ns	VDD = 4.5V to 5.5V VDD = 4.5V to 5.5V (E temp) VDD = 3.0V to 4.5V
13	THO	Output Hold Time	0	—	ns	**Note**
14	TDIS	Output Disable Time	—	200	ns	**Note**

Tabela 7.3-Característica AC da interface SPI do MCP2510[7]

7.5.2 **Comando de Escrita (Write)**

A Figura 7.16 mostra a temporização de uma rotina de escrita (write) através da SPI. Este comando, baseado em uma instrução de escrita, inicia-se pela descida do sinal de controle do pino CS do MCU-CAN fornecido pelo PC. A instrução de escrita (02H) é então enviada ao microcontrolador CAN, seguida pelo endereço e pelo menos um byte de dados. É possível escrever vários bytes em uma seqüência de registradores deste que o sinal de clock continue a ser enviado ao pino de controle do MCU-CAN enquanto o outro sinal de controle permaneça baixo. Cada bit do dado é escrito nos registradores no momento da subida do sinal de controle aplicado no pino SCK do MCU-CAN. Então cada bit é armazenado ou escrito nos registradores internos através do pino SI do MCU-

CAN, sinal este fornecido pela porta paralela através do bit D0 do registro de dados. Portanto, para cada byte a ser escrito, 8 estados ou períodos de clocks são necessários. Se o sinal de controle CS do controlador CAN for para nível alto, o processo de escrita será abortado.Verifica-se que este comando estabelece o período de tempo exato do sinal SCK enviado pelo pino 3 da LPT de acordo com as características elétricas do MCU-CAN. Os sinais de controle necessários são vistos no diagrama da Figura 7.17. A Rotina 7.2 é o código fonte de programação em VB em que estabelece os tempos e sinais necessários para uma operação de escrita (WRITE).

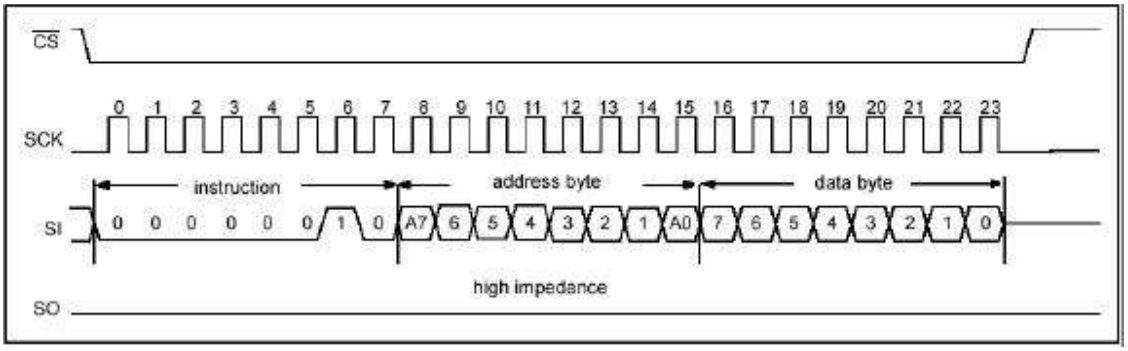


Figura 7.16-Temporização para instrução de escrita (write) pela SPI[9]

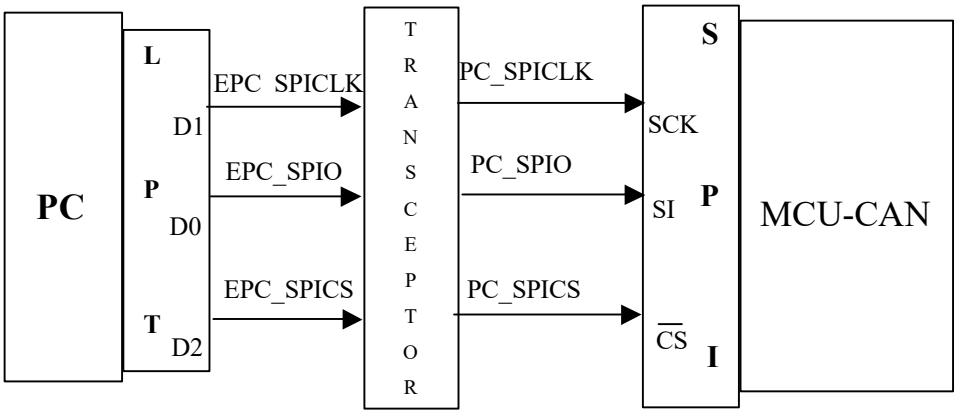


Figura 7.17-Diagrama de blocos com detalhes do hardware de ligação PC e MCU-CAN

Para manipulação dos dados e constantes foram criadas as variáveis da Rotina 7.1 que foram utilizadas em todas as demais rotinas. Uma vez que o byte deve ser escrito na SPI e esta deverá acolher cada um dos 8 bits serialmente, desenvolveu-se a rotina sub rotina definida na Rotina 7.3, para que cada bit do byte fosse escrito no controlador CAN.

```
Option Explicit
    'Constantes e variáveis utilizadas
    'porta lpt1
    'Entradas para controle de saída:
    Const SPICLK = 2:  'bit 1 - PC_SPICLK da porta de dados
    Const SPICS = 4:  'bit 2  - PC_SPICS da porta de dados
    Const RESET = 8:  'bit 3  - PC_RES da porta de dados
    Const RTS0 = &H10:  'bit 4 - requisição de transmissão buffer TXB0
    Const RTS1 = &H20  'bit 5 - requisição de transmissão buffer TXB1
    Const RTS2 = &H40  'bit 6 - requisição de transmissão buffer TXB2
    Dim BaseAddress%  'variável que armazena endereço base da LPT
    Dim dataw% 'variável utilizada para armazenar dados para escrita em geral
    Dim datar% 'variável utilizada para armazenar dados lidos.
    Dim datin% 'variável utilizada para armazenar dados secundários lidos.
    Dim bitnumber% 'variável utilizada para armazenar byte correspondente ao bit que se deseja controlar
    Dim dataw1% 'É utilizada para armazenar temporariamente dados para controle.
    Dim SPIO% 'É utilizada para acessar ao bit de dado D0 - saida para a SPII
    'Variáveis em geral, utilizadas para dados numéricos e caracteres.
    Dim prompt$
    Dim reply$
    Dim Search$
    Dim cont%
```

Rotina 7.1- Declaração de variáveis e constantes utilizadas no programa desenvolvido

Uma vez criada a Rotina 7.3, ela pode ser chamada para escrever qualquer informação (byte) no controlador CAN, através da Rotina 7.2.

A função definida pela Rotina 7.4, foi desenvolvida para recuperar bit a bit da informação ou byte. Um exemplo do seu uso pode ser visto na Rotina 7.4, que executa uma operação de escrita.

```
Private Sub cmd_Wr_Click()

    'Operação de escrita de informação da tela principal
    'Define as variáveis
    Dim bitnumber2%
    Dim bit_cont%
    Dim data%
    data = 0
    'data é a variável que deverá ser escrita no MCU-CAN através da SPI
   'Converte em byte os bits do dado que se deseja escrever no MCU-CAN.
    For bitnumber2 = 7 To 0 Step –1
          bit_cont = txt_Con_wr(bitnumber2).Text
          'recupera bit a bit
          data = (data + bit_cont * 2 ^ bitnumber2)
    Next
   'Armazena o valor convertido na variável byte
    txtCont_Wr.Text = (Hex$(data))
    'mostra informação convertida em hexadecimal no objeto txtCont_Wr
     Dim prompt, title As String
   title = "Escrevendo nos Registradores do 2510"
```

```
    prompt = "...entre com o endereço em HEX"
    dataw3 = Cint("&H" & InputBox(prompt, title))
    'variável inteira dataw3 armazena endereço para escrita em hexadecimal.
    prompt = "...entre com o dado em HEX"
    txtAd_Wr.Text = Hex$(dataw3)
     'mostra endereço na tela de escrita
    txtCont_Wr.Text = Hex$(data)
    dataw1 = SPICS + RESET
     'armazena na variável os bits a serem aplicados nos pinos CS e RESET do MCU-CAN
    BitReset dataw1, 2   'CS vai para baixo para selecionar CHIP
    DataPortWrite BaseAddress, dataw1
    'coloca byte dataw1 com valor para selecionar chip
    'MCU-CAN habilitado para escrita
    WriteData &H2
    'escreve instrução 02H de escrita no MCU-CAN através do SI
    WriteData dataw3
    'define o valor do endereço do byte
    WriteData data
    'escreve o conteúdo no endereço definido acima
    BitSet dataw1, 2 'CS vai para alto para desabilitar CHIP
    DataPortWrite BaseAddress, dataw1
    'desabilita o MCU-CAN através do pino CS

End Sub
```

Rotina 7.2-Escrita de dados no MCU-CAN através da interface SPI – operação write

```
Private Sub WriteData(DataToWrite%)

     'Sub-rotina desenvolvida para escrever  informação na SPI do MCU-CAN
    'determina o estado de todos os bits da variável DataToWrite
     For bitnumber = 7 To 0 Step –1
          SPIO = BitRead(DataToWrite, bitnumber)
         'define o estado do bit (0 ou 1) da informação a ser lida
         'A função criada BitRead(DataToWrite,bitnumber) define o estado do bit
          'do byte DataToWrite – veja rotina abaixo
          dataw2 = dataw1 Or SPIO
          'BitReset dataw2, 1 'clock baixo
          'envia bits para pino SI, CS e CLK do MCU-CAN
           DataPortWrite BaseAddress, dataw2
           DataPortWrite BaseAddress, dataw2 + SPICLK 'subida do sinal clock
           time_a (6) 'temporizador – matem clock alto
           'seta o pino CLK do 2510 – ½  estado ou ½ período
          DataPortWrite BaseAddress, dataw1
          'proximo bit de data
    Next bitnumber

End Sub
```

Rotina 7.3-Escrita de cada bit de um byte na SPI e geração do sinal de clock

```
Function BitRead%(Variable%, bitnumber%)
          'rotinas criadas para apoio ao projeto do MCU-CAN
```

```
        'Devolve o valor do estado do bit (0 ou 1) na informação definida na variável  "Variable".
        Dim BitValue%
        BitValue = 2 ^ bitnumber  'o valor do byte apenas para o bit considerado
        BitRead = (Variable And BitValue) \ BitValue 'valor do bit 0 ou 1
       'exemplo se Variable=64 ; o bitnumber = 6
       'portanto BitValue= 2^6 =64
       'BitRead= (64 And 64) / 64 = 1
        'para qualquer outro valor do bitnumber o resultado será igual a 0
End Function
```

Rotina 7.4-Função que define o estado do bit de uma informação

Os sinais gerados pelo computador pelas rotinas desenvolvidas em Visual Basic foram aplicados nos pinos de controle da SPI de modo sincronizado, como recomendado pelo fabricante do controlador CAN, para que as informações (dados e comandos) recebidas pelo microcontrolador de CAN possam ser armazenadas corretamente para processamento interno.

7.5.3 Comando Leitura (Read)

A Figura 7.18 mostra a temporização de uma rotina de read (leitura) através da SPI. Verifica-se que a instrução de leitura inicia-se pela descida do sinal no pino CS do MCU-CAN, e portanto a instrução de leitura (READ) é enviada para o MCU-CAN seguida pelos 8 bits de endereçamento (A7 até A0). Após enviar a instrução de leitura, a informação armazenada no registrador de endereço selecionada é deslocada serialmente para fora através do pino SO. O endereço interno do ponteiro é automaticamente incrementado para o próximo endereço após cada byte da informação ser deslocado para fora. Por esta razão é possível ler o endereço do registro consecutivo desde que tenha pulsos de clock disponíveis. Assim sendo é possível ler um registrador qualquer usando este método. A operação é terminada na subida do sinal de controle fornecido ao pino CS do MCU-CAN. Verifica-se que para este comando tem-se o período de tempo exato do sinal de clock para o pino SCK do microcontrolador enviado pelo pino 3 da LPT, da mesma forma que foi citado para o comando de escrita, regido pelas características elétricas do MCP2510. O esquema de ligação entre o PC e o MCU-CAN na operação de leitura é mostrado no diagrama de blocos da Figura 7.19. A Figura 7.18 apresenta os tempos e sinais necessários para uma operação de leitura (READ).

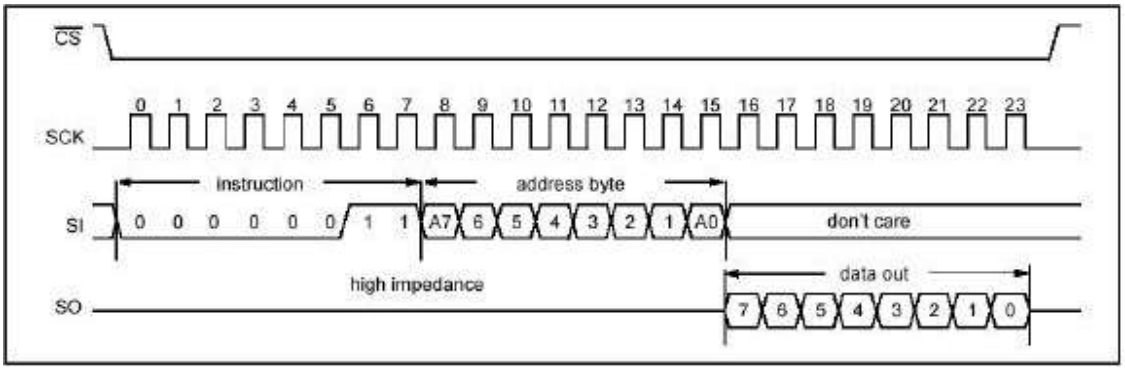


Figura 7.18-Temporização para instrução de leitura (read) pela SPI[7]

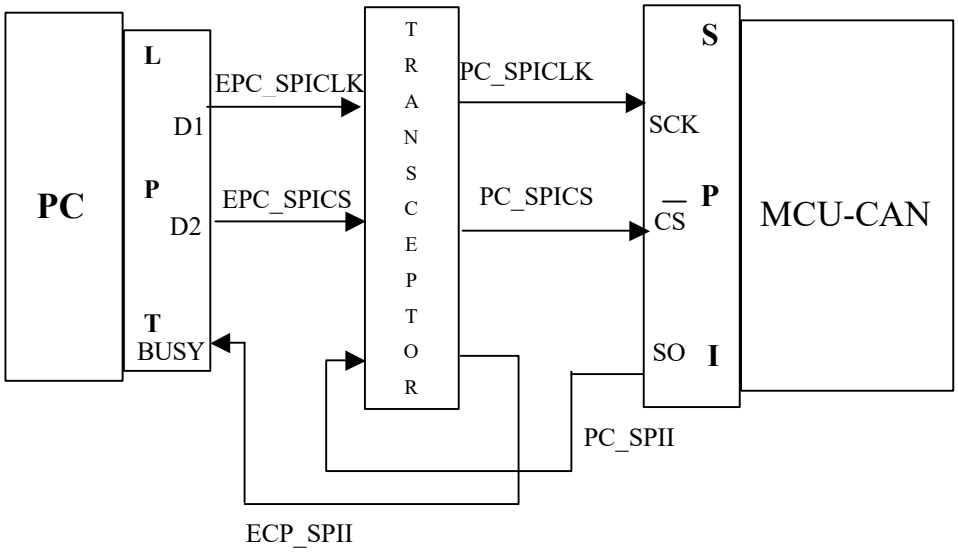


Figura 7.19-Diagrama de blocos com detalhes do hardware de ligação PC e MCU-CAN para instrução de leitura (READ)

O código fonte em VB mostrado na Rotina 7.5 foi desenvolvido para executar uma operação de leitura a partir da tela de teste desenvolvido em Visual Basic. Definido o endereço de leitura desejado é possível ler qualquer informação de qualquer registrador interno do MCU-CAN através desta rotina.

```
Private Sub cmd_Rd_Click()
  Dim datin% 'define variável que recebe informação apos leitura.
  Dataw1 = RTS0 + RTS1 + RTS2 + SPICS + RESET 'define o estado da SPI no 2510
  BitReset dataw1, 2 'chip select baixo
  Dim prompt, title As String
  title = "Lendo os Registradores do 2510"
```

```
  prompt = "...entre com o endereço do registro em HEX"
  dataw = Cint("&H" & InputBox(prompt, title))
  'dataw recebe o endereço via teclado para receber o byte de entrada após a operação de leitura
  DataPortWrite BaseAddress, dataw1
  WriteData &H3 'instrução de leitura
  txtAd_inst_rd.Text = Hex$(dataw)
  WriteData dataw 'endereço para receber byte
  datin = ReadData 'leitura do byte
  txtCont_rd.Text = Hex$(datin) 'mostra byte na interface visual
  BitSet dataw1, 2 'desativa CS
  DataPortWrite BaseAddress, dataw1 'atua no MCU-CAN
  Dim bit_cont%'converte byte lido em representação binária para interface visual
  Dim bitnumber2%
  For bitnumber2 = 0 To 7
      bit_cont = BitRead(datin, bitnumber2)
      txtDataR(bitnumber2).Text = bit_cont
  Next
End Sub
```

Rotina 7.5-Leitura de dados (bytes)

7.5.4 Instrução de Reinicialização (Reset)

Na interface gráfica foi programado um botão para reinicializar a Interface de Teste através da instrução 03H aplicada através do pino SI do controlador CAN. Com isto os registradores do MCU-CAN são ajustados e o circuito vai para o modo de configuração. O diagrama de blocos da Figura 7.20 mostra que também é possível reinicializar o controlador CAN aplicando um nível baixo de tensão no pino RESET através da interface SPI.

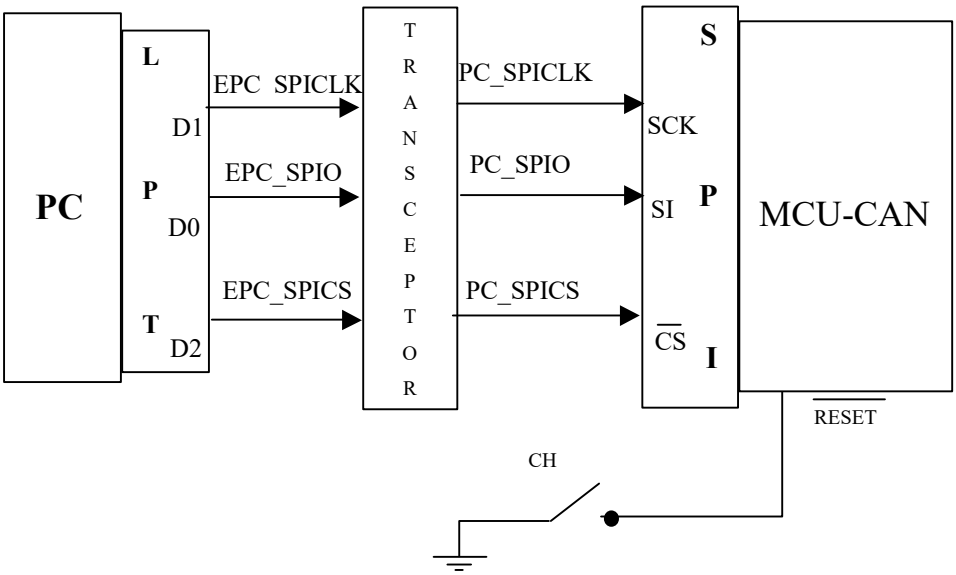


Figura 7.20 -Diagrama de blocos para uma ação de Reset por hardware e por software

A instrução de Reset é um byte simples e requer que neste momento seja aplicado um nível baixo de tensão no pino CS do controlador CAN enquanto o byte de instrução é enviado para o controlador, como é mostrado no diagrama de tempo da Figura 7.21. A Rotina 7.6 em linguagem fonte em VB, foi desenvolvida para produzir a ação descrita acima. Verifica-se a necessidade de sincronização dos sinais para que a instrução atue no MCU-CAN.

