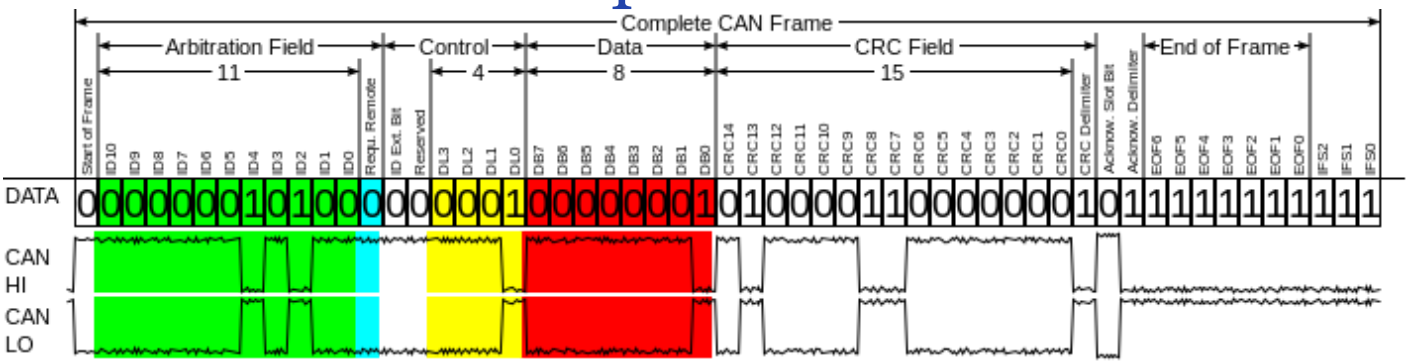
Rede CAN – O que é e como funciona



O CAN Bus (ou Barramento Controller Area Network) foi desenvolvido pela empresa alemã Robert BOSCH e disponibilizado em meados dos anos 80. Sua aplicação inicial foi realizada em ônibus e caminhões. Atualmente, é utilizado na indústria, em veículos automotivos, navios e tratores, entre outros.

CONCEITUAÇÃO BÁSICA

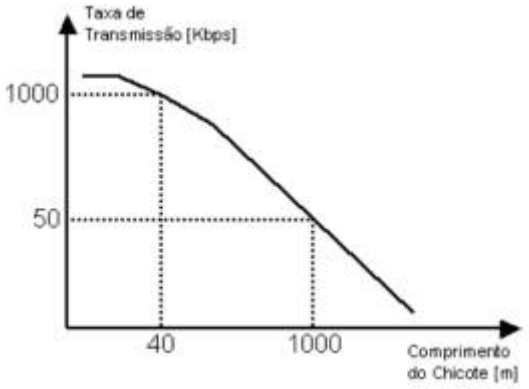
O CAN é um protocolo de comunicação serial síncrono. O sincronismo entre os módulos conectados a rede é feito em relação ao início de cada mensagem lançada ao barramento (evento que ocorre em intervalos de tempo conhecidos e regulares).

Trabalha baseado no conceito multi-mestre, onde todos os módulos podem se tornar mestre em determinado momento e escravo em outro, além de suas mensagens serem enviadas em regime multicast, caracterizado pelo envio de toda e qualquer mensagem para todos os módulos existentes na rede.

Outro ponto forte deste protocolo é o fato de ser fundamentado no conceito CSMA/CD with NDA (Carrier Sense Multiple Access / Collision Detection with Non-Destructive Arbitration). Isto significa que todos os módulos verificam o estado do barramento, analisando se outro módulo está ou não enviando mensagens com maior prioridade. Caso isto seja percebido, o módulo cuja mensagem tiver menor prioridade cessará sua transmissão e o de maior prioridade continuará enviando sua mensagem deste ponto, sem ter que reiniciá-la.

Outro conceito bastante interessante é o NRZ (Non Return to Zero), onde cada bit (0 ou 1) é transmitido por um valor de tensão específico e constante.

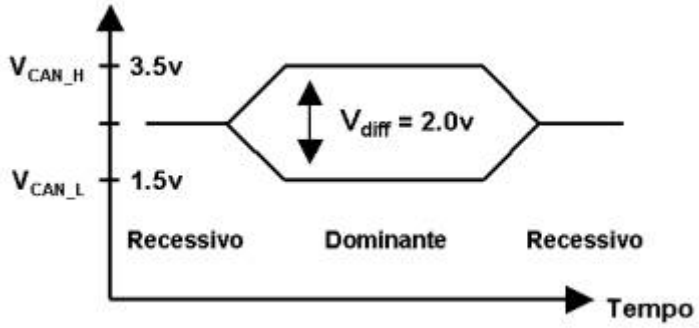
A velocidade de transmissão dos dados é inversamente proporcional ao comprimento do barramento. A maior taxa de transmissão especificada é de 1Mbps considerando-se um barramento de 40 metros. A Figura 1 representa a relação entre o comprimento da rede (barramento) e a taxa de transmissão dos dados.



Considerando-se fios elétricos como o meio de transmissão dos dados, existem três formas de se constituir um barramento CAN, dependentes diretamente da quantidade de fios utilizada. Existem redes baseadas em 1, 2 e 4 fios. As redes com 2 e 4 fios trabalham com os sinais de dados CAN_H (CAN High) e CAN_L (CAN Low). No caso dos barramentos com 4 fios, além dos sinais de dados, um fio com o VCC (alimentação) e outro com o GND (referência) fazem parte do barramento, levando a alimentação às duas terminações ativas da rede. As redes com apenas 1 fio têm este, o fio de dados, chamado exclusivamente de linha CAN.

Considerando o CAN fundamentado em 2 e 4 fios, seus condutores elétricos devem ser trançados e não blindados. Os dados enviados através da rede devem ser interpretados pela análise da diferença de potencial entre os fios CAN_H e CAN_L. Por isso, o barramento CAN é classificado como Par Trançado Diferencial. Este conceito atenua fortemente os efeitos causados por interferências eletro-magnéticas, uma vez que qualquer ação sobre um dos fios será sentida também pelo outro, causando flutuação em ambos os sinais para o mesmo sentido e com a mesma intensidade. Como o que vale para os módulos que recebem as mensagens é a diferença de potencial entre os condutores CAN_H e CAN_L (e esta permanecerá inalterada), a comunicação não é prejudicada.

No CAN, os dados não são representados por bits em nível “0” ou nível “1”. São representados por bits Dominantes e bits Recessivos, criados em função da condição presente nos fios CAN_H e CAN_L. A Figura 2 ilustra os níveis de tensão em uma rede CAN, assim como os bits Dominantes e Recessivos.



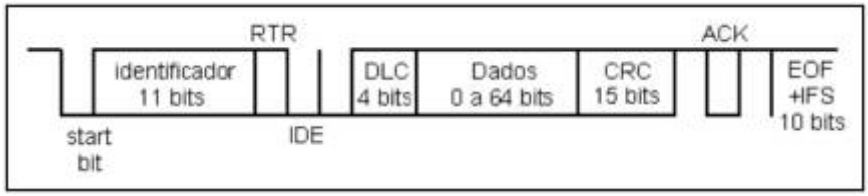
Como mencionado no início, todos os módulos podem ser mestre e enviar suas mensagens. Para tanto, o protocolo é suficientemente robusto para evitar a colisão

entre mensagens, utilizando-se de uma arbitragem bit a bit não destrutiva. Podemos exemplificar esta situação, analisando o comportamento de dois módulos enviando, ao mesmo tempo, mensagens diferentes. Após enviar um bit, cada módulo analisa o barramento e verifica se outro módulo na rede o sobrescreveu (vale acrescentar que um bit Dominante sobrescreve eletricamente um Recessivo). Um módulo interromperá imediatamente sua transmissão, caso perceba que existe outro módulo transmitindo uma mensagem com prioridade maior (quando seu bit recessivo é sobrescrito por um dominante). Este módulo, com maior prioridade, continuará normalmente sua transmissão.

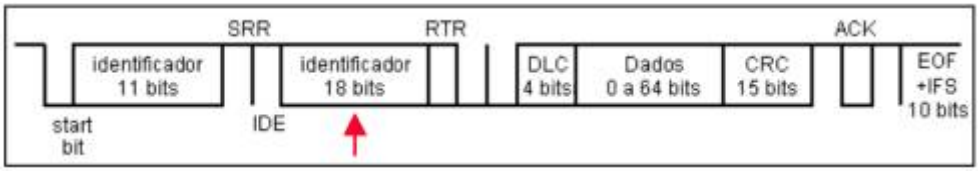
FORMATOS DAS MENSAGENS

Existem dois formatos de mensagens no protocolo CAN:

CAN 2.0A – Mensagens com identificador de 11 bits. É possível ter até 2048 mensagens em uma rede constituída sob este formato, o que pode caracterizar uma limitação em determinadas aplicações. A Figura 3 apresenta o quadro de mensagem do CAN 2.0A.