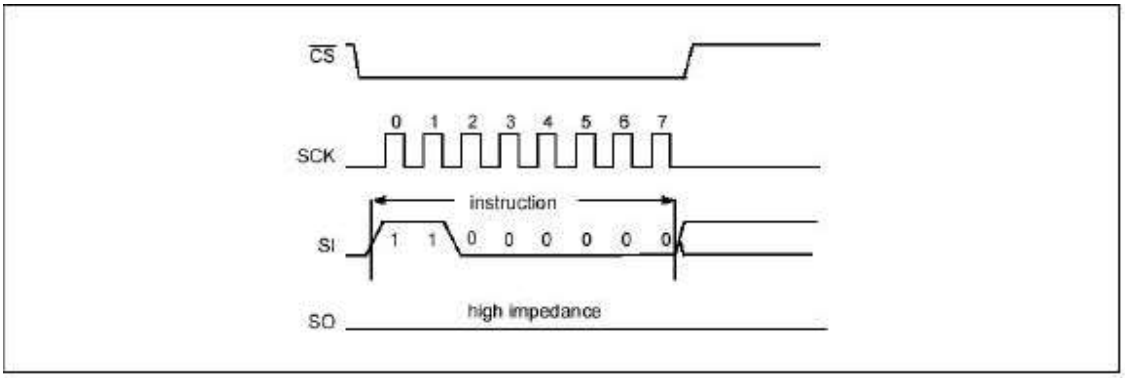


Figura 7.21-Temporização para instrução de Reset[7]

```
Private Sub cmdReset_Click()

    MsgBox "Esta ação produz um reset físico - ATIVA MODO CONFIGURAçÃO"
    dataw1 = SPICS + RESET ‘sinal para reset
```

```
        BitReset dataw1, 3 'reset=0 -fisico
        BitReset dataw1, 2 'chip select baixo
        DataPortWrite BaseAddress, dataw1
        BitSet dataw1, 3 'reset alto
        DataPortWrite BaseAddress, dataw1
        BitSet dataw1, 2 'chip select alto
        DataPortWrite BaseAddress, dataw1
        WriteData &HC0 'reset lógico
        BitSet dataw1, 2 'chip select alto
        DataPortWrite BaseAddress, dataw1

End Sub
```

Rotina 7.6-Programa fonte para ação de Reset

7.5.5 Requisição de Envio de Dados

Na Tela de Teste (Figura 7.22) foi acrescentada uma opção de escolha para transmitir dados armazenados nos buffers de transmissão. Para isto pode-se programar ou escolher quais dos buffers de transmissão será(ão) utilizado(s). O comando RTS é utilizado para iniciar o processo de transmissão de um dos três ou até três buffers de transmissão do MCU-CAN. Para envio de um frame, para o MCU-CAN o sinal no pino CS do controlador é colocado em nível baixo juntamente com o byte da instrução do comando RTS, como mostrado na. Pelo diagrama de tempo da Figura 7.24 verifica-se que os três últimos bits do comando indicam qual será o buffer habilitado a transmitir (80H OR TXB2 TXB1 TXB0-detalhe na Figura 7.22).

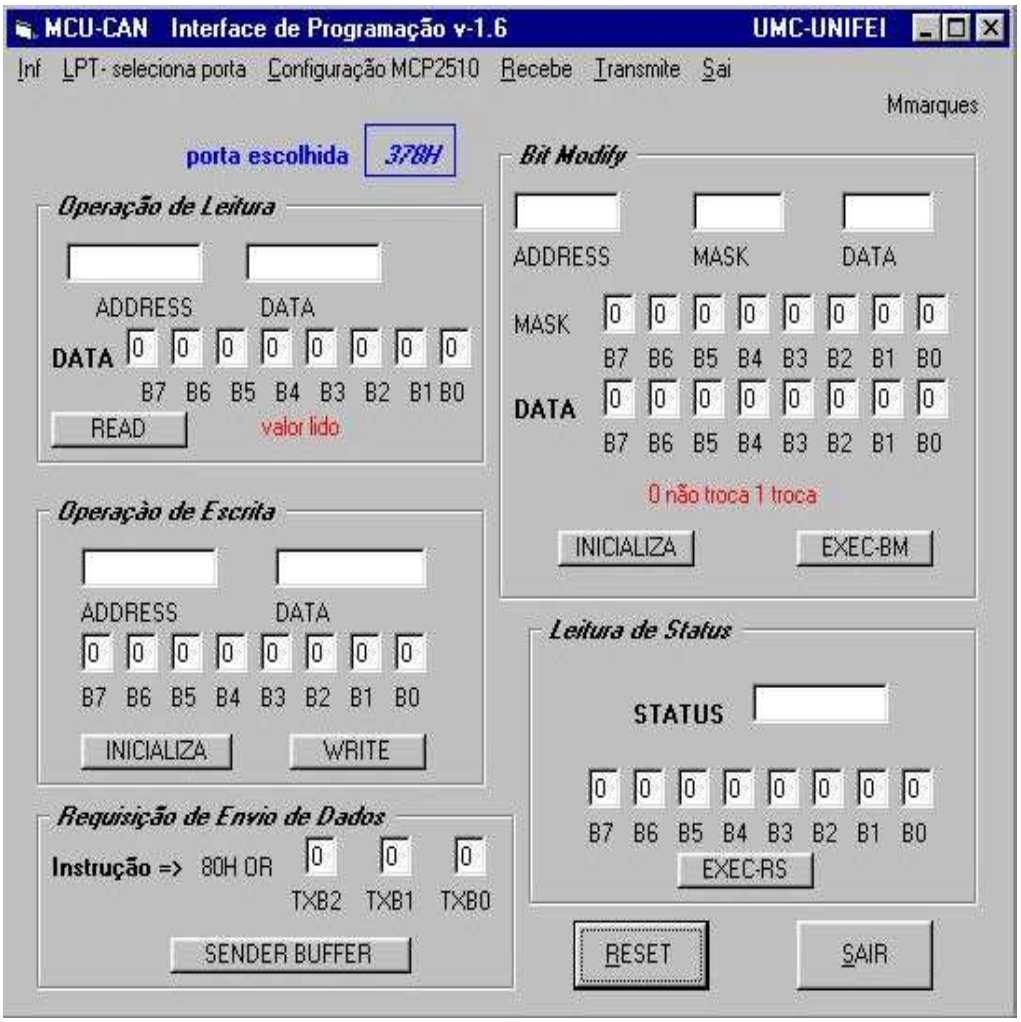


Figura 7.22 – Interface de Programação – detalhe do botão "*SENDER BUFFER*" instrução RTS

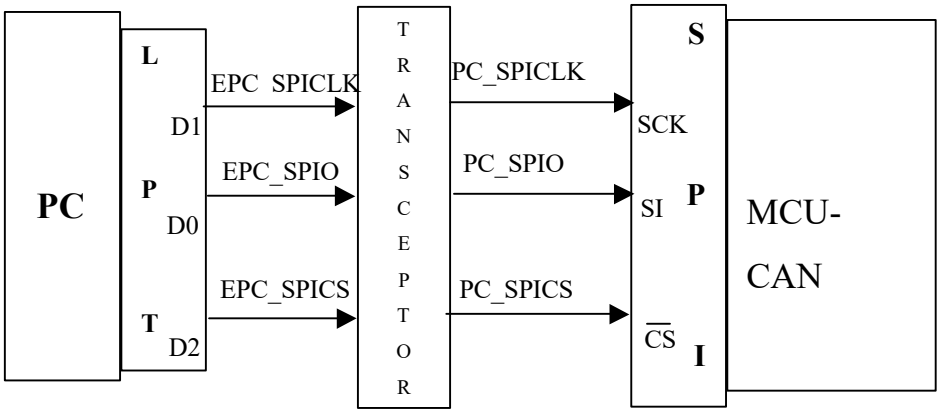


Figura 7.23 -Detalhe do hardware para o instrução RTS

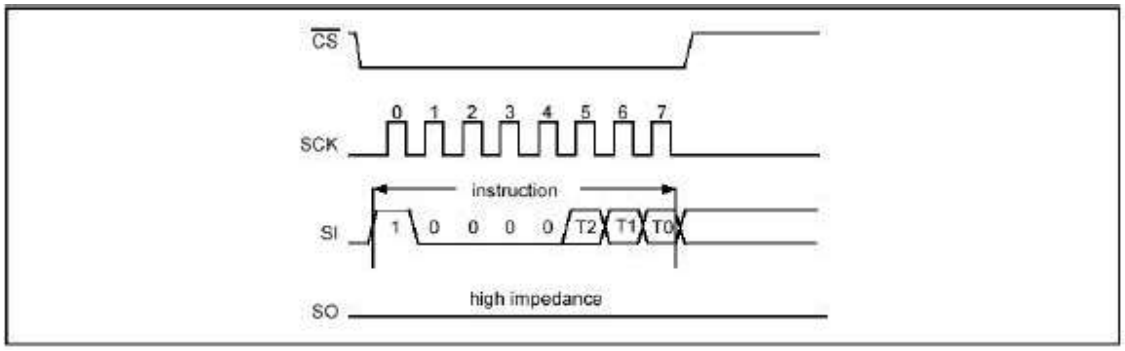


Figura 7.24-Temporização para a instrução RTS[7]

O comando RTS irá ligar o bit do registro TxBnCTRL.TXREQ para o seu respectivo(s) buffer(s). Qualquer bit dos três bits pode ser programado isoladamente ou mesmo tempo. A Rotina 7.7 mostra o programa fonte desenvolvido para habilitar a transmissão.

```
Private Sub cmdSend_Instr_Click()
        MsgBox ("..defina qual buffer vai transmitir")
        Dim bit_T%, bitnumber2%, bit_cont%;
         'bit_T variável que define qual buffer vai transmitir
        bit_T = 0
        'rotina de conversão binario para decimal
        For bitnumber2 = 0 To 2
                bit_cont = txtTrans(bitnumber2).Text
                bit_T = (bit_T + bit_cont * 2 ^ bitnumber2)
        Next
        dataw1 = SPICS + RESET
        BitReset dataw1, 2 'chip select baixo
        dataw = bit_T Or &H80 'associa bit_T com 80H
        DataPortWrite BaseAddress, dataw1
        WriteData dataw
        BitSet dataw1, 2
        DataPortWrite BaseAddress, dataw1
End Sub
```

Rotina 7.7-Programa fonte em VB para o botão “Sender Buffer”, da Tela de Teste

7.5.6 Modificando o Estado dos Bits de Registradores

Pode-se alterar um determinado estado lógico de um determinado bit de alguns registros no controlador CAN, ou seja, ligar ou desligar bit de registros de controle. Como nos outros comandos, a estrutura de hardware mostrada na Figura 7.25, habilita o envio dos dados de instrução: o byte da instrução é enviado primeiro e logo após, é enviado o endereço do registro seguidos pelos bytes da máscara e pelo data byte. O diagrama de tempo deste comando é visto na Figura 7.26 . O nível lógico '1' no byte máscara permitirá mudança no bit correspondente do registrador: e se for um '0 ', não haverá mudança. Um '1' no byte de informação ligará o bit e um '0' limpará o bit, mesmo, que a mascara para este bit esteja ligado ('1'). No exemplo da Figura 7.27 é dado um exemplo da ação do comando Bit Modify sobre o conteúdo de um registrador.

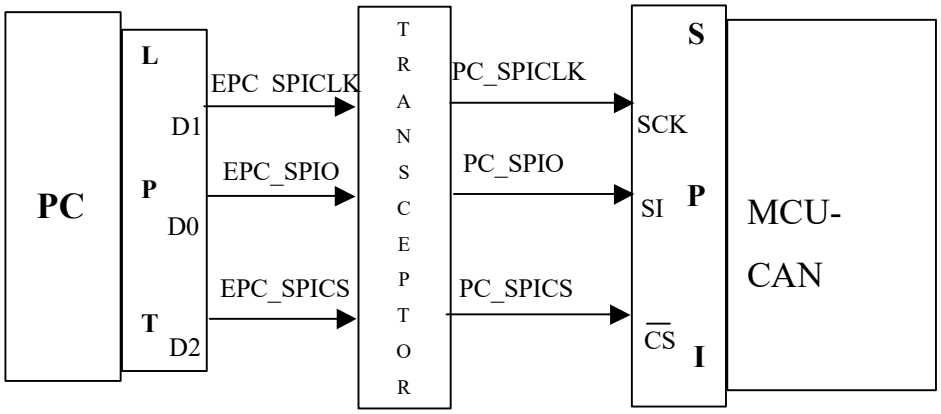


Figura 7.25-Detalhe do hardware para a instrução do Bit Modify

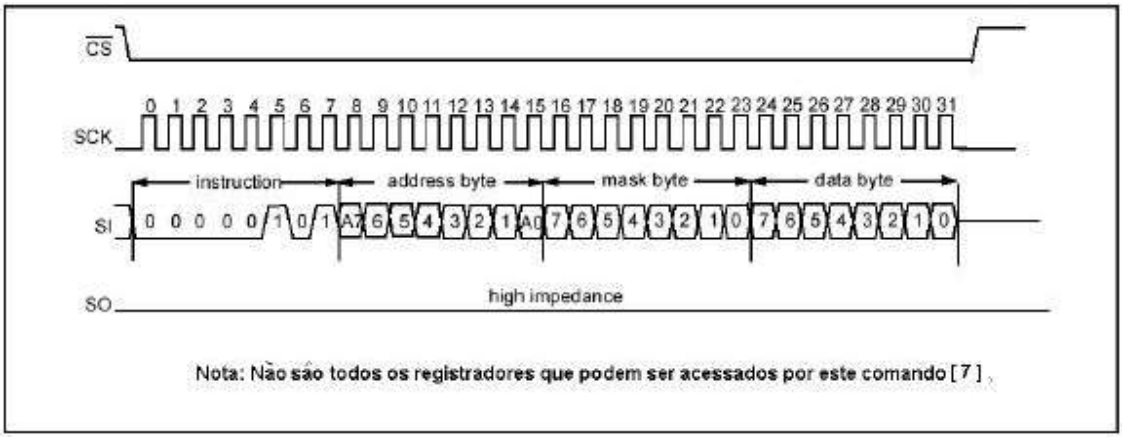


Figura 7.26-Temporização para a instrução do bit modify[7]

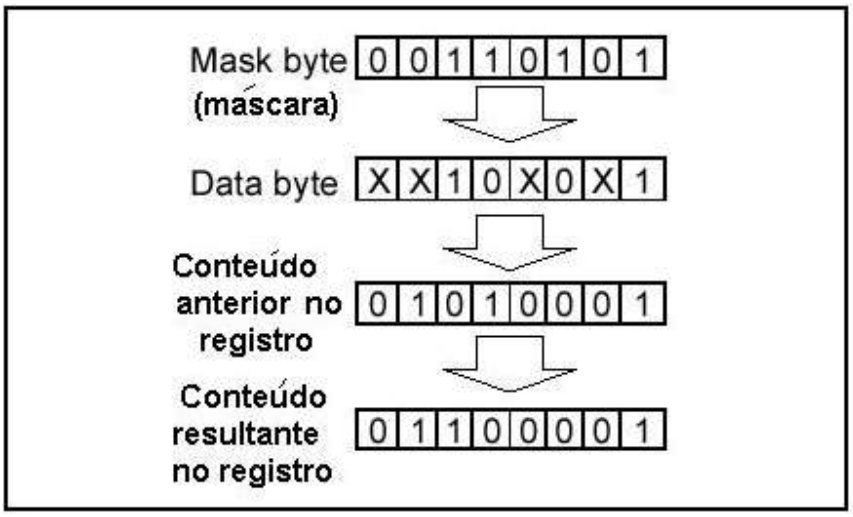


Figura 7.27-Exemplo da ação da instrução do Bit Modify em um conteúdo de registro qualquer[7]

Foi desenvolvida uma sub rotina para modificar o estado de um bit qualquer através da Tela de Teste cuja o código fonte é mostrado na Rotina 7.8.

```
Private Sub cmdBM_Click()

    Dim prompt, title As String
    title = "Modificando os bits dos registros possíveis do MCU-CAN"
    prompt = "...entre com o endereço do registro em HEX"
    dataw = CInt("&H" & InputBox(prompt, title))
    txtAdd_BM.Text = Hex$(dataw)
    Dim bitnumber2%
    Dim bit_cont%
    Dim datas1%
    Dim datas2%
    datas1 = 0
    For bitnumber2 = 7 To 0 Step -1
       bit_cont = txtMask_bit(bitnumber2).Text
       datas1 = (datas1 + bit_cont * 2 ^ bitnumber2)
    Next
    txtMask_BM.Text = (Hex$(datas1))
    datas2 = 0
    For bitnumber2 = 7 To 0 Step -1
    bit_cont = txtData_bit(bitnumber2).Text
    datas2 = (datas2 + bit_cont * 2 ^ bitnumber2)
```

```
        Next
        txtData_BM.Text = (Hex$(datas2))
        dataw1 = SPICS + RESET
        BitReset dataw1, 2 'chip select baixo
        DataPortWrite BaseAddress, dataw1
        WriteData &H5 ‘05H é o valor hexadecimal da instrução para “BIT MODIFY”
        WriteData dataw
        WriteData datas1
        WriteData datas2
        BitSet dataw, 2
        DataPortWrite BaseAddress, dataw1

End Sub
```

Rotina 7.8- Código fonte para o modificar estado de um bit qualquer

7.5.7 Leitura de Estado dos Registros do Controlador CAN

Pode-se ler o estado lógico de cada bit de alguns registros no controlador CAN. Isto é importante para avaliar uma ocorrência antes e após o processo de transmissão e recepção da informação. A instrução de leitura de estado deverá ser enviada ao processador CAN obedecendo a temporização mostrada na Figura 7.29 e habilitada pelo hardware estabelecido no diagrama da Figura 7.28. Cada um dos estados lidos poderá ser comparado por uma leitura padrão nos endereços dos registradores do controlador CAN através da Tela de Teste. O programa fonte em VB da Rotina 7.9, foi desenvolvido para executar a leitura de estado dos principais registradores internos do MCU-CAN.