CAN 2.0B – Mensagens com identificador de 29 bits. É possível ter, aproximadamente, 537 milhões de mensagens em uma rede constituída sob este formato. Percebe-se que a limitação em virtude da quantidade de mensagens não mais existe. Por outro lado, o que pode ser observado em alguns casos é que, os 18 bits adicionais no identificador aumentam o tempo de transmissão de cada mensagem, o que pode caracterizar um problema em determinadas aplicações que trabalhem em tempo-real (problema conhecido como overhead). A Figura 4 apresenta o quadro de mensagem do formato CAN 2.0B.



PADRÕES EXISTENTES

Os fundamentos do CAN são especificados por duas normas: a ISO11898 e a ISO11519-2. A primeira, ISO11898, determina as características de uma rede trabalhando com alta velocidade de transmissão de dados (de 125Kbps a 1Mbps). A segunda, ISO11519-2, determina as características de uma rede trabalhando com baixa velocidade (de 10Kbps a 125Kbps).

Ambos os padrões especificam as camadas Física e de Dados, respectivamente 1 e 2 se considerado o padrão de comunicação OSI de 7 camadas (ISO7498). As demais camadas, da 3 à 7, são especificadas por outros padrões, cada qual relacionado a uma aplicação específica.

Existem diversos padrões fundamentados no CAN, dentre os quais podemos destacar:

· NMEA 2000: Baseado no CAN 2.0B e utilizado em aplicações navais e aéreas.

· SAE J1939: Baseado no CAN 2.0B e utilizado em aplicações automotivas, especialmente ônibus e caminhões.

· DIN 9684 – LBS: Baseado no CAN 2.0A e utilizado em aplicações agrícolas.

· ISO 11783: Baseado no CAN 2.0B e também utilizado em aplicações agrícolas.

Estes padrões especificam o equivalente às camadas de Rede (3), Transporte (4), Sessão (5), Apresentação (6) e Aplicação (7), do padrão OSI, incluindo-se as mensagens pertinentes ao dicionário de dados de cada aplicação em especial.

DETECÇÃO DE FALHAS

Algumas das maiores vantagens do CAN é a sua robustez e a capacidade de se adaptar às condições de falha, temporárias e/ou permanentes. Podemos classificar as falhas de uma rede CAN em três categorias ou níveis: Nível de Bit, Nível de Mensagem e Nível Físico.

Nível de Bit – Possui dois tipos de erro possíveis:

Bit Monitoring: Após a escrita de um bit dominante, o módulo transmissor verifica o estado do barramento. Se o bit lido for recessivo, significará que existe um erro no barramento.

Bit Stuffing: Apenas cinco bits consecutivos podem ter o mesmo valor (dominante ou recessivo). Caso seja necessário transmitir seqüencialmente seis ou mais bits de mesmo valor, o módulo transmissor inserirá, imediatamente após cada grupo de cinco bits consecutivos iguais, um bit de valor contrário. O módulo receptor ficará encarregado de, durante a leitura, retirar este bit, chamado de Stuff Bit. Caso uma mensagem seja recebida com pelo menos seis bits consecutivos iguais, algo de errado terá ocorrido no barramento.

Nível de Mensagem – São três os tipos de erro possíveis:

CRC ou Cyclic Redundancy Check: Funciona como um checksum. O módulo transmissor calcula um valor em função dos bits da mensagem e o transmite juntamente com ela.

Os módulos receptores recalculam este CRC e verificam se este é igual ao transmitido com a mensagem.

Frame Check: Os módulos receptores analisam o conteúdo de alguns bits da mensagem recebida. Estes bits (seus valores) não mudam de mensagem para mensagem e são determinados pelo padrão CAN.

Acknowledgment Error Check: Os módulos receptores respondem a cada mensagem íntegra recebida, escrevendo um bit dominante no campo ACK de uma mensagem resposta que é enviada ao módulo transmissor. Caso esta mensagem resposta não seja recebida (pelo transmissor original da mensagem), significará que, ou a mensagem de dados transmitida estava corrompida, ou nenhum módulo a recebeu.

Toda e qualquer falha acima mencionada, quando detectada por um ou mais módulos receptores, fará com que estes coloquem uma mensagem de erro no barramento, avisando toda a rede de que aquela mensagem continha um erro e que o transmissor deverá reenviá-la.

Além disso, a cada mensagem erroneamente transmitida ou recebida, um contador de erros é incrementado em uma unidade nos módulos receptores, e em oito unidades no transmissor. Módulos com estes contadores iguais a zero são considerados Normais. Para os casos em que os contadores contêm valores entre 1 e 127, os módulos são considerados Error Active. Contadores contendo valores entre 128 e 255 colocam os módulos em condição de Error Passive. Finalmente, para contadores contendo valores superiores a 255, os módulos serão considerados em Bus Off e passarão a não mais atuar no barramento. Estes contadores também são decrementados a medida que mensagens corretas são recebidas, o que reduz o grau de incerteza em relação a atividade dos módulos ora com contadores contendo valores diferentes de zero e possibilita novamente a plena participação deles no barramento.

Nível Físico – Para os barramentos com 2 e 4 fios, caso algo de errado venha a ocorrer com os fios de dados CAN_H e CAN_L, a rede continuará operando sob uma espécie de modo de segurança. Seguem abaixo algumas das condições de falha nas linhas de comunicação que permitem a continuidade das atividades da rede:

· Curto do CAN_H (ou CAN_L) para GND (ou VCC);

· Curto entre os fios de dados CAN_H e CAN_L;

· Ruptura do CAN_H (ou CAN_L);

ASPECTOS DE IMPLEMENTAÇÃO: DICIONÁRIO DE DADOS

É a parte mais dedicada à aplicação quando se trabalha com um protocolo como o CAN. O Dicionário de Dados (ou Data Dictionary) é o conjunto de mensagens que podem ser transmitidas naquela determinada rede.

A forma mais interessante de se organizar um dicionário de dados é criando uma matriz com todos os módulos da rede. Esta matriz mostrará cada mensagem sob a responsabilidade de cada módulo, relacionando quem a transmite e quem a recebe. Outros dados importantes nesta matriz são: o tempo de atualização dos valores da mensagem, o intervalo de transmissão da mesma e o valor relativo ao seu identificador. Além desta matriz, a documentação referente ao Dicionário de Dados deverá conter uma descrição detalhada de cada mensagem, bit a bit.

O Dicionário de Dados é implementado numa rede CAN via software e deverá ser o mesmo (ter a mesma versão de atualização, inclusive) em todos os módulos conectados à rede. Isto garantirá total compatibilidade entre os participantes do barramento.

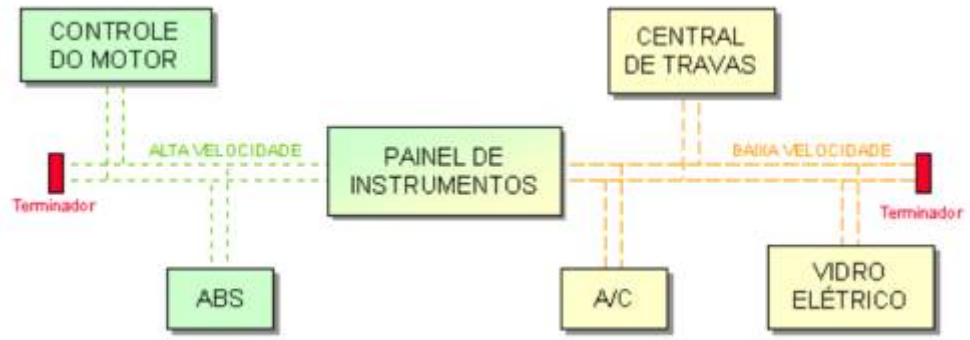
ASPECTOS DE IMPLEMENTAÇÃO: EXEMPLO DE REDE

Uma rede CAN, dependendo da sua aplicação, poderá ter até centenas de módulos conectados. O valor máximo para a conexão de módulos em um barramento depende da norma que se utiliza na dada aplicação.

Toda rede CAN possui 2 Terminadores. Estes terminadores nada mais são que resistores com valores entre 120 e 124 ohms, conectados à rede para garantir a perfeita propagação dos sinais elétricos pelos fios da mesma. Estes resistores, um em cada ponta da rede, garantem a reflexão dos sinais no barramento e o correto funcionamento da rede CAN.

Outra característica de determinadas aplicações fundamentadas no CAN é que estas poderão ter duas ou mais sub-redes trabalhando, cada qual, em uma velocidade diferente. Os dados são transferidos de uma sub-rede para a outra através de módulos que atuam nas duas sub-redes. Estes módulos são chamados de Gateways.