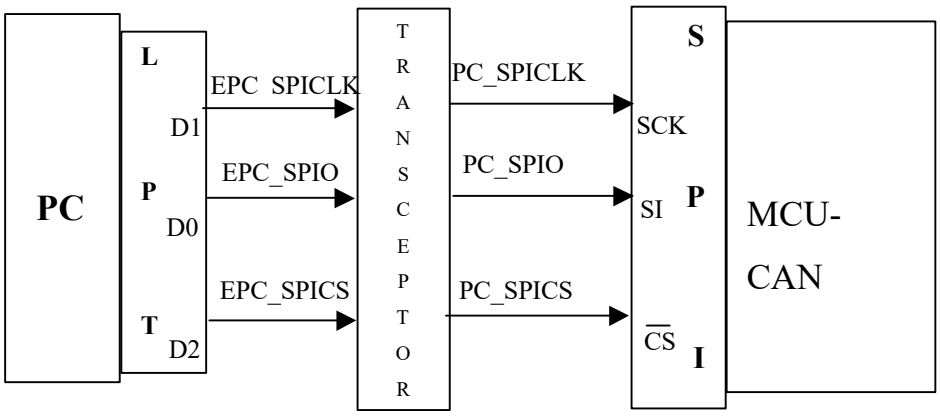


Figura 7.28-Sinais atuantes na operação de leitura de estado

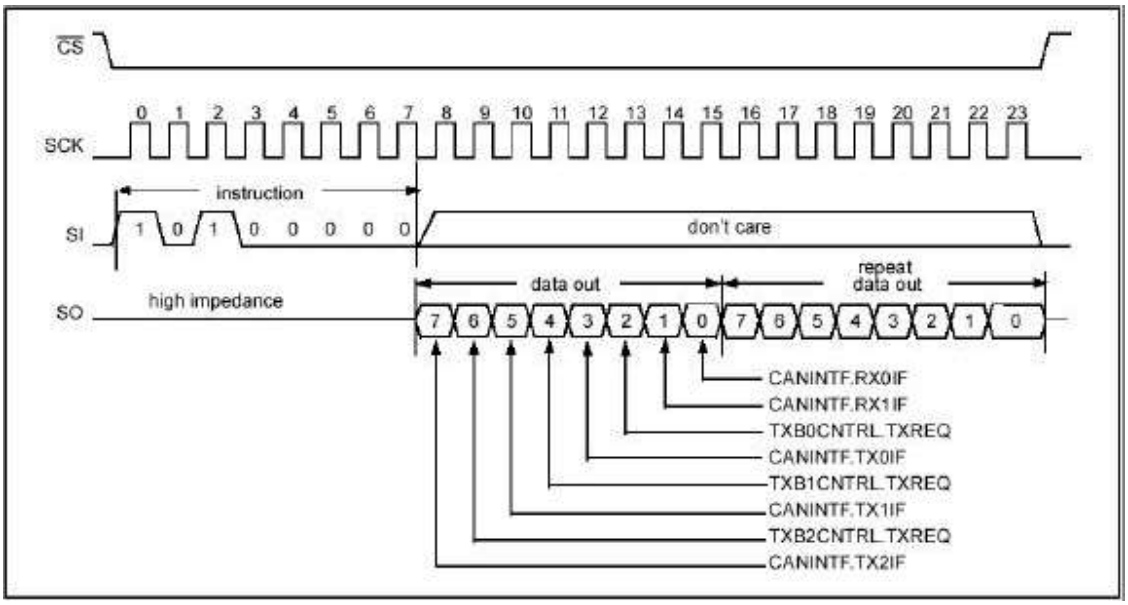


Figura 7.29-Temporização de operação de leitura de estado[7]

```
Private Sub cmdRd_ST_Click()

    dataw1 = SPICS + RESET
    BitReset dataw1, 2 'chip select baixo
    DataPortWrite BaseAddress, dataw1
    WriteData &HA0 ‘instrução para leitura de estado em hexadecimal
    datar = ReadData ‘variável que armazenará bits dos registradores e poderá ser verificado
    ‘pela própria tela.
    txtCont_ST.Text = Hex$(datar)
    Dim bit_cont%
    Dim bitnumber2%
    For bitnumber2 = 0 To 7
        bit_cont = BitRead(datar, bitnumber2)
        txtBit_S(bitnumber2).Text = bit_cont
    Next
    BitSet dataw1, 2
    DataPortWrite BaseAddress, dataw1

End Sub
```

Rotina 7.9- Código fonte em VB para ação do botão para ler estado

7.6 TELAS DE CONFIGURAÇÃO E MONITORAMENTO

Para configuração do controlador CAN para recepção ou transmissão de dados foram construídas três telas, onde cada uma delas poderá manipular byte a byte ou mesmo, um conjunto ou bloco de bytes endereçados pelo identificador do frame. O Visual Basic oferece um conjunto poderoso e flexível de ferramentas para desenvolvimento do banco de dados. Por esta razão utilizou-se a associação do Visual Basic com o banco de dados Access, através dos objetos específicos da biblioteca do VB. O controle dbGrid é uma ferramenta de conexão entre as telas desenvolvidas (VB) e o banco de dados Access. Esta ferramenta incorpora cada campo dos bancos dados criados (*tx_can* e *rx_can*) a serem utilizados com as telas desenvolvidas. através das tabelas que são mostradas no capítulo 8 que mostram os resultados de recepção e transmissão dos frames.

7.6.1 Tela de Configuração Modo/Bit

A tela mostrada na Figura 7.30 possibilita a programação do formato e tempos de transmissão e recepção. Ele esta relacionada ao tempo de quantização (TQ) e ao formato do bit com base no tempo de quanta (TQ).

A configuração utilizada para sincronizar a interface desenvolvida com o nó monitorado (carro), incluindo o Baud Rate de 50Kbits/s e demais bits dos registradores de configuração do MCU-CAN: CNF1, CNF2 e CNF3 é apresentada na Figura 7.30.

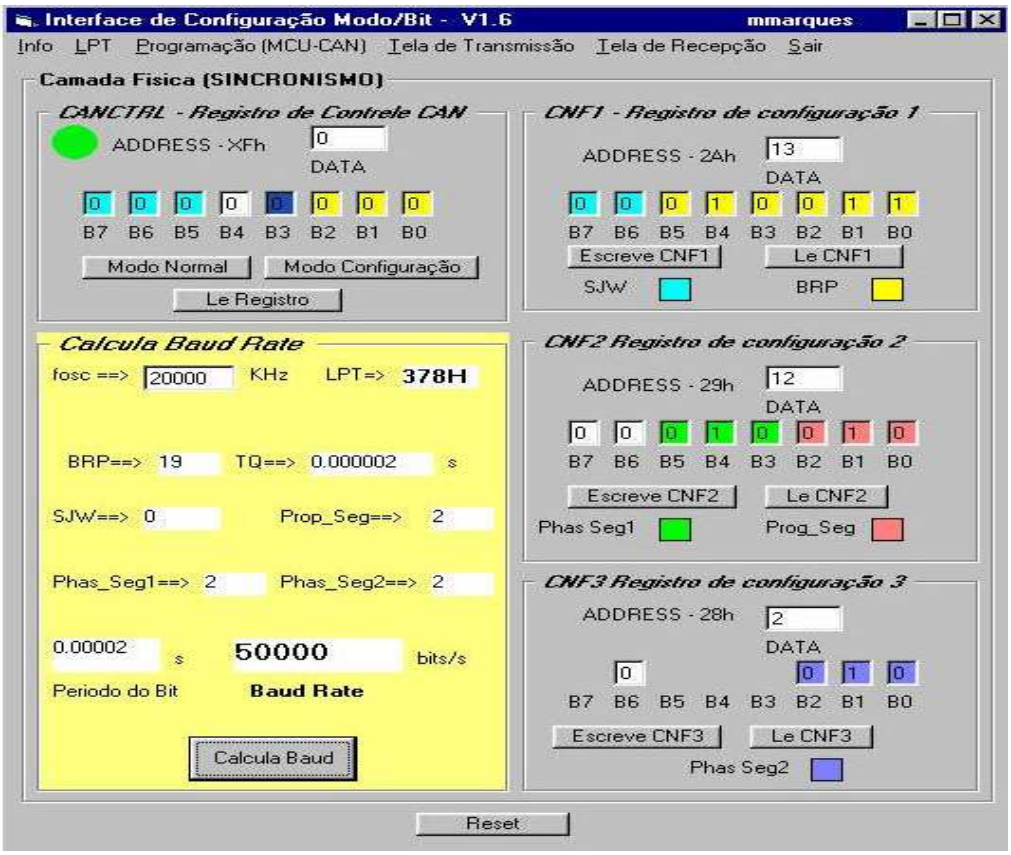


Figura 7.30-Tela de Configuração Modo/Bit

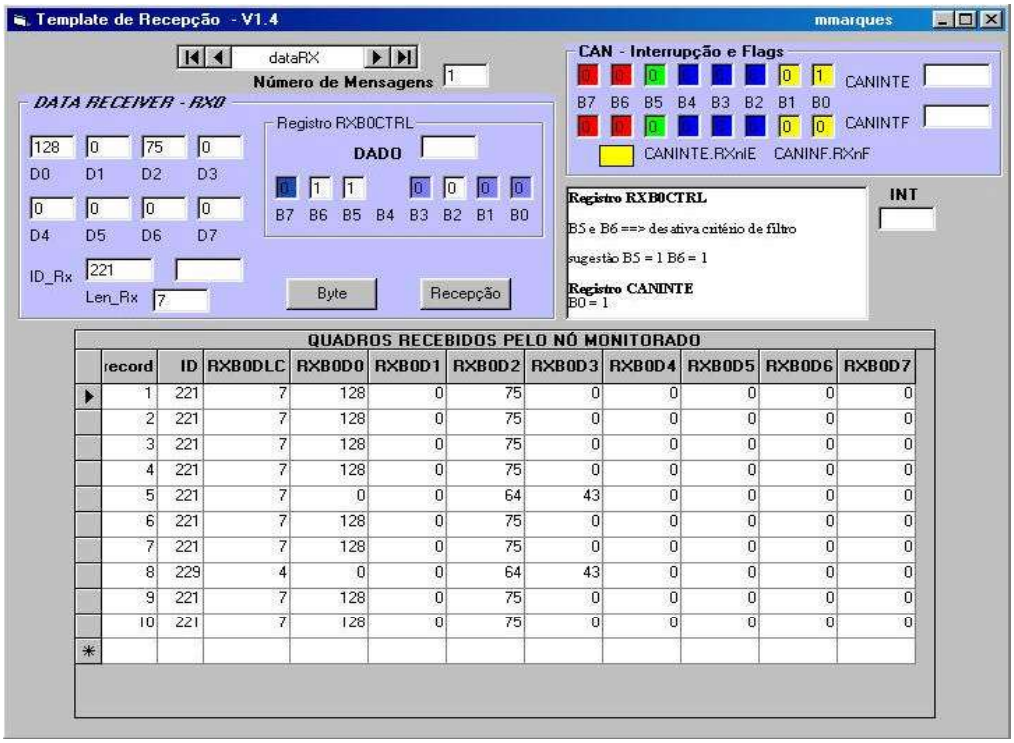
7.6.2 Tela de Recepção

A tela de recepção desenvolvido em Visual Basic (Figura 7.31) possibilita receber um frame de dados no Buffer de recepção RX0 e decompô-lo nos dados fundamentais deste frame. A área da tela mostra os valores dos bytes recebidos colocando-os no seu devido lugar facilitando assim a análise do conteúdo do frame e ainda associando este dado a uma posição de registro no banco de dados Access [19].

Para verificar um bloco de dados recebido e armazenado no arquivo ***"can_rx.mdb",*** utiliza-se o programa de banco de dados Access, podendo analisar e confrontar os dados recebidos a partir de relatórios. Com os dados transmitidos conhecidos a partir de um nó PC transmissor é portanto possível conferir a integridade da recepção dos dados em um nó PC receptor.

A seqüência da Rotina 7.10 resume a rotina, que representa a ação do botão "*Recebe_byte*" da Tela de Recepção e permite armazenar os dados recebidos nos

registros dos bytes baixo e alto do identificador padrão (RXBnSIDH e RXBnSIDL) e ainda os 8 bytes associados a este identificador.



record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
1	221	7	128	0	75	0	0	0	0	0
2	221	7	128	0	75	0	0	0	0	0
3	221	7	128	0	75	0	0	0	0	0
4	221	7	128	0	75	0	0	0	0	0
5	221	7	0	0	64	43	0	0	0	0
6	221	7	128	0	75	0	0	0	0	0
7	221	7	128	0	75	0	0	0	0	0
8	229	4	0	0	64	43	0	0	0	0
9	221	7	128	0	75	0	0	0	0	0
10	221	7	128	0	75	0	0	0	0	0

Figura 7.31-Tela de recepção

Rotina cmd_Rx_byte

Recepção de frames

Define as variáveis a serem utilizadas

Objeto associado ao Access

Variáveis de propósitos geral para apoio

Contador de mensagens

Definindo registro CANINTE – registro de habilitação de interrupção

A partir da tela, ajustam-se os estados dos bits do registro CANINTE

Converte o valor da forma binária para forma hexadecimal

Valor na tela

Definindo o CANINTF – registro de sinalização de interrupção.

A partir da tela de recepção, ajusta-se o estado dos bits do registro CANINTF

Registro de controle do buffer 0 de recepção – RXB0CTRL

Converte para decimal os estados de RXB0CTRL programados na tela

Lendo os registros de identificação - ID

Captura Rx0SIDL – padrão baixo

```
Definindo o datarxctrl
Liga os bits de RXM1 e RXM0 do registro RXB0CTRL, desligando a mascara dos filtros podendo
assim  receber qualquer dado
 Programando o registro CANINTE
 Programando o registro CANINTF
 Reset os bit CANINTF-RX0IF e CANINTF-RX1IF do registro CANINTF
 Lendo o SIDL
 Captura Rx0SIDH
  Ajusta os bits 0, 1 e 2 do registro SIDL
  Calcula o valor de TXB0SIDL (3 primeiros bits)
  Ajusta os bits de SIDH
  Chip select baixo
  Le o tamanho da informação em bytes
   Recebe D0
   Recebe D1
   Recebe D2
   Recebe D3
   Recebe D4
   Recebe D5
   Recebe D6
   Recebe D7
   Le rxb0ctrl
   Le caninte
   Le canintf
Fim
```

Rotina 7.10-Seqüência da rotina para monitoramento de um frame de um nó transmissor

7.6.3 Tela de Transmissão

O Tela de Transmissão assim como a de Recepção, foi desenvolvido em Visual Basic (Figura 7.32). Esta tela possibilita a transmissão de um frame de dados a partir do Buffer de transmissão TX0 e compor dados fundamentais para este frame. A área da tela mostra os valores dos bytes a serem transmitidos que também estão associados ao banco de dados Access. Portanto, pode-se transmitir um bloco de dados a partir do banco.

O código fonte da Rotina 7.11 representa a ação do botão "*Transmitir_byte*" da tela de transmissão e possibilita a transmissão de byte a byte do banco de dados do identificador padrão (TXBnSIDH e TXBnSIDL) e ainda os 8 bytes associados a este identificador.

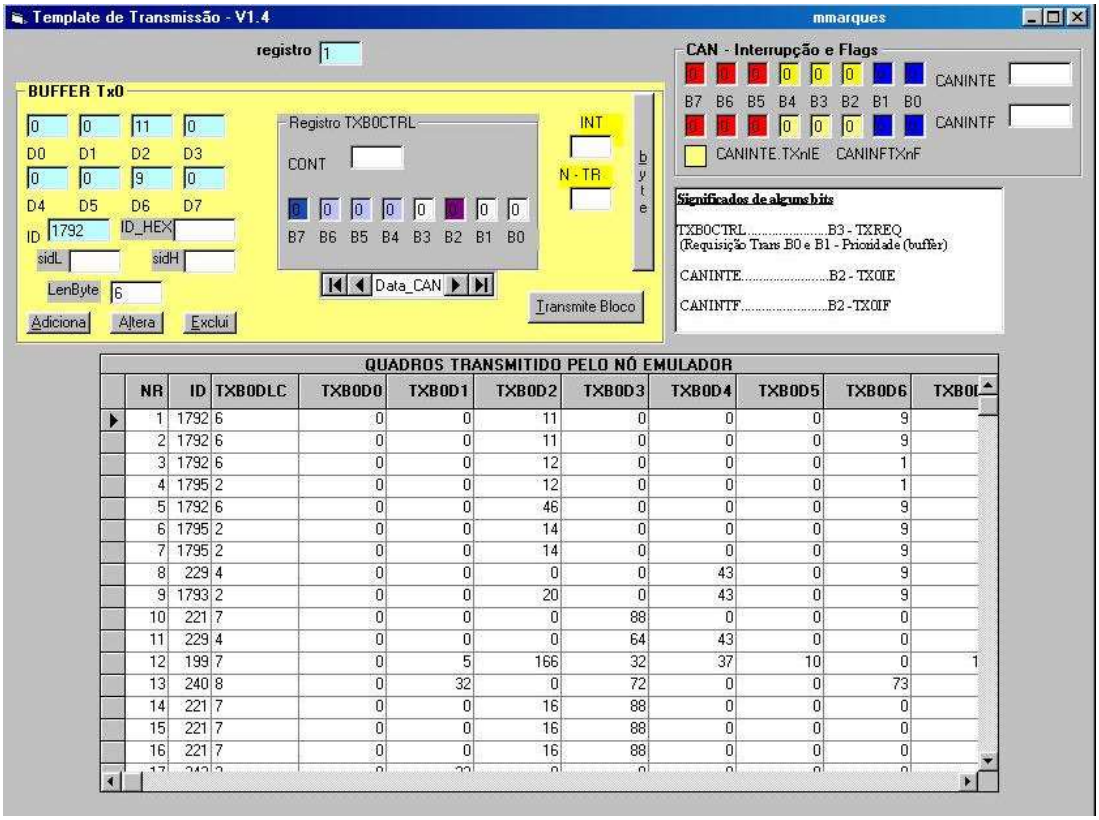


Figura 7.32-Tela de transmissão de dados desenvolvida

Nesta tela tem-se a possibilidade de acrescentar, apagar e modificar um registro qualquer do banco de dados. Para verificar e modificar o arquivo de dados recebidos, "***can_tx.mdb"***,utiliza-se o Access como meio de acesso rápido.

A seqüência da Rotina 7.11 resume a rotina que representa a ação do botão "*Transmite_byte*" da Tela de Transmissão e permite transmitir dados dos registros dos bytes baixo e alto do identificador padrão (TXBnSIDH e TXBnSIDL) e ainda os 8 bytes associados a este identificador.

Rotina cmd_Init_Click()

Transmissão dos frames

Variáveis auxiliares

Variável relacionada ao registro de controle do buffer de transmissão

Variável relacionada ao registro do identificador do frame

Objeto relacionado ao banco de dados

Loop para o próximo frame
Inicilaliza variável relacionada ao registro de controle do buffer de transmissão
Converte representação binária para hexadecimal
Armazena valor em datactrl
Conversão na tela
Calcula o valor para o registrador TXB0SIDL (3 primeiros bits do ID)
Ajusta os bits 0, 1 e 2 do registro SIDL
Transmite apenas modo standard
Converte os 3 bits correspondente X em hexadecimal para visualização
Ajusta os bits de SIDH – identificadores dos bits mais altos
Visualização na tela.
Atualiza MCU-CAN
CS baixo
Eceve sidL
CS alto
CS baixo
Escreve sidH
CS alto
INICIA A TRANSMISSÃO DOS BYTES DO QUADRO DE DADOS (DATA FRAME)
Escreve D0
Escreve D1
Escreve D2
Escreve D3
Escreve D4
Escreve D5
Escreve D6
Escreve D7
Define tamanho da mensagem
Transmite pelo buffer 0
Habilita buffer 0 a transmitir
Ajusta o estado do bit TXREQ para transmissão do buffer TX0
Transmite
Leitura do registro de controle de transmissão do buffer TX0
Mostra os valores dos estados do TXB0CTRL
fim

Rotina 7.11- Seqüência para a rotina de transmissão byte a byte

8 CAPITULO 8 -TESTES, LEVANTAMENTO DE DADOS E RESULTADOS

8.1 SIMULAÇÃO COM NÓ PC

Inicialmente, foi desenvolvido uma simulação com o nó PC para possibilitar vários ajustes e análises de sincronização entre os dois nós e o ajuste das taxas de transmissão e recepção.

Para tanto, foram levantados subsídios técnicos necessários para o monitoramento do sistema automotivo, possibilitando a análise do comportamento da camada física e da camada de enlace correspondentes aos sinais elétricos que trafegam pela rede. Este procedimento contribuiu na construção do chicote CAN, especifico para conexão entre dois conectores DB9, dando suporte fundamental na ligação com o veículo.

Para este teste preliminar do sistema de monitoramento foram utilizadas duas interfaces físicas (circuito eletrônico): uma necessária para simular o nó receptor (carro), que neste caso foi ligado em um PC com Pentium 133 e a outra no nó para monitoramento, que foi ligada a um PC com processador Celeron.

8.2 SIMULAÇÃO DA TRANSMISSÃO E RECEPÇÃO

Para entender o sincronismo e recuperação dos quadros (frames) de transmissão, cada interface física construída conectada a seu PC recebeu um cristal de valor diferente, sendo que o PC1 funcionou com cristal de 16MHz e PC2 com cristal de 20MHz, detalhados em Figura 8.1. Isto possibilitou se compreender a sincronização dos nós feita pelo próprio sinal, através da programação dos registros de configuração.

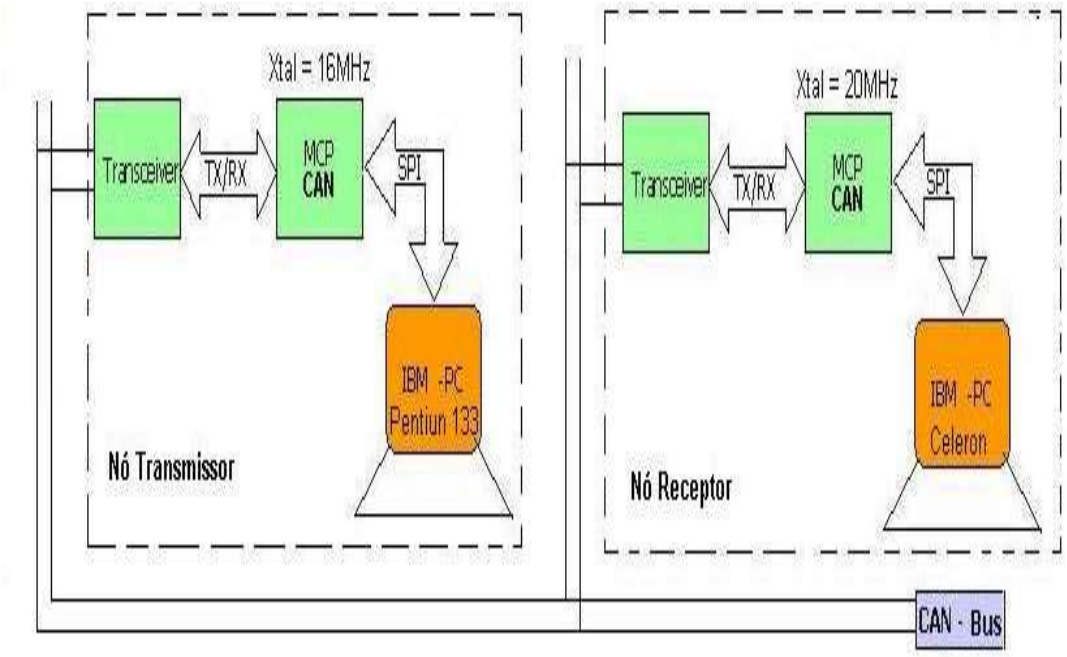


Figura 8.1 -Esquema para simulação da transmissão e recepção

A programação realizada no teste PC-PC, considerou uma taxa de transmissão de 500Kbits/s, ou seja, 500K baud rate (HS-CAN), de modo que cada computador ajusta-se ao valor de baud rate não importando o valor da freqüência do oscilador local de cada interface. Com base no que foi estudado no capítulo 4, tem-se então:

1- **Nó PC (transmissor) com cristal de 16MHz (Baud Rate = 500Kbits/s)**;

Para determinar o tempo de bit (Tbit), tem-se;

Tempo de Bit = 1 / Baude Rate (1) [7]

Tbit = 1 / 500.0000 = 2 μs

Tempo de bit em função do tempo de quanta TQ;

Tbit = TQ * ((SJW + 1) + (Prop_Seg + 1) + (Phas_Seg1 + 1) + (Phas_Seg2 + 1) (2) [7]

Os requisitos para a programação dos tempos dos segmentos:

Prop_Seg + Phas_Seg1 > = Phas_Seg2. [7]

Prop_Seg + Phas_Seg1 > = T_{ATRASO} (tempo de atraso físico da rede elétrica). [7]

Phas_Seg2 > SJW [7]

O tempo de atraso esta comumente entre 1 TQ e 2 TQ [7].

Assim sendo, tem-se:

SJW = 0 Prop Seg = 2 Phase Seg1 = 1 Phase Seg2 = 1 (valores ótimos estimados que obedecem aos requisitos sugeridos pelo fabricante).

Substituindo os dados acima na equação 2 e relacionado-a com a equação 1 resulta:

TQ = Tbit / ((SJW + 1) + (Prop_Seg + 1) + (Phas_Seg1 + 1) + (Phas_Seg2 + 1))

TQ = 2 μseg / (1 + 3 + 2 + 2) = 0,25 μs (3)

Bit(s) de controle da taxa de transmissão BRP;

TQ = 2 x (BRP + 1) / fosc (4) [7] ***(tempo de Quanta)***

Substituindo a fosc=16MHz, 3 em 4 resulta:

BRP = (TQ x Fosc / 2) – 1 ➔ **BRP = 1 (deve ser programado na camada física do PC transmissor -** Figura 8.2**)**

2- Nó PC (receptor) com cristal de 20MHz (Baud Rate = 500Kbits/s);

Para determinar o tempo de bit (Tbit), tem-se;

Tbit = 1 / Baude Rate = 1 / 500.0000 = 2 μs (1)

Tempo de bit em função do tempo de quanta TQ;

Tbit = TQ * ((SJW + 1) + (Prop_Seg + 1) + (Phas_Seg1 + 1) + (Phas_Seg2 + 1) (2)

SJW = 0 Prop Seg = 2 Phase Seg1 = 1 Phase Seg2 = 3 (valores ótimos estimados que obedecem aos requisitos sugeridos pelo fabricante). Substituindo os dados acima e relacionando 1 com 2 tem-se:

TQ = Tbit / ((SJW + 1) + (Prop_Seg + 1) + (Phas_Seg1 + 1) + (Phas_Seg2 + 1))

TQ = 2 μseg / (1 + 3 + 2 + 4) = 0,2 μs (3)

Bit(s) de controle da taxa de transmissão BRP;

TQ = 2 x (BRP + 1) / fosc (4)

Com o valor de fosc=20MHz e substituindo 3 em 4 resulta:

BRP = (TQ x Fosc / 2) – 1 ➔ **BRP = 1 (deverá ser programado na camada física do PC receptor** Figura 8.2**)**

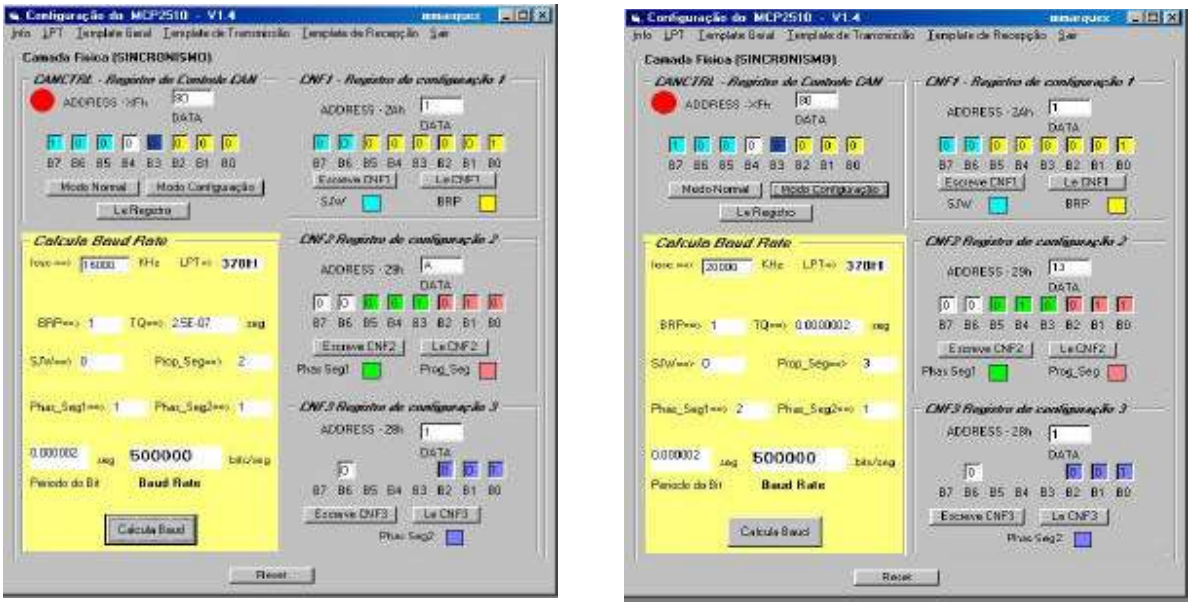


Figura 8.2 –Configuração da transmissão – clock = 16MHz e Configuração da Recepção – clock = 20MHz

PC TRANSMISSOR PC RECEPTOR

A partir dos cálculos realizados para o nó transmissor e para o nó receptor programam-se as telas de configuração no transmissor e no receptor de modo que a partir dos dados programados o programa calcule a taxa de transmissão, de acordo com a Figura 8.2.

As fotos da Figura 8.3 apresentam a estrutura de comunicação através do barramento CAN, simulando a comunicação entre o PC transmissor e o PC (lap top) receptor.

Figura 8.3-Conexão PC transmissor com PC receptor

A partir do arquivo teste armazenado no PC transmissor, ***can_tx,*** pôde-se enviar todos os quadros (frames) pertencentes ao arquivo várias vezes e recuperá-lo no nó receptor. Para

garantir que o dado a ser transmitido fosse carregado apenas uma única vez, garantindo assim um sincronismo da recepção com a transmissão, foram utilizados recursos de interrupção. Portanto o registrador de interrupção estabeleceu assim a sincronização física da recepção com a transmissão, ou seja,, utilizou-se a saída INT do controlador CAN ligada à entrada da porta paralela, (pino 10 AKN) referente ao registro de estado; (Figura 8.4). Com esta providência ficou garantida a recepção do arquivo frame a frame sem a inconveniência do recebimento de um frame repetido no mesmo arquivo que estaria sendo transmitido.

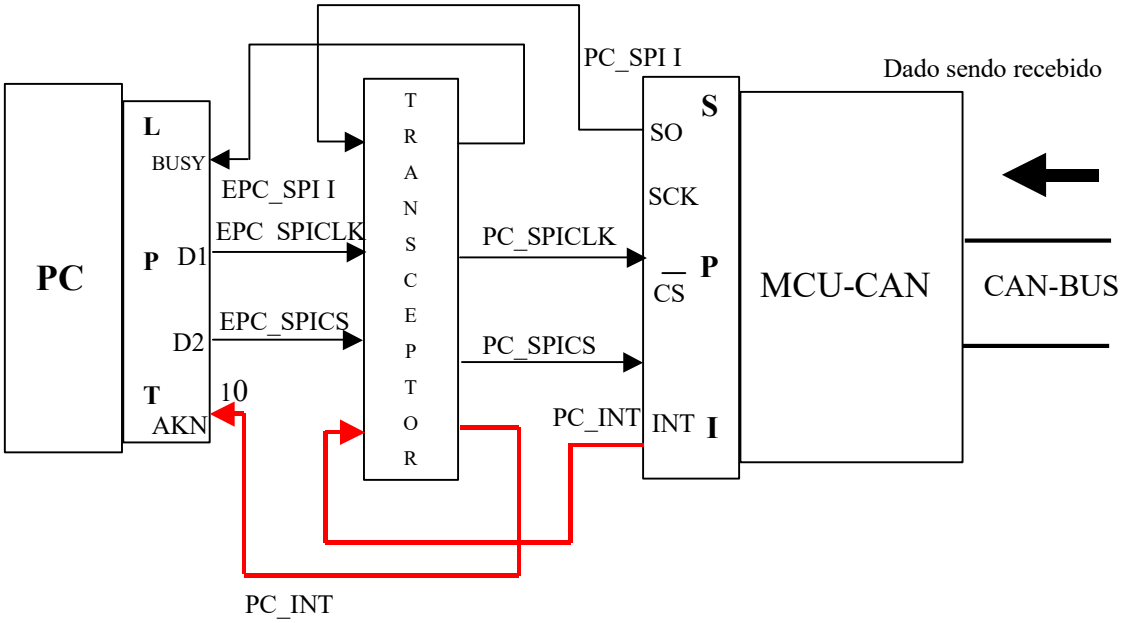


Figura 8.4-Handshaking físico na recepção - Trava

Para estabelecer o sincronismo lógico entre a transmissão e a recepção foi desenvolvido dentro do código fonte de recepção em VB, a Rotina 8.1 Esta rotina testa a trava física, que é baseada em interrupção por hardware. Na estrutura da trava a entrada D6 (pino 10 AKN) da porta de estado (H379) recebe o sinal de interrupção, vindo do MCU-CAN, que passa de nível lógico baixo para nível lógico alto toda vez que um frame é recebido.

Programação da Trava

Para implementar o mecanismo de trava descrito acima foi desenvolvido dentro do programa do nó receptor a rotina que fará com que o bit RxnIF do registrador, no caso foi

utilizado o buffer 0, CANINTF(CANINTT.RX0F:bit 0 do registrador) sinalize ao MCU-CAN o recebimento de um frame por parte do buffer n de recepção. Inicialmente ele é programado 1 lógico. Ao receber uma mensagem no buffer de recepção (RXB0) o bit RX0F do registrador de flag (CANINTF) é desligado (0 lógico). A partir deste ponto o filtro poderá ser testado, no caso da implementação não foi utilizado.

Se o bit RX0F não sinalizar a entrada de dado no buffer (buffer vazio e capaz de receber a mensagem) por motivo de estar ainda carregado o laço interno inicia um novo procedimento até que o buffer seja liberado.

Ao aceitar a mensagem o bit RX0F sinalizará 0 lógico. Após esta ocorrência será testado o bit RX0IE do registro que habilita a interrupção (CANINTT.RX0IE: bit 0 do registrador).

Na implementação o programa em VB irá desligar o bit RX0F e ligar o bit RX0IE a cada processo de recepção, conforme pseudo código da Figura 8.5.

Com a interrupção habilitada, o pino INT do MCU-CAN irá para o nível lógico 1 toda vez que uma mensagem for carregada no buffer 0 e para iniciar a nova carga.

Com este sinal físico ligado a uma entrada pelo registro de estado da LPT pode-se estabelecer a rotina descrita acima. A Figura 8.5 descreve a filosofia, estabelecida a partir da estrutura operacional de recepção do MCU-CAN.

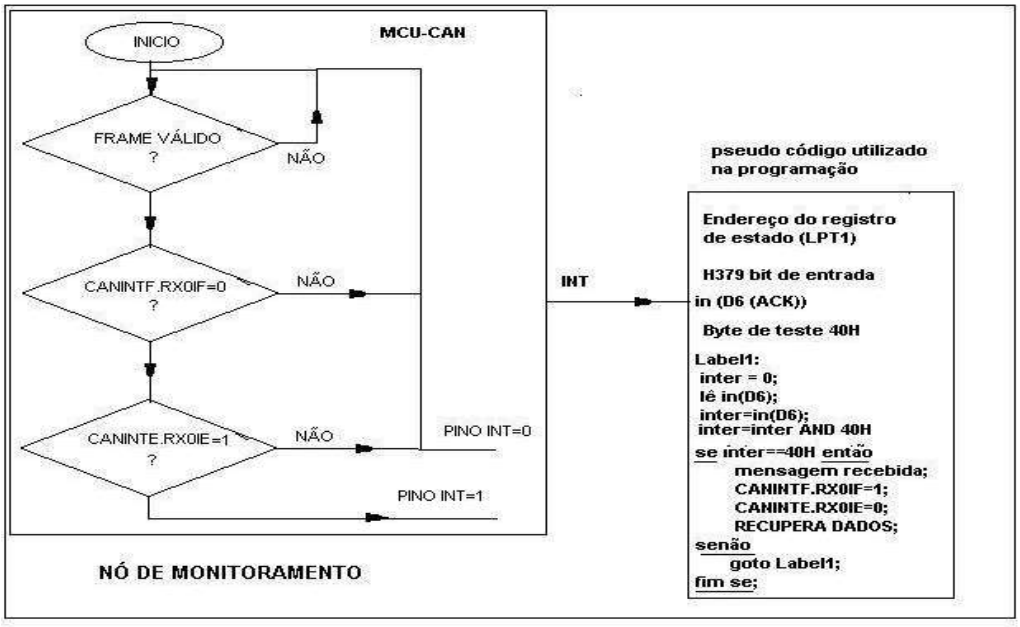


Figura 8.5-Estrutura da programação da trava

```
Private Sub cmd_Rx_byte_Click()
Dim bitnumber2%
Dim bit_cont%
Dim i%
Dim cont%
Dim datinRL%
'sinais que habilitam a SPI do MCP 2510
dataw1 = SPICS + RESET
prompt$ = "Entre com  o numero de frames a ser recebidos e gravados no banco de dados"
cont = Val(InputBox(prompt$, "Recepção"))
For i = 1 To cont
        Data_rx.Recordset.OpenRecordset
       'Define programação do registro RXB0CTRL - BUFFER 0
        Dim datarx0ctrl%
        Dim dataid%
        Dim bitnumber3%
        bit_cont = 0
        datarx0ctrl = 0
        For bitnumber3 = 7 To 0 Step -1
            bit_cont = txt_RCTRLb(bitnumber3).Text
            datarx0ctrl = (datarx0ctrl + bit_cont * 2 ^ bitnumber3)
        Next
        txt_RXB0CTRL(0).Text = Hex$(datarx0ctrl) & " h"
        BitReset dataw1, 2 'chip select baixo
        DataPortWrite BaseAddress, dataw1
        WriteData &H2
        WriteData &H60
        WriteData datarx0ctrl '&H60 desliga filtro e mascara recebe qquer men.
        BitSet dataw1, 2 'chip select alto
        DataPortWrite BaseAddress, dataw1
       'definindo o CANINTE
       'rotina usada para escrever no registrador CANINTE a partir da tela RX
        Dim datacaninte%
        datacaninte = 0
        For bitnumber2 = 7 To 0 Step -1
            bit_cont = txt_Bitint(bitnumber2).Text
            datacaninte = (datacaninte + bit_cont * 2 ^ bitnumber2)
```

```
Next
txt_CANINTE.Text = Hex$(datacaninte) & " h"
BitReset dataw1, 2 'chip select baixo
 DataPortWrite BaseAddress, dataw1
 WriteData &H2
 WriteData &H2B
 WriteData datacaninte
 BitSet dataw1, 2
 DataPortWrite BaseAddress, dataw1
 Dim inter%
‘Variável  inter que receberá dado da porta de estado da LPT, onde é verificado apenas a entrada
 AKN (bit D6).
 'Verificando se dado foi recebido pelo RX0
 'lendo o pino INT do 2510 através do pino 10 da porta de estado
 'Se 1 dado recebido se zero não
 Label:
 inter = 0
 inter = StatusPortRead(BaseAddress) 'porta de status D6
 inter = inter And &H40 ’LÓGICA AND ENTRE 0100 0000 B (40H)
 ‘Verifica-se o nível lógico do bit que esta sendo acesssado pela porta status
 txt_Status.Text = Hex(inter)
 If inter = &H40 Then
     'modificando bit - ligando o bit CANINF.RX0BIF=1
      timer_INT.Enabled = False
     'INICIALIZANDO O BUFFER0
     Dim mask%, datas%
     mask = 1
     datas = 1
     dataw1 = SPICS + RESET
     BitReset dataw1, 2 'chip select baixo
     DataPortWrite BaseAddress, dataw1
     WriteData &H5
     WriteData &H2C
     WriteData mask
     WriteData datas
     BitSet dataw1, 2
     DataPortWrite BaseAddress, dataw1
```

```
'contador de mensagens
txt_rec.Text = i
'le registro CANINTF
BitReset dataw1, 2
DataPortWrite BaseAddress, dataw1
WriteData &H3
WriteData &H2C
datin = ReadData
BitSet dataw1, 2
DataPortWrite BaseAddress, dataw1
txt_CANINTF.Text = Hex$(datin)
For bitnumber2 = 0 To 7
    bit_cont = BitRead(datin, bitnumber2)
    txt_BitintF(bitnumber2).Text = bit_cont
Next
'le caninte
BitReset dataw1, 2
DataPortWrite BaseAddress, dataw1
WriteData &H3
WriteData &H2B
datin = ReadData
BitSet dataw1, 2
DataPortWrite BaseAddress, dataw1
txt_CANINTE.Text = Hex$(datin)
For bitnumber2 = 0 To 7
    bit_cont = BitRead(datin, bitnumber2)
    txt_Bitint(bitnumber2).Text = bit_cont
Next
‘programa fonte que recupera ID e os dados do frame de D0 a D7
'le rxb0ctrl a partir da tela RX
BitReset dataw1, 2
DataPortWrite BaseAddress, dataw1
WriteData &H3
WriteData &H60
datin = ReadData
BitSet dataw1, 2
DataPortWrite BaseAddress, dataw1
```

```
            txt_RXB0CTRL(0).Text = (datin)
            For bitnumber2 = 0 To 7
                bit_cont = BitRead(datin, bitnumber2)
                txt_RCTRLb(bitnumber2).Text = bit_cont
            Next
            Data_rx.Recordset.AddNew
        Else
          GoTo Label1
        End If
Next
'desliga o bit RXBOINTE
BitReset dataw1, 2 'chip select baixo
DataPortWrite BaseAddress, dataw1
WriteData &H2
WriteData &H2B
WriteData 0
BitSet dataw1, 2
DataPortWrite BaseAddress, dataw1

End Sub
```

Rotina 8.1-Código fonte em VB da trava de recepção

Para o teste de simulação foi utilizado o arquivo *can_tx* de 10 frames para transmissão, Tabela 8.1, os quais foram transferidos do PC transmissor para o PC receptor. No modo transmissão, o arquivo de frames foi enviado várias vezes. O resultado obtido foi a recuperação dos frames do arquivo de transmissão ***can_tx*** no arquivo de recepção ***can_rx.*** A recuperação das informações (frames) pode ser vista conforme a Tabela 8.2.

NR	ID	TXB0DLC	TXB0D1	TXB0D2	TXB0D3	TXB0D4	TXB0D5	TXB0D6	TXB0D7
1	1792	6	0	11	0	0	0	9	0
2	1792	6	0	11	0	0	0	9	0
3	1792	6	0	12	0	0	0	1	0
4	1795	2	0	12	0	0	0	1	0
5	1792	6	0	46	0	0	0	9	0
6	1795	2	0	14	0	0	0	9	0
7	1795	2	0	14	0	0	0	9	0

NR	ID	TXB0DLC	TXB0D1	TXB0D2	TXB0D3	TXB0D4	TXB0D5	TXB0D6	TXB0D7
8	229	4	0	0	0	43	0	9	0
9	1793	2	0	20	0	43	0	9	0
10	221	7	0	0	88	0	0	0	0
11	229	4	0	0	64	43	0	0	0
12	199	7	5	166	32	37	10	0	115
13	240	8	32	0	72	0	0	73	13
14	221	7	0	16	88	0	0	0	0
15	221	7	0	16	88	0	0	0	0
16	221	7	0	16	88	0	0	0	0
17	243	3	32	0	0	0	0	0	0
18	221	7	0	16	88	0	0	0	0
19	221	7	0	16	88	0	0	0	0
20	229	4	0	16	88	0	0	0	0

Tabela 8.1-Tabela de transmissão exemplo

NR	ID	RXB0DLC	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
1	1792	6	0	11	0	0	0	9	0
2	1792	6	0	11	0	0	0	9	0
3	1792	6	0	12	0	0	0	1	0
4	1795	2	0	12	0	0	0	1	0
5	1792	6	0	46	0	0	0	9	0
6	1795	2	0	14	0	0	0	9	0
7	1795	2	0	14	0	0	0	9	0
8	229	4	0	0	0	43	0	9	0
9	1793	2	0	20	0	43	0	9	0
10	221	7	0	0	88	0	0	0	0
11	229	4	0	0	64	43	0	0	0
12	199	7	5	166	32	37	10	0	115
13	240	8	32	0	72	0	0	73	13
14	221	7	0	16	88	0	0	0	0
15	221	7	0	16	88	0	0	0	0
16	221	7	0	16	88	0	0	0	0
17	243	3	32	0	0	0	0	0	0
18	221	7	0	16	88	0	0	0	0
19	221	7	0	16	88	0	0	0	0
20	229	4	0	16	88	0	0	0	0

Tabela 8.2-Resultado após processo de recepção

O número de informações recebidas é programado no registro TX0DLC, do processador CAN. Isto é importante para composição do arquivo de recepção. No teste realizado, o processo manteve a integridade dos bytes transmitidos.

8.3 MONITORAMENTO DO SISTEMA AUTOMOTIVO

A partir dos resultados obtidos na simulação passou-se para o teste final com sistema automotivo como transmissor de dados pertinentes a vários módulos de segurança, conforto e motor.

Para tanto, foi utilizado o automóvel da montadora Fiat de modelo Palio Weekend Fire com sistema Ve.N.I.C.E, com diagrama de blocos visto na Figura 8.6.

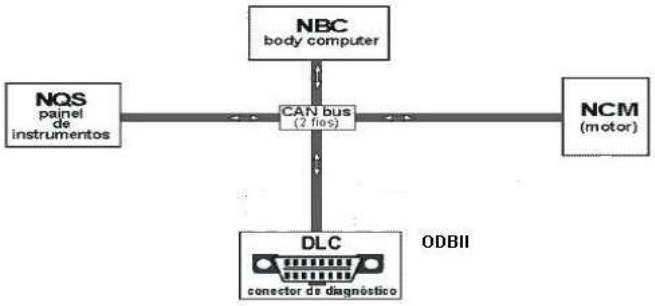


Figura 8.6-Diagrama de blocos do sistema de eletrônica embarcada para o sistema Fire[5]

O BC (body computer) gerencia as informações trocadas entre as várias centrais eletrônicas através da rede CAN. O Palio 1.3 16V Fire, assim que foi lançado em 2000, sem o sistema Ve.N.I.C.E., possuía um sensor de temperatura do líquido de arrefecimento – CTS com 4 fios: dois fios indicavam o sinal a UCE e dois fios informavam ao painel de instrumentos, Figura 8.7 .

Com a introdução do sistema Ve.N.I.C.E., observa-se que o mesmo sensor passou a ter somente 2 fios que são ligados diretamente a UCE. Nos modelos atuais, a temperatura de arrefecimento é informada ao painel de instrumentos através da rede CAN pelo BYTE D2 do frame de ID 221D, detalhe na Figura 8.8.

- A UCE lê o valor da temperatura da água informada pelo sensor CTS;
- Através da rede CAN o sinal é repassado ao NBC;
- O BC repassa agora esta informação para o painel de instrumentos (também através da rede CAN).

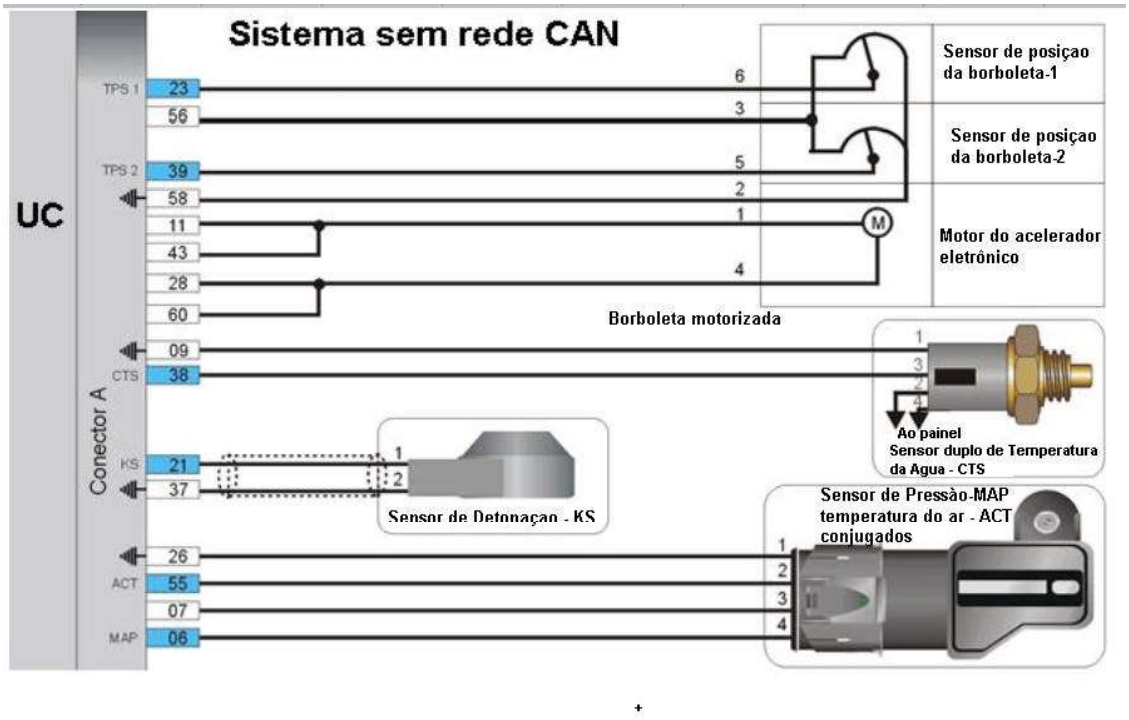


Figura 8.7 – Localização do sensor de temperatura no sistema rede CAN[14]

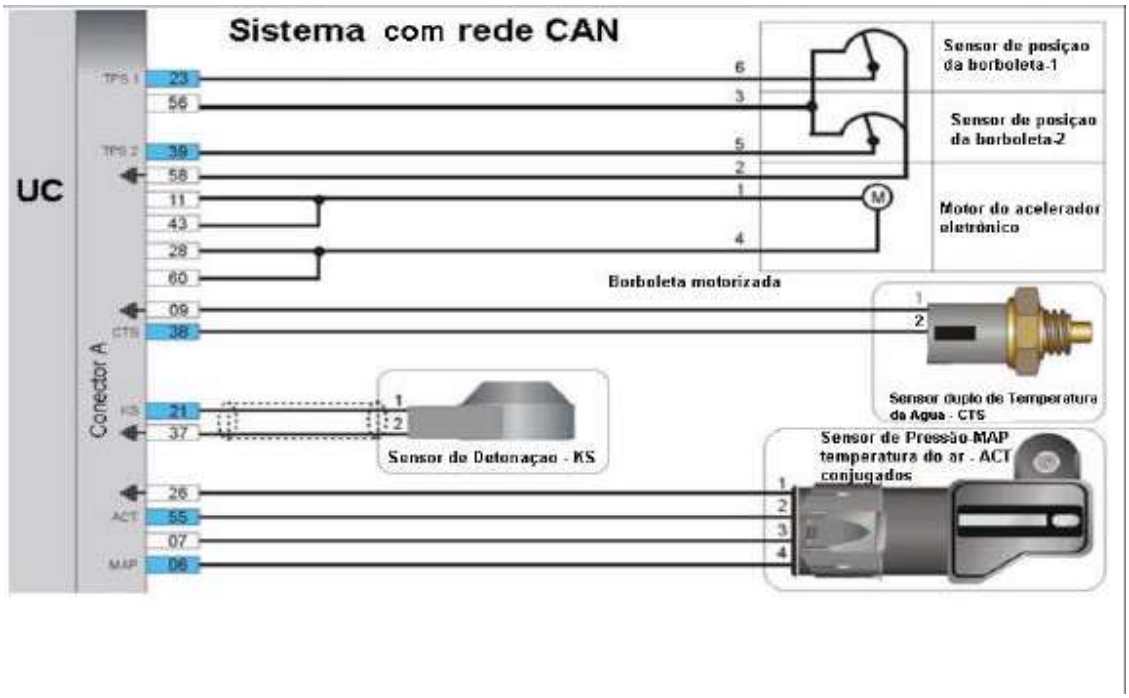


Figura 8.8 – Localização do sensor de temperatura no sistema com rede CAN[14]

De maneira similar (através do NBC e da rede CAN) são informadas a rotação do motor, nível de combustível, sensoriamento de portas, lâmpadas, etc). Desta forma, conseguiu-se diminuir de maneira significativa os fios, sensores e conectores redundantes no sistema. A Figura 8.9 mostra o diagrama elétrico do sistema Bosch Motronic, utilizados nos carros da FIAT a partir de 2001 com vários sensores e atuadores. A Figura 8.10 mostra o detalhe da rede CAN no sistema.

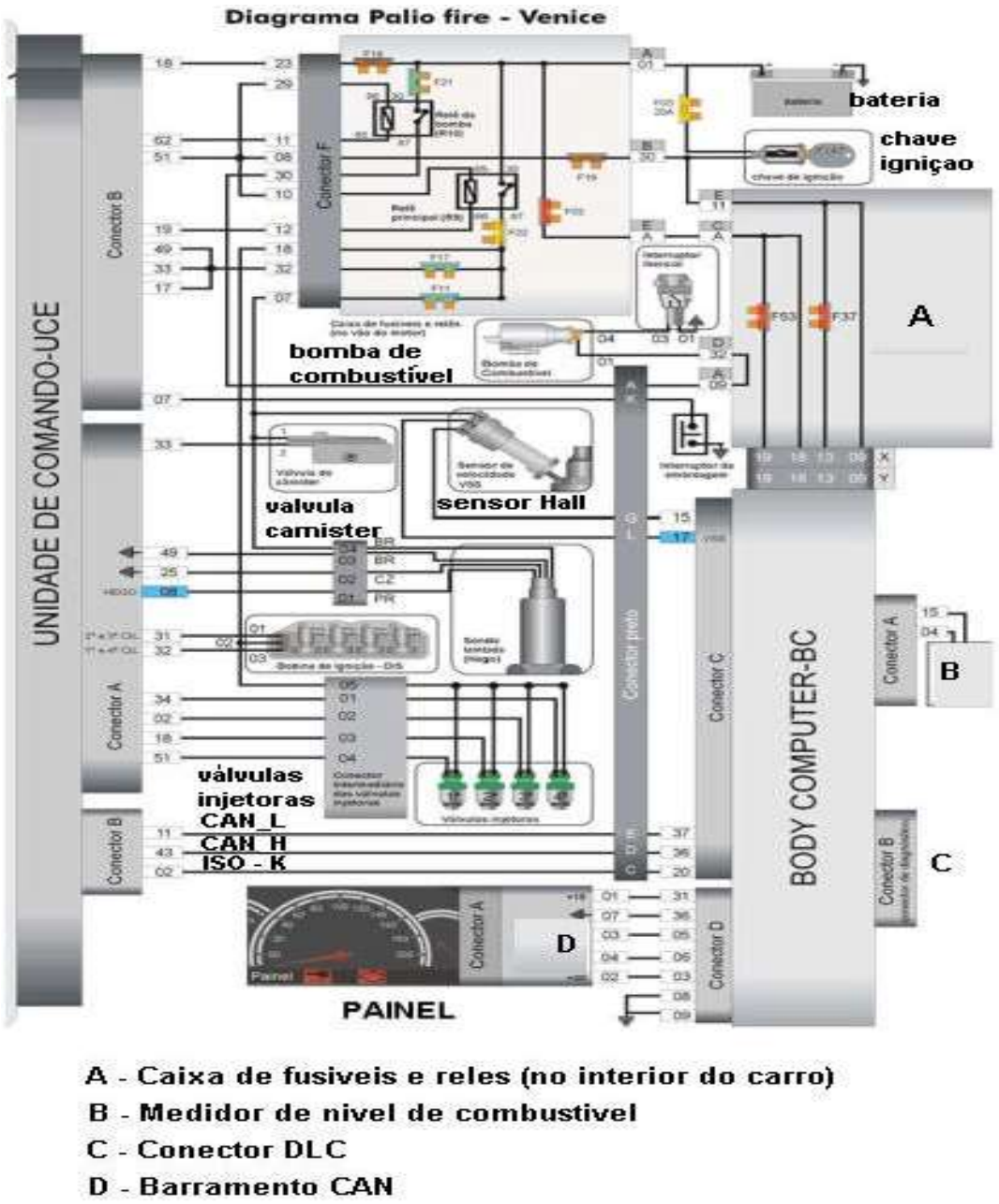


Figura 8.9 - Diagrama elétrico do sistema Bosch Motronic – Com sistema Ve.N.I.C.E. [14]

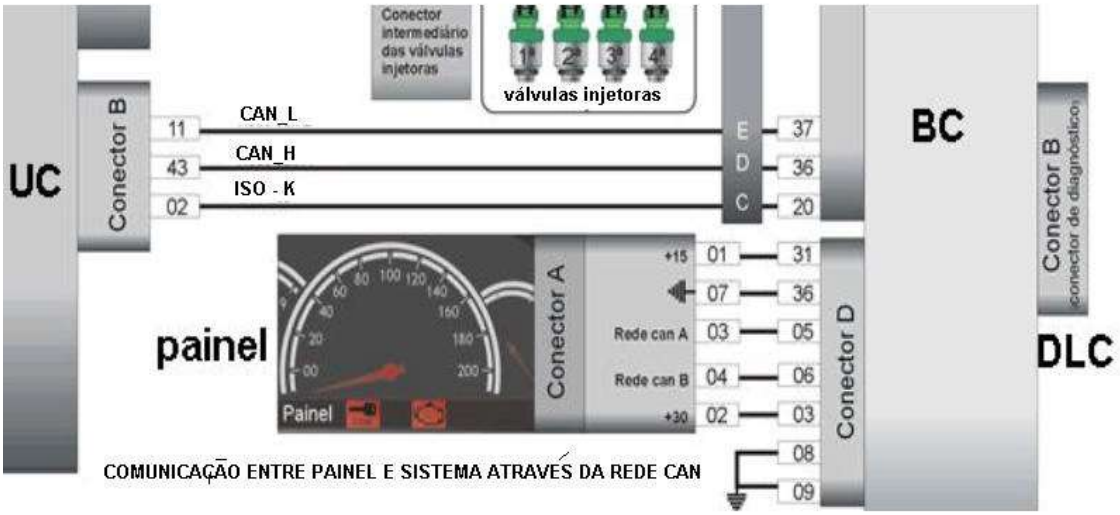


Figura 8.10 -Detalhe do barramento CAN entre o UC-BC-painel[14]

8.3.1 Localização de Conector OBDII e Identificação do Sinal de Rede

Datas do Monitoramento: 30/06/03, 07/07/03, 08/07/03 e 09/07/03

Primeiro contato com o carro da Fiat que utiliza rede CAN Bus para comunicação do computador de bordo (BC) com painel de instrumentos (QS) e motor (CM).

a) Identificação do conector DLC (J9262) no Palio Weekend, Figura 8.11e Figura 8.12. Utilizou-se o informativo Tabela de Aplicação Rasther MPI (Tecnomotor) para encontrar a provável localização do conector DLC.

Figura 8.11-Visão interna do Palio Weekend

Figura 8.12-Detalhe do painel e localização do OBDII

b) Conexão do chicote CAN entre conector DLC (OBDII) e a interface física (Figura 8.13 e Figura 8.14).

Figura 8.13-Ligação chicote CAN

Figura 8.14-Detalhe geral da bancada de Teste

O nó de monitoramento foi composto de um PC e da interface física desenvolvida, conforme apresentado na Figura 8:13 e conectada ao carro pelo chicote CAN (Figura 8.15).

Figura 8.15-Conexão bancada e carro

Figura 8.16-Elementos da bancada

Para confirmação da conexão entre o carro e a interface VisualCAN utilizou-se uma bancada de teste composta por um osciloscópio fixado na entrada da interface visual, para facilitar a identificação de sinal de rede, conforme visto na Figura 8.17.

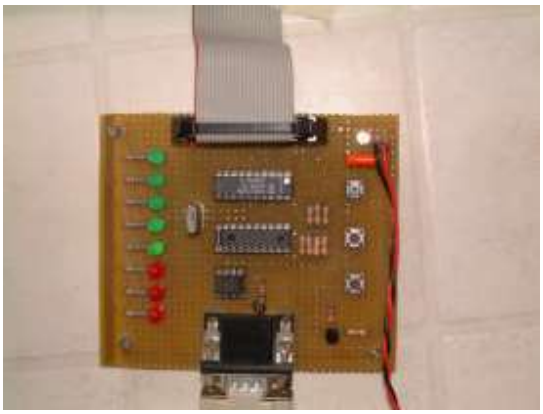


Figura 8.17-Nó de monitoramento- osciloscópio

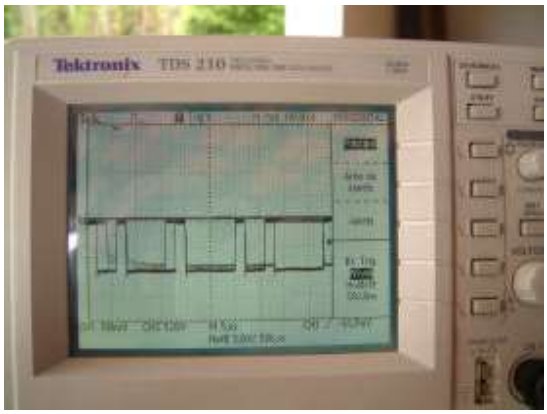


Figura 8.18-Oscilograma no barramento CAN

O ajuste de sincronismo e a inspeção para a taxa de transmissão do carro poder ser descoberta, foram feitos pela análise do sinal do período do sinal no oscilograma, como visto na Figura 8:13, no momento da transmissão de dados pelo sistema automotivo. A taxa estimada foi de 50000 bit/s.

O ajuste de sincronismo e a inspeção para a taxa de transmissão do carro poder ser descoberta, foram feitos pela análise do sinal do período do sinal no oscilograma, como visto na Figura 8:13, no momento da transmissão de dados pelo sistema automotivo. A taxa estimada foi de 50000 bit/s.

8.3.2 Programando a Comunicação com o Carro

Para a programação da comunicação com o carro, fez-se o ajuste dos registradores de configuração do MCU-CAN (MCP-2510) através dos registradores CNF1, CNF2 e CNF3, com auxílio da tela de Configuração Modo/Bit. Repetem-se os mesmos procedimentos realizados na emulação a partir do Baud Rate descoberto e estabelecido. Deste modo, o nó PC de monitoramento fica com o crystal de 20MHz (fosc=20MHz).

Tempo de Bit = 1 / Baude Rate = 1 / 50.000 = 200 μs (1)

SJW = 0 Prop Seg = 2 Phase Seg = 2 Phase Seg2 = 2 (valores estimados)

Tbit = TQ * ((SJW + 1) + (Prop_Seg + 1) + (Phas_Seg1 + 1) + (Phas_Seg2 + 1)

TQ = Tbit / ((SJW + 1) + (Prop_Seg + 1) + (Phas_Seg1 + 1) + (Phas_Seg2 + 1)) (2)

Substituindo 1 em 2 e 3 em 4 resulta:

TQ = 200 μs / (1 + 3 + 3 + 3) = 20 μs (3)

TQ = 2 x (BRP + 1) / fosc (4) [7]

BRP = (TQ x fosc / 2) – 1 ➔ **BRP = 19 (Deverá ser ajustado na camada física)**

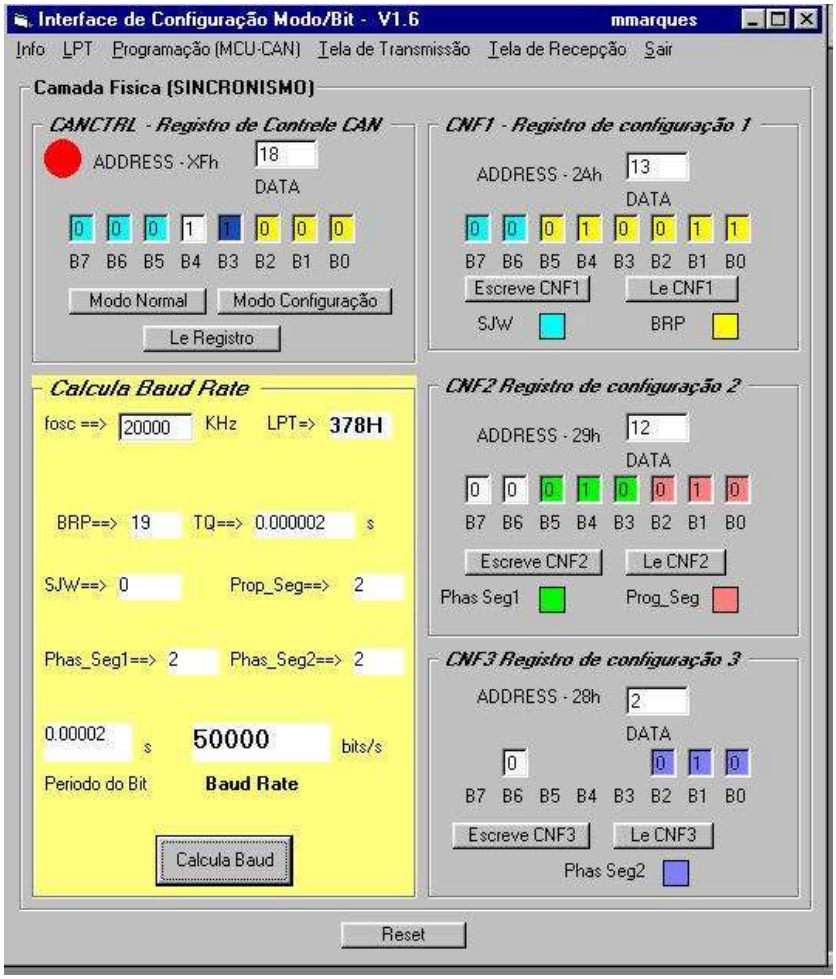


Figura 8.19-Tela de configuração ajustada para os valores estimados e calculados (Palio)

A comunicação, então, foi iniciada com a configuração do BIT TIME, estabelecida a partir de dados programados mostrados na Figura 8.13. Foram observados que o não reconhecimento de recepção faz com que o nó de monitoramento implementado pelo PC gere um frame de erro que é enviado ao nó monitorado (carro). Isto confunde o seu funcionamento e impede o funcionamento do carro.

A partir do sincronismo correto a tela de monitoramento é capaz de receber os dados e ativar as interrupções necessárias.

O monitoramento de várias ocorrências geraram dados que foram armazenados no arquivo dataRX. Estes dados foram analisados e comparados com seus dados antes da

ocorrência através do Banco de Dados Access. Arquivos estes analisados antes e depois de provocar as diversas ocorrências. Dados foram capturados pela tela de recepção do VisualCAN, Figura 8.17.

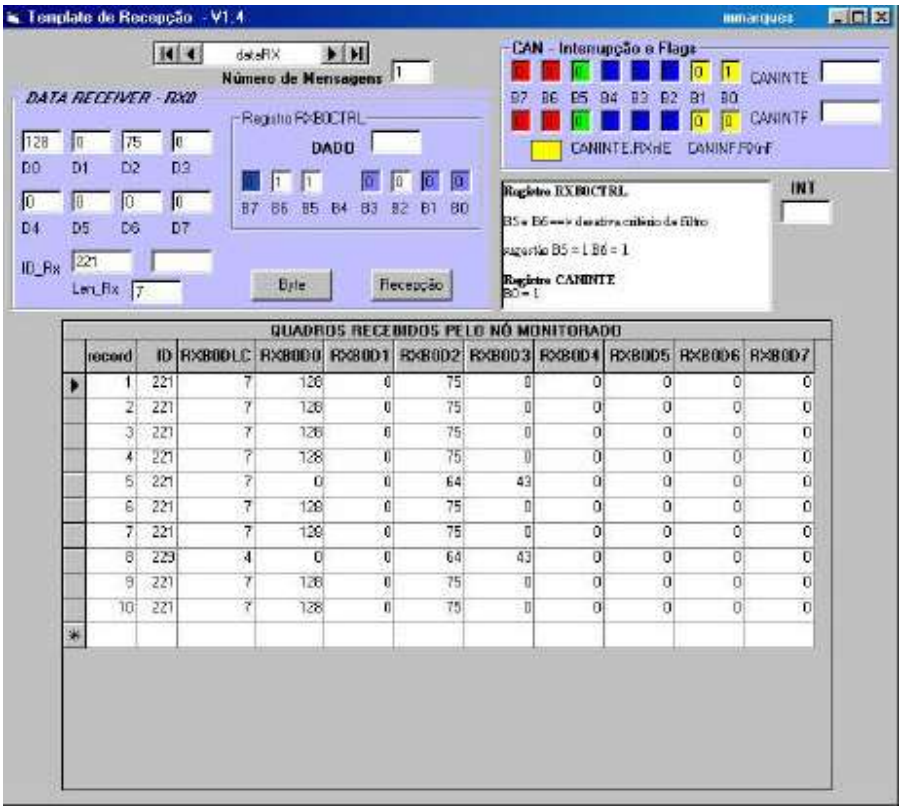


Figura 8.20-Tela de recepção

A partir dos arquivos gerados e analisados foi possível monitorar as várias ações de porta, luzes e motor. Obviamente as análises foram investigativas e baseadas em pura análise dos arquivos gerados uma vez que o fabricante não disponibiliza nenhum tipo de informação.

A partir de vários testes de monitoramento consegue-se identificar vários ID(s) e os seus respectivos sensores ligados a eles. Sendo assim, a partir disto, chegou-se à conclusão relacionada abaixo com relação ao sistema automotivo Palio Weekend .

8.3.2.1 Grupo de Sensores e Atuadores do Palio Weekend

Para localizar o identificador que endereçam as portas do carro prepara-se a tela de recepção e inicia-se a recepção em torno de 2000 frames com o carro apenas com a chave

de ligada sem motor. Durante o processo de recepção, abriu-se e se fechou a porta duas vezes. Após o término da recepção, com ajuda do Access, abriu-se o banco de dados gerado no processo anterior comparando vários ID(s) e se constatam mudanças nos dados identificados como ID da porta.

8.4 RESULTADOS PARA SISTEMA AUTOMOTIVO I

Após vários testes similares ao descrito acima chega - se aos resultados relacionados:

GRUPO PORTAS

ID = 240D.....BYTE = D1 (B7 B6 **B5 B4 B3 B2** B1 B0)

B2 ➔ porta da frente/ lado motorista exemplo: D1 = 4

B3 ➔ porta da frente/ carona exemplo: D1 = 8

B4 ➔ porta atrás/ lado motorista exemplo: D1 = 16D

B5 ➔ porta atrás/ lado carona exemplo: D1 = 32D

Exemplo: Portas frontais abertas D1= 12D

A Tabela 8.3 foi o resultado do monitoramento do grupo portas, onde a partir da ocorrência de todas as portas fechadas (D1=0), a porta do motorista foi aberta e fechada novamente. A seqüência foi repetida para a porta atrás motorista, porta do carona e finalmente porta atrás do carona.

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
4	240	8	32	0	72	0	0	14	13	0
5	240	7	128	0	113	0	0	0	0	0
25	240	8	32	0	72	0	0	14	13	0
34	240	8	32	0	72	0	0	14	13	0
45	240	8	32	0	72	0	0	14	13	0
66	240	8	32	0	72	0	0	14	13	0
106	240	8	32	0	72	0	0	14	13	0
107	240	8	32	0	72	0	0	0	0	0
125	240	8	32	0	72	0	0	14	13	0
146	240	8	32	4	72	0	0	14	13	0
155	240	8	32	4	72	0	0	14	13	0
166	240	8	32	4	72	0	0	14	13	0

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
176	240	8	32	4	72	0	0	14	13	0
223	240	8	32	4	72	0	0	14	13	0
224	240	8	32	4	72	0	0	0	0	0
241	240	8	32	4	72	0	0	14	13	0
280	240	8	32	4	72	0	0	14	13	0
299	240	8	32	4	72	0	0	14	13	0
318	240	8	32	0	72	0	0	14	13	0
328	240	8	32	0	72	0	0	14	13	0
338	240	8	32	0	72	0	0	14	13	0
357	240	8	32	0	72	0	0	14	13	0
377	240	8	32	0	72	0	0	14	13	0
397	240	8	32	0	72	0	0	14	13	0
416	240	8	32	0	72	0	0	14	13	0
436	240	8	32	0	72	0	0	14	13	0
437	240	8	32	0	113	0	0	0	0	0
456	240	8	32	16	72	0	0	14	13	0
466	240	8	32	16	72	0	0	14	13	0
476	240	8	32	16	72	0	0	14	13	0
496	240	8	32	16	72	0	0	14	13	0
506	240	8	32	16	72	0	0	14	13	0
515	240	8	32	16	72	0	0	14	13	0
535	240	8	32	16	72	0	0	14	13	0
536	240	8	128	0	113	0	0	0	0	0
554	240	8	32	16	72	0	0	14	13	0
574	240	8	32	16	72	0	0	14	13	0
584	240	8	32	0	72	0	0	14	13	0
593	240	8	32	0	72	0	0	14	13	0
613	240	8	32	0	72	0	0	14	13	0
631	240	8	32	0	72	0	0	14	13	0
651	240	8	32	0	72	0	0	14	13	0
652	240	8	32	0	113	0	0	0	0	0
670	240	8	32	0	72	0	0	14	13	0
690	240	8	32	0	72	0	0	14	13	0
709	240	8	32	0	72	0	0	14	13	0
749	240	8	32	8	72	0	0	14	13	0
750	240	8	32	8	113	0	0	0	0	0
759	240	8	32	8	72	0	0	14	13	0

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
770	240	8	32	8	72	0	0	14	13	0
779	240	8	32	8	72	0	0	14	13	0
790	240	8	32	8	72	0	0	14	13	0
800	240	8	32	8	72	0	0	14	13	0
811	240	8	32	8	72	0	0	14	13	0
831	240	8	32	8	72	0	0	14	13	0
851	240	8	32	8	72	0	0	14	13	0
871	240	8	32	0	72	0	0	14	13	0
872	240	8	32	0	113	0	0	0	0	0
892	240	8	32	0	72	0	0	14	13	0
901	240	8	32	0	72	0	0	14	13	0
912	240	8	32	0	72	0	0	14	13	0
933	240	8	32	0	72	0	0	14	13	0
972	240	8	32	0	72	0	0	14	13	0
990	240	8	32	0	72	0	0	14	13	0
991	240	8	32	0	113	0	0	0	0	0
1010	240	8	32	32	72	0	0	14	13	0
1029	240	8	32	32	72	0	0	14	13	0
1048	240	8	32	32	72	0	0	14	13	0
1066	240	8	32	32	72	0	0	14	13	0
1086	240	8	32	32	72	0	0	14	13	0
1104	240	8	32	32	72	0	0	14	13	0
1124	240	8	32	32	72	0	0	14	13	0
1143	240	8	32	0	72	0	0	14	13	0
1162	240	8	32	0	72	0	0	14	13	0
1182	240	8	32	0	72	0	0	14	13	0
1183	240	8	128	0	113	0	0	0	0	0
1192	240	8	32	0	72	0	0	14	13	0
1202	240	8	32	0	72	0	0	14	13	0
1212	240	8	32	0	72	0	0	14	13	0
1222	240	8	32	0	72	0	0	14	13	0
1232	240	8	32	0	72	0	0	14	13	0
1241	240	8	32	0	72	0	0	14	13	0

Tabela 8.3-Monitoramento do grupo portas

GRUPO LUZES

Neste grupo o teste investigativo foi similar ao feito com a porta - neste caso dois bytes pertencentes ao mesmo identificador (ID) - e que levou a seguinte conclusão:

ID = 207D.....BYTES = D1 e D2

D1 = 96D = 60H.....Lanterna

D1 = 104D= 68H.....Lanterna + Farol baixo ➔ D1= 8.....Farol baixo

D1 = 120D = 78H.....Lanterna + Farol baixo + Farol alto ➔ D1 = 16D = 10H

D1 = 16D = 10H.....Pisca farol alto (confirmando)

D2 = 64D = 40H.....Seta esquerda

D2 = 32D = 20H.....Seta direita

D2 = 96D.= 60H.....Luzes de alerta

A Tabela 8.4 mostra as mudanças no ID 207 para a seqüência de ocorrência provocada seguindo:

-luzes desligadas – lanterna – lanterna + farol baixo – lanterna + farol baixo + farol alto – lanterna – luzes desligas – seta esquerda – luzes desligadas – seta direita – pisca farol alto- pisca alerta.

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
53	207	6	0	0	0	0	0	0	0	0
61	207	6	0	0	0	0	0	0	0	0
70	207	6	0	0	0	0	0	0	0	0
113	207	6	0	0	0	0	0	0	0	0
149	207	6	32	0	72	0	0	14	13	0
182	207	6	0	0	0	0	0	0	0	0
192	207	6	0	96	0	0	0	0	0	0
253	207	6	0	96	0	0	0	0	0	0
288	207	6	0	96	0	0	0	0	0	0
331	207	6	0	96	0	0	0	0	0	0
494	207	6	0	104	0	0	0	0	0	0
511	207	6	0	104	0	0	0	0	0	0
571	207	6	0	104	0	0	0	0	0	0
649	207	6	0	104	0	0	0	0	0	0
658	207	6	0	104	0	0	0	0	0	0
666	207	6	0	104	0	0	0	0	0	0

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
684	207	6	0	120	0	0	0	0	0	0
727	207	6	0	120	0	0	0	0	0	0
872	207	6	0	120	0	0	0	0	0	0
898	207	6	0	120	0	0	0	0	0	0
941	207	6	0	120	0	0	0	0	0	0
967	207	6	0	120	0	0	0	0	0	0
976	207	6	0	120	0	0	0	0	0	0
992	207	6	0	120	0	0	0	0	0	0
1025	207	6	0	120	0	0	0	0	0	0
1033	207	6	0	120	0	0	0	0	0	0
1093	207	6	0	96	0	0	0	0	0	0
1094	207	6	0	0	0	0	0	0	0	0
1113	207	6	0	0	0	0	0	0	0	0
1130	207	6	32	0	72	0	0	14	13	0
1139	207	6	0	0	0	0	0	0	0	0
1156	207	6	0	16	99	0	0	0	0	0
1182	207	6	0	0	72	0	0	14	13	0
1199	207	6	0	16	0	0	0	0	0	0
1277	207	6	0	16	0	0	0	0	0	0
1282	207	6	0	0	0	0	0	0	0	0
1320	207	6	0	0	99	0	0	0	0	0
1353	207	6	0	0	0	0	0	0	0	0
1396	207	6	0	0	0	0	0	0	0	0
1421	207	6	128	0	99	0	0	0	0	0
1438	207	6	0	0	0	0	0	0	0	0
1456	207	6	0	0	0	0	0	0	0	0
1533	207	6	0	0	64	0	0	0	0	0
1547	207	6	0	0	64	0	0	0	0	0
1560	207	6	0	0	0	0	0	0	13	0
1583	207	6	0	0	0	0	0	0	0	0
1584	207	6	0	0	0	0	0	0	0	0
1596	207	6	0	0	64	0	0	0	0	0
1603	207	6	0	0	64	0	0	0	0	0
1627	207	6	0	0	0	0	0	0	0	0
1654	207	6	0	0	0	0	0	0	0	0
1663	207	6	0	0	72	0	0	14	13	0
1678	207	6	0	0	0	0	0	0	0	0

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
1715	207	6	0	0	32	0	0	0	0	0
1716	207	6	0	0	32	0	0	0	0	0
1727	207	6	0	0	0	0	0	0	0	0
1739	207	6	0	0	32	0	0	0	0	0
1751	207	6	0	0	0	0	0	0	0	0
1762	207	6	0	0	32	0	0	0	0	0
1817	207	6	0	0	32	0	0	0	0	0
1832	207	6	0	0	32	0	0	0	0	0
1833	207	6	0	0	32	0	0	0	0	0
1855	207	6	0	0	32	0	0	0	0	0
1867	207	6	0	0	0	0	0	0	0	0
1879	207	6	0	0	32	0	0	0	0	0
1890	207	6	0	0	0	0	0	0	0	0
1891	207	6	0	0	0	0	0	0	0	0
1902	207	6	0	0	0	0	0	0	0	0
1937	207	6	0	0	0	0	0	0	0	0
1954	207	6	0	0	0	0	0	0	0	0
1963	207	6	0	0	0	0	0	0	0	0
2023	207	6	0	0	0	0	0	0	0	0
2039	207	6	0	0	0	0	0	0	0	0
2073	207	6	128	0	99	0	0	0	0	0
2090	207	6	0	0	0	0	0	0	0	0
2141	207	6	0	0	72	0	0	14	13	0
2158	207	6	32	0	72	0	0	14	13	0
2184	207	6	0	0	0	0	0	0	0	0
2201	207	6	0	0	0	0	0	0	0	0
2261	207	6	0	0	0	0	0	0	0	0
2279	207	6	0	0	0	0	0	0	0	0
2288	207	6	0	0	0	0	0	0	0	0
2357	207	6	0	0	0	0	0	0	0	0
2391	207	6	0	0	0	0	0	0	0	0
2399	207	6	0	0	0	0	0	0	0	0
2408	207	6	0	0	0	0	0	0	0	0
2417	207	6	0	0	0	0	0	0	0	0
2433	207	6	0	0	0	0	0	0	0	0
2467	207	6	0	0	0	0	0	0	0	0
2640	207	6	0	0	99	0	0	0	0	0

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
2665	207	6	0	0	0	0	0	0	0	0
2691	207	6	0	0	0	0	0	0	0	0
2699	207	6	0	0	0	0	0	0	0	0
2708	207	6	0	0	0	0	0	0	0	0
2716	207	6	0	0	0	0	0	0	0	0
2741	207	6	0	0	0	0	0	0	0	0
2774	207	6	0	0	0	0	0	0	0	0
2808	207	8	32	0	72	0	0	14	13	0
2842	207	6	0	0	0	0	0	0	0	0
2851	207	6	0	0	0	0	0	0	0	0
2930	207	6	0	0	0	0	0	0	0	0
2934	207	6	0	0	96	0	0	0	0	0
2939	207	6	0	0	96	0	0	0	0	0
2946	207	6	0	0	0	0	0	0	0	0
2947	207	6	0	0	0	0	0	0	0	0
2948	207	6	0	0	0	0	0	0	0	0
2991	207	6	0	0	96	0	0	0	0	0

Tabela 8.4-Frames do grupo de luzes monitorado

GRUPO FREIOS

Testes semelhantes para o monitoramento para identificar do grupo freios, freio de mão e freio pedal.

ID = 207 D ----------D0 = 128D = F0H .....lâmpada do freio pedal (traseira)

ID = 243D.....BYTES = D0

D0 = 32D…….Freio mão

A Tabela 8.5 mostra o monitoramento relacionado aos freios.

Record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
580	207	6	0	0	0	0	0	0	0	0
589	207	6	0	0	0	0	0	0	0	0
598	207	6	0	0	0	0	0	0	0	0
607	207	6	0	0	0	0	0	0	0	0
665	207	6	0	0	0	0	0	0	0	0
783	207	6	0	0	0	0	0	0	0	0
808	207	6	0	0	0	0	0	0	13	0
816	207	6	0	0	0	0	0	0	0	0

Record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
859	207	6	0	0	0	0	0	0	0	0
954	207	6	0	0	0	0	0	0	0	0
995	207	6	128	0	0	0	0	0	0	0
1028	207	6	128	0	0	0	0	0	0	0
1037	207	6	128	0	0	0	0	0	0	0
1107	207	6	128	0	0	0	0	0	0	0
1201	207	6	128	0	0	0	0	0	0	0
1235	207	6	128	0	0	0	0	0	0	0
1262	207	6	128	0	0	0	0	0	0	0
1357	207	6	128	0	0	0	0	0	0	0
1408	207	6	128	0	0	0	0	0	0	0
1442	207	6	128	0	0	0	0	0	0	0
1460	207	6	128	0	0	0	0	15	13	0
44	243	3	0	0	0	0	0	0	0	0
79	243	3	0	0	0	0	0	0	0	0
80	243	3	0	0	0	0	0	0	0	0
133	243	3	0	0	0	0	0	0	0	0
169	243	3	0	0	0	11	0	0	0	0
205	243	3	0	0	0	0	0	0	0	0
238	243	3	0	0	0	0	0	0	0	0
239	243	3	0	0	0	0	0	0	0	0
307	243	3	0	0	0	0	0	11	0	0
359	243	3	32	0	0	0	0	15	13	0
360	243	3	32	0	0	0	0	15	0	0
395	243	3	32	0	0	0	0	0	0	0
431	243	3	32	0	0	0	0	0	0	0
447	243	3	32	0	0	11	0	0	0	0
466	243	3	32	0	0	0	0	0	0	0
865	243	3	0	0	0	0	0	0	0	0
883	243	3	0	0	76	0	0	0	0	0
900	243	3	0	0	0	0	0	0	0	0
901	243	3	0	0	76	0	0	0	0	0
917	243	3	0	0	0	0	0	0	0	0
970	243	3	0	0	0	0	0	0	0	0

Tabela 8.5-Monitoramento do freio

GRUPO MOTOR

Testes semelhantes levaram aos dados relacionados ao Grupo Motor.

ID = 221D.....D0, D1, D2, D3, D4, D5 e D6

D0 = 128D.....Motor desligado

D0 = 0.....Motor ligado.

D1 = 16D .....Injeção de combustível.

D2 ..... Temperatura do motor

D5 e D6.....Rotação do motor.

A Tabela 8.6 mostra o monitoramento das informações enviadas ao painel de instrumentos. A ocorrência mostra a chave ligada, injeção de combustível, temperatura do motor semi-aquecido e motor sendo ligado e desligado.

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
8	221	7	0	0	113	0	0	0	0	0
11	221	7	0	16	113	0	0	0	0	0
12	221	7	0	16	113	0	0	0	0	0
13	221	7	0	16	113	0	0	0	0	0
14	221	7	128	16	113	0	0	0	0	0
15	221	7	128	16	113	0	0	0	0	0
17	221	7	128	16	113	0	0	0	0	0
18	221	7	128	16	113	0	0	0	0	0
19	221	7	128	16	113	0	0	0	0	0
20	221	7	128	16	113	0	0	0	0	0
21	221	7	128	16	113	0	0	0	0	0
24	221	7	128	16	113	0	0	0	0	0
25	221	7	128	16	113	0	0	0	0	0
26	221	7	128	16	113	0	0	0	0	0
41	221	7	128	16	113	0	0	0	0	0
42	221	7	128	16	113	0	0	0	0	0
820	221	7	0	0	113	0	0	198	38	0
821	221	7	0	0	113	0	0	127	37	0
823	221	7	0	0	113	0	0	195	37	0
824	221	7	0	0	113	0	0	195	37	0
825	221	7	0	0	113	0	0	241	37	0
828	221	7	0	0	113	0	0	139	35	0
830	221	7	0	0	113	0	1	213	34	0
3609	221	7	128	0	124	0	0	0	0	0
3611	221	7	128	0	124	0	0	0	0	0
3612	221	7	128	0	124	0	0	0	0	0

record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
3616	221	7	128	0	124	0	0	0	0	0
3618	221	7	128	0	124	0	0	0	0	0

Tabela 8.6-Monitoramento motor

8.4.3 Confirmação das Identificações dos Atuadores Monitorados

As informações monitoradas pelo sistema desenvolvido comprovaram as informações fornecidadas pela Alfateste, a respeito da troca de dados entre as unidades da eletrônica embarcada e painel pela rede CAN [5].

A partir dos identificadores e dados coletados no monitoramento foi possível a transmissão de um dado para um atuador qualquer, como exemplo: a rotação do motor. Para confirmar o identificador e respectivo byte do sistema de sinalização foi transmitido a partir do VisualCAN, um dado de valor máximo, 255D, detalhado na Figura 8.21, indicando a maior rotação, confirmando o acionamento no painel.

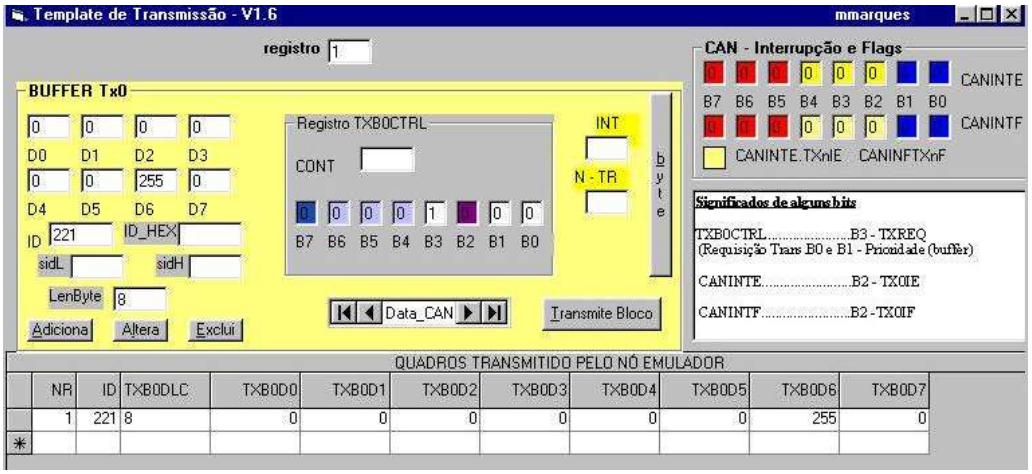


Figura 8.21 - Tela de transmissão do VisualCAN-dado indicador de RPM do motor do painel

8.5 MONITORAMENTO DO SISTEMA AUTOMOTIVO II

O sistema monitorado é o automóvel modelo Fiesta-Personalite da FORD com motor ZETEC ROCAN, que possui rede interna CAN. Neste sistema automotivo, o acesso ao barramento também é feito pelo conector DLC (data link connector), com o padrão de conector OBDII / J1962 (pinos 6 e 14), com uma taxa de transmissão de 500Kbits/s. (HS-CAN) definido pela SAE J2284 e baseada na ISO 11898. A identificação do conector foi

feita através do informativo “Tabela de Aplicação Rasther “ (Tecnomotor) [2]. O procedimento realizado para conexão com a interface física foi o mesmo para o primeiro sistema automotivo, Figura 8.22.

Figura 8.22 Detalhe do conector OBDII/J1962 no Fiesta

Figura 8.23 - Conexão cabo/DLC

Foi montada a estrutura de ligação através da conexão entre a interface e sistema automotivo da Ford mostrada na Figura 8.23, O sinal do barramento CAN foi monitorado pela bancada composta de PC monitor, osciloscópio e interface física , mostrado na Figura 8.25.

Figura 8.24 - Sistema II – Ford Fiesta

Figura 8.25– Mesa de teste com Interfaces

Após a conexão da mesa de teste ao barramento CAN do carro, mostrada na Figura 8.26, foi estabelecida a identificação do sinal e a programação da comunicação, Figura 8.27.

Figura 8.26- Ligação carro – bancada

Figura 8.27 – Tela de Monitoramento

O mesmo procedimento executado com o Palio foi realizado no Fiesta para estimar a taxa de transmissão no barramento CAN. Com osciloscópio de alta freqüência identificou-se o sinal no barramento, Figura 8.29, sinal este estimado em 500.000 bits/s.

Figura 8.28 - Sistema sob monitoramento

Figura 8.29 - Sinal do barramento CAN

Para ajustar a programação da interface física, através da Tela de Configuração, foram estabelecidos os parâmetros que definem a forma e o tempo do sinal serial de comunicação entre a interface física e o sistema automotivo.

Deste modo, o nó PC de monitoramento fica com o crystal de 20MHz.

Freqüência do oscilador da interface física (nó PC de monitoramento)➔ fosc = 20MHz

Tempo de Bit = 1 / Baude Rate = 1 / 500.000 = 2 μs (1)

Registro CNF1➔ SJW = 0

Registro CNF2➔ Prop Seg = 2 Phase Seg1 = 2

Registro CNF3➔ Phase Seg2 = 2

Obs: Valores acima foram estimados

Tbit = TQ * ((SJW + 1) + (Prop_Seg + 1) + (Phas_Seg1 + 1) + (Phas_Seg2 + 1)[7]

TQ = Tbit / ((SJW + 1) + (Prop_Seg + 1) + (Phas_Seg1 + 1) + (Phas_Seg2 + 1)) (2)

Substituindo 1 na equação 2 resulta:

TQ = 2μs / (1 + 3 + 3 + 3) = 200 ns (3)

TQ = 2 x (BRP + 1) / fosc (4) [7]

Substituindo fosc=20MHz e a equação 3 tem-se:

BRP = (TQ x Fosc / 2) – 1 ➔ **BRP = 1 (que deve ser ajustado na camada física)**

$$TQ = \frac{2*(BRP+1)}{fosc} = \frac{2*(1+1)}{20*10^6} * \ldots\ldots\ldots\ldots TQ = 200\eta seg$$

Definidos os parâmetros dos registradores de configuração, configura-se a interface visual, como na Figura 8.30, e transmite-se esta configuração para a interface física.

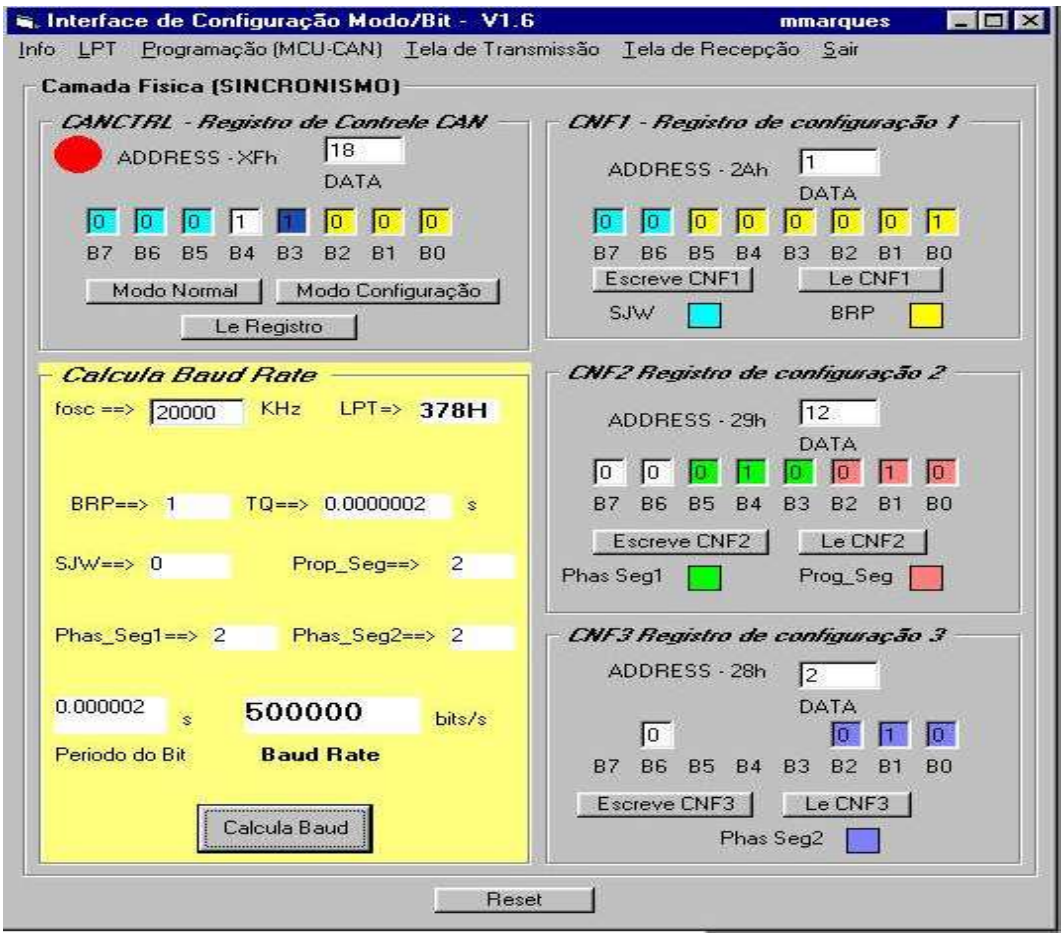


Figura 8.30 - Configuração dos registros de configuração do processador CAN

A taxa de transmissão e recepção de 500Kbits (HS-CAN – pinos 6 e 14), característica utilizada pela FORD [12], para transmissão de informações relacionadas a vários sistemas do carro (informações de parâmetros do motor para o painel do carro e informações entre os vários módulos da eletrônica embarcada), algumas delas comprovadas no monitoramento realizado a seguir.

8.5.4 Monitoramento de Sensores e Atuadores

Informações colhidas a partir do monitoramento do Fiesta deram conta que as informações disponíveis no barramento CAN através do conector são mais relacionadas ao motor e suspensão, enquanto que no Palio as informações estão relacionadas ao motor e a carroceria. Como a taxa de transmissão do Fiesta é 10 vezes maior que a taxa de transmissão no barramento do Palio, trafega uma maior quantidade de informações no mesmo período de tempo.

No monitoramento realizado foram identificados sensores e atuadores relacionados ao motor e carroceria.

Sensores Monitorados

A partir da Tabela 8.1, consegui-se monitorar dois sensores relacionados ao freio, a partir das ocorrências monitoradas:

1. Freio de mão acionado;
2. Freio de mão solto;
3. Freio de mão acionado;
4. Sensor de nível de óleo aberto + freio de mão solto;

Record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
43	1072	7	232	254	214	0	0	0	32	0
394	1072	7	232	254	214	0	0	0	32	0
397	1072	7	232	254	213	0	0	0	32	0
398	1072	7	232	254	213	0	0	0	32	0
400	1072	7	232	254	212	0	0	0	32	0
403	1072	7	232	254	213	0	0	0	32	0
404	1072	7	232	254	213	0	0	0	32	0

Record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
406	1072	7	232	254	213	0	0	0	32	0
410	1072	7	232	254	213	0	0	0	32	0
412	1072	7	232	254	213	0	0	0	32	0
415	1072	7	232	254	212	0	0	0	32	0
416	1072	7	232	254	212	0	0	0	32	0
552	1072	7	232	254	214	0	0	0	0	0
553	1072	7	232	254	214	0	0	0	0	0
555	1072	7	232	254	214	0	0	0	0	0
557	1072	7	232	254	213	0	0	0	0	0
558	1072	7	232	254	213	0	0	0	0	0
559	1072	7	232	254	213	0	0	0	0	0
564	1072	7	232	254	213	0	0	0	0	0
565	1072	7	232	254	213	0	0	0	0	0
570	1072	7	232	254	213	0	0	0	0	0
571	1072	7	232	254	213	0	0	0	0	0
574	1072	7	232	254	213	0	0	0	0	0
577	1072	7	232	254	214	0	0	0	0	0
578	1072	7	232	254	214	0	0	0	0	0
580	1072	7	232	254	213	0	0	0	0	0
617	1072	7	232	254	214	0	0	0	32	0
618	1072	7	232	254	214	0	0	0	32	0
621	1072	7	232	254	214	0	0	0	32	0
623	1072	7	232	254	214	0	0	0	32	0
624	1072	7	232	254	214	0	0	0	32	0
628	1072	7	232	254	213	0	0	0	32	0
1060	1072	7	232	254	213	0	0	0	0	0
1066	1072	7	232	254	213	0	0	0	0	0
1067	1072	7	232	254	213	0	0	0	0	0
1069	1072	7	232	254	213	0	0	0	0	0
1072	1072	7	232	254	213	0	0	0	0	0
1073	1072	7	232	254	213	0	0	0	0	0
1075	1072	7	232	254	213	0	0	0	0	0
1078	1072	7	232	254	213	0	0	0	0	78
1079	1072	7	232	254	213	0	0	0	0	0
1085	1072	7	232	254	212	0	0	0	0	0
1086	1072	7	232	254	212	0	0	0	0	0
1088	1072	7	232	254	212	0	0	0	0	0

Record	ID	RXB0DLC	RXB0D0	RXB0D1	RXB0D2	RXB0D3	RXB0D4	RXB0D5	RXB0D6	RXB0D7
1092	1072	7	232	254	213	0	0	0	0	0
1094	1072	7	232	254	213	0	0	0	0	0
1098	1072	7	232	254	213	0	0	0	0	0
1287	1072	7	232	254	213	0	0	0	0	0
1666	1072	7	232	254	213	0	0	0	64	0
1667	1072	7	232	254	213	0	0	0	64	78
1673	1072	7	232	254	214	0	0	0	64	0
1679	1072	7	232	254	213	0	0	0	64	0
1684	1072	7	232	254	213	0	0	0	64	0
1685	1072	7	232	254	213	0	0	0	64	0
1686	1072	7	232	254	213	0	0	0	64	0
1688	1072	7	232	254	213	0	0	0	64	0
1691	1072	7	232	254	214	0	0	0	64	0
1692	1072	7	232	254	214	0	0	0	64	0
1697	1072	7	232	254	213	0	0	0	64	0
2214	1072	7	232	254	212	0	0	0	0	0

Tabela 8.7: Monitoramento sensores do freio

A partir da Tabela 8.7 chegou-se a seguinte conclusão:

ID = 1072D .....byte ➔ D6

D6 = 32D (freio de mão)

D6 = 64D (nível de óleo do freio)

Figura 8.31 - Simulação de sensor aberto – nível de óleo de freio

Figura 8.32 - Simulação do sensor aberto temperatura do ar de admissão

Para a identificação dos atuadores no painel do carro, foram analisados 8000 frames, com ocorrências diversas, como por exemplo: motor desligado um período de tempo, ligado logo após e desligado novamente. A partir desta análise chegou-se aos identificadores e seus respectivos bytes.

- Atuadores:

ID = 513D .....bytes ➔ D0, D1 e D4

D0 = 0 a 255D (valor analógico da rotação do motor – painel)

D1 (zero da rotação do motor – painel)

D4 = 0 a 255D (valor analógico referente à velocidade do carro – painel)

ID = 1056D .....bytes➔ D0 e D5

D0 = 0 a 255D (valor analógico referente ao arrefecimento – painel)

Entre 0 e 164D (até o limite de temperatura)

Entre 165D a 255D (aciona sistema de sinalização visual no painel).

Figura 8.33 -Transmissão do byte D0=255D - Indicador digital de temperatura indicando máxima temperatura no motor

Obs: No monitoramento do sistema de arrefecimento do motor constatou-se que na faixa de temperatura de 0 a 164D corresponde ao limite tolerado sem alarme, ou seja, lâmpada de sinalização de temperatura do motor. A partir de 165D a 255D representa a faixa de alarme ativado, onde a lâmpada sinalizadora do painel é ativada.

D5 = 4D (lâmpada da trava no painel - cadeado)

Confirmação de Identificador

Após monitoramento e identificação de dispositivos sensores e atuadores no sistema automotivo II, procedeu-se o teste de confirmação dos dados identificados. Utilizou-se do ID de identificação do velocímetro, que está situado no painel do carro. Sendo que o dado transmitido é analógico, correspondente digital na faixa de 0 a 255D. A transmissão ocorreu a partir da tela de transmissão no byte D4 identificado pelo ID513D, conforme Figura 8.34.

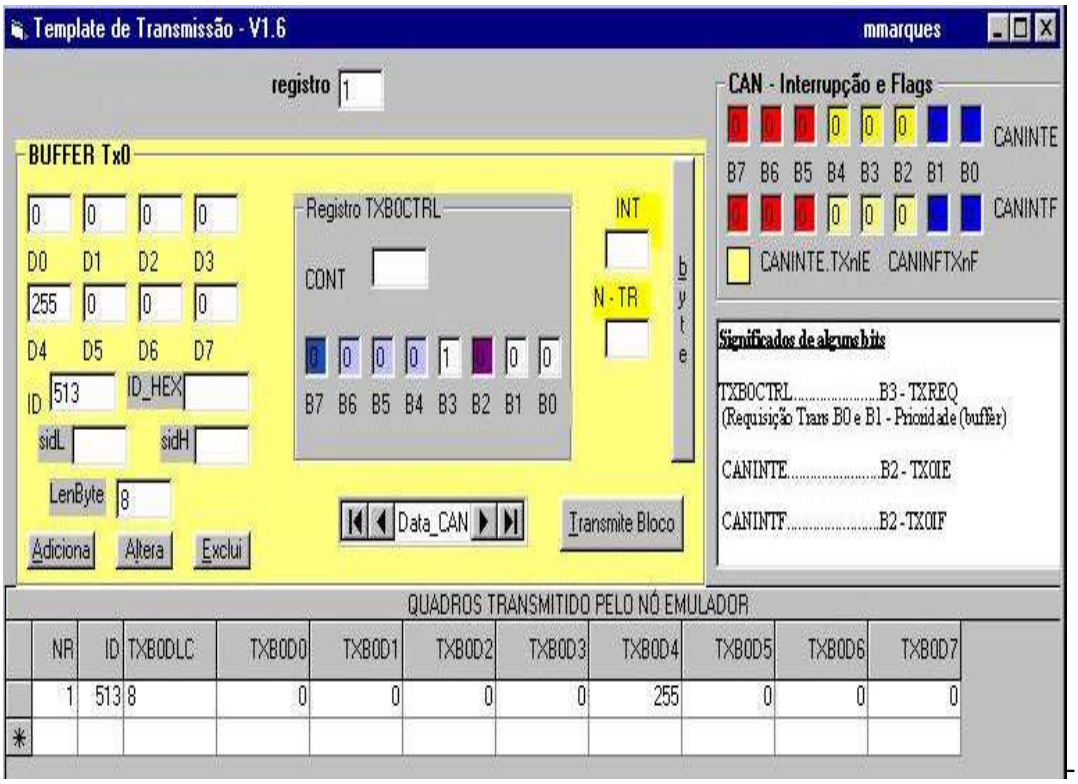


Figura 8.34 - Interface visual de transmissão – transmitido dado para o velocímetro do Fiesta

Figura 8.35- Resultado da transmissão do frame que identifica velocímetro – valor digital máximo

Ajustado o ID monitorado anteriormente na tela de transmissão do frame com o byte de endereçamento ajustado, por exemplo, com o valor digital máximo (255D), resultou na indicação mostrada pela foto da Figura 8.35. No detalhe, da figura, verifica-se que a rotação do motor é nula enquanto a indicação de velocidade do carro é máxima. Confirma-se o identificador e o byte correspondente desse dispositivo de indicação de velocidade.

9 CONCLUSÕES E TRABALHOS FUTUROS

Algumas das características importantes do protocolo CAN estão na sua velocidade de transferência e processamento de informações atingindo ótimos tempo, na alta confiabilidade, na possibilidade de corrigir possíveis erros de transmissão e recepção e no baixo custo da implementação de um nó em atividades remotas com microcontroladores como os das linhas Microdigital (PIC-CAN) e Texas (DSP-CAN) para aplicações em tempo real.

No entanto, pelo fato dos dados na área automotiva serem confidenciais, tanto dos fabricantes de automóveis nacionais como internacionais, houve muita dificuldade em se obter informações especificas dos seus comandos, status, endereços e significados, trocados pela rede CAN, independentemente de sua marca e modelo.

A realização deste trabalho envolveu então muita pesquisa, foi de grande importância para o entendimento do funcionamento do barramento CAN, e disponibilizou uma ferramenta poderosa para monitoramento de dados automotivos que trafegam pela rede CAN para qualquer interessado no campo. Possibilitou o entendimento do hardware e do software da estrutura padronizada automotiva, abrindo a possibilidade de acesso a qualquer modelo de automóvel baseado no protocolo CAN. Isto se tornou importante pois a indústria automotiva nacional carece de informações técnicas pertinentes a este tipo de aplicação tão recente neste setor, uma vez que hoje o setor automotivo está a mercê de tecnologias importadas. O trabalho poderá, portanto, contribuir para o acesso de profissionais no setor de manutenção automotiva, agregando-os ao mercado nacional e internacional, podendo abrir oportunidades de crescimento técnico e pessoal.

Além disto, o protocolo CAN, por ser um protocolo não proprietário, vem também sendo implementado em barramentos de campo na automação industrial. Assim a manufatura e o controle de processo estão hoje utilizando esta tecnologia para troca de dados de naturezas diversas entre controladores e mesmo entre redes de tecnologias diferentes.

A utilização do sistema desenvolvido com uma linguagem de programação visual somada à pesquisa envolvida neste trabalho possibilitará um melhor entendimento do barramento CAN em qualquer aplicação. Deste modo, poderá contribuir para o desenvolvimento de ferramentas diversas que podem ser integradas a sistemas de controle e produção na indústria nacional.

As perspectivas para a utilização deste trabalho ainda na área automotiva estão na sua utilização em aplicações mais específicas, tais como: sistemas de monitoramento do rendimento do veículo, sistemas de manutenção para oficinas e concessionárias, sistemas de manutenção à distância acoplados a um sistema de comunicação por satélite ou tecnologias sem fio, sistemas de vigilância de veículos, sistemas de controle de veículos para *Field Service*, sistemas de controle de semeadura, sistemas de controle de trens, assim como na indústria marítima, na aviação, nas áreas de saúde como a medicina, bioengenharia e muitas outras.

Não há dúvidas que, em alguns poucos anos, o mercado nacional deverá estar adaptado a estas novas tecnologias para poder ser mais competitivo com o mercado internacional, tendência evidente na presença desta e de outras tecnologias em alguns dos automóveis atualmente em circulação.

REFERÊNCIA BIBLIOGRÁFICA

[1] - INTREPIDCS [*on line*, **http://www.intrepidcs.com/neovi/help/neoOBDII.htm**, capturado em outubro de 2003]

[2] - TECNOMOTOR [*on line,* **www.tecnomotor.com**, capturado em setembro de 2001]

[3] - ALFATEST – [*on line*, **http://www.alfateste.com.br**; capturado em 4/set/2003]

[4] - OKI SEMICONDUCTOR – [on line, **http://www2.okisemi.com/us/docs/Can.html**, capturado em 11/set/2003]

[5] - ALFATEST2 –[*on line*, **http://alfatest.com.br/noticias/multiplexagem.html**; capturado em 10/out/2003].

[6] - CAN-CIA – [*on line*, **http://www.can-cia.de/donwloads/files**, capturado em 05/set/2003].

[7] - MICROCHIP - Microchip Thechnology ,MCP-CAN Development Guide, 2002.

[8] - PARET, Dominique, Le Bus CAN Description – De la théorie à la pratique, Dunod, Paris, 1996.

[9] - MICROCHIP - MCP2520 Development Kit User's Guide, 2002

[10] - AXELSON, Jan, Parallel Port Complete – Programming, Interfacing, & Using the PC's Parallel Port, Lakeview Research, USA,2000
capturado em outubro de 2003]

[11] – LUPINI, Christhopher A. , Multiplex Bus Progression 2003, 2003 SAE World Congress, Detroit, Michigan, March 2003

[12] - FORD – [*on line*, http://www.eng.ford.com, capturado em 10/03/2003].[13] - KVASER – [*on line*, http://www.kvaser.com.usa, capturado em 15/06/2003]

[14] - INJETRONIC [*on line*, http://www.injetronic.com.br/dicas.html, capturado em 20/08/2003]

[15] – COMER, Douglas E., Redes de Computadores e Internet / Douglas E. Comer; trad. Marinho Barcellos. – 2.ed. , BookMan, Porto Alegre, 2001

[16] - ATV – Advanced Vehicle Technologies, Inc– [*on line*, http://www.avt-hq.com, capturado em 17/07/2003]

[17] – MOTOROLA – [*on line*, http://www.motorola.com.usa, In Vehicle Networking, capturado em 20/04/2004]

[18] – B&B –ELECTRONICS – [*on line*, http://autotap.com.usa; capturado em 22/09/2003]

[19] - CALLAHAN, Evan Access 2002 Visual Basic, Makron Books, São Paulo, 2003

[20] – ABCWC – [on line, http://www.abcwc.net/accounts/quanta/FAQ.html, capturado em 28/11/2002]

[21] – TRAN, Eushian, Multi-Bit Error Vulnerabilities in CAN Protocol, Carnegie Mellon University, Pittsburgh, PA, May 1999

[22] – HARTTWICH, Florian, BASSEMIR, Armin, The Configuration of the CAN Bit Timing, Robert Bosch GmbH, 6th International CAN Conference, November 2000

[23] – TOMGREN, Martin, A perpective to the Design of Distributed Real Time Control Aplplications based on CAN, The Royal Institute of Technology, Stockholm, Swedem, May 2000

[24] – FREDRIKSSON, Lars Berno, Advantages with a global clock in CAN systems, KVASER, Sweden, 1999

BIBLIOGRAFIA COMPLEMENTAR

TANENBAUM, ANDREW S., Organização Estruturada de Computadores, ABDR-Printice Hall, rio de Janeiro,1999

LANG, TOMAS, Sistemas Digitais, Bookman, Porto Alegre, 2000

GMBH, ROBERT BOSCH, CAN Especification V2.0, 1991

KVASER, CAN Especification, 2000

MICROCHIP THECHNOLOGY ,MCP-CAN Development Guide, 2002

FRANCO, LUCIA, Anotações de aula, Curso de FieldBus, 2000

Manuais e Recursos de Multimídia

1. CD ROM Demo Networking Tools, Vector Informatik GmbH, Alemanha- 04/2001
2. VideoCarro – Injeção Eletrônica- Serviços de Treinamento Especiais –SETE
    a. Sistema IAW 1 AVB
    b. Conceitos de Injeção
    c. Sistema MPI
3. CD ROM Microchip Technical Library, First Edition 2001, USA- 03/2001
4. Papers – ICC (International CAN Congress), 1998, 1999, 2000, 2001
5. Manuais VolksWagen 2001 (Golf)
6. Manual PHILIPS PCA28C250, Controlador de interface CAN - 01/2000
7. Manuais Microchip:
8. MCP2510 - Controlador de interface SPI - 2000

Home Pages:

9. Motorola- www.motorola.com
10. Alfatest – www.alfatest.com
11. Tecnmotor – www.tecnomotor.com
12. CIA – www.can-cia.de
13. www.can-cia.org
14. Microchip – www.microchip.com
15. Vector – www.vector-cantech.com
16. Accutest - www.accutest.co.uk
17. IME- http://www.ime-actia.de
18. Embedded Systems Academy - www.esacademy.com
19. Kvaser CAN Solutions - www.kvaser.com
20. Ford - www.ford.com
21. Fiat – www.fiat.com
22. GM – www.gm.com