A Figura 5 ilustra a rede CAN de um sistema automotivo, com duas sub-redes e dois terminadores. O Gateway desta aplicação é o Painel de Instrumentos.



ASPECTOS DE IMPLEMENTAÇÃO: MONTAGEM DA REDE

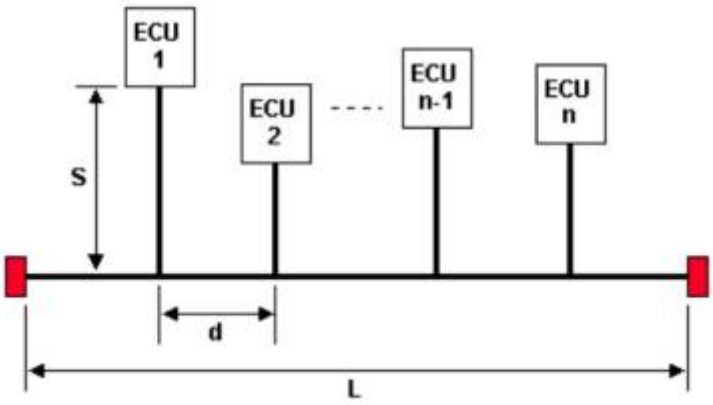
Barramento é o termo técnico que representa os condutores elétricos das linhas de comunicação e a forma como eles são montados. Apesar de parecer simples, o ato de interligar os módulos requer bastante atenção.

Sobre o cabeamento necessário, considerando-se uma aplicação CAN de dois fios, deve-se utilizar par trançado onde a secção transversal de cada um dos fios deve ser de no mínimo 0,35mm².

As duas terminações (resistores de aproximadamente 120 ohms), do ponto de vista teórico, podem ser instaladas nas extremidades do chicote, diretamente nos fios de dados CAN_H e CAN_L. Do ponto de vista prático isto é extremamente complexo. O que deve ser feito é adicionar as terminações nas duas ECUs (Unidades Eletrônicas de Controle) conectadas aos extremos da rede. Se as ECUs forem montadas dependendo dos opcionais do veículo, deve-se procurar instalar as terminações nas ECUs que sempre estarão presentes nele (veículo). As terminações são mandatórias numa rede CAN.

No momento de se projetar o roteamento do barramento, algumas regras em relação ao comprimento dos chicotes devem ser observadas. O sincronismo das operações das ECUs no CAN é fundamentado no tempo de propagação física das mensagens no barramento. Assim, a relação do comprimento de determinados intervalos do chicote no barramento são fundamentais ao bom funcionamento da rede.

A Figura 6 mostra um diagrama que ilustra as medidas que devem ser observadas no desenvolvimento do chicote.



Onde: S (máximo comprimento da ramificação) = 0,3m
d (mínima distância entre ramificações) = 0,1m
L (máximo comprimento da rede a 1Mbps) = 40m

Obs: O valor da distância "d" deve ser aleatório.

Destacamos que, após o barramento ser montado, caso seja necessário qualquer retrabalho no mesmo, é aconselhável a troca do chicote elétrico danificado. Emendas poderão alterar a impedância característica da rede e com isso afetar o seu funcionamento.

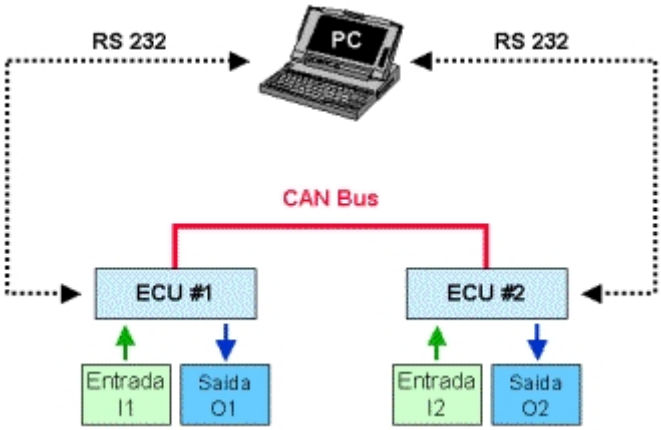
DETERMINAÇÃO DA ARQUITETURA

Considerando uma determinada aplicação, a primeira tarefa que devemos considerar durante o projeto de sua rede de comunicação de dados é a determinação da sua arquitetura. Neste ponto podemos enfrentar duas situações: ter que estabelecer uma rede de comunicação entre ECUs prontas e que não trabalhem em rede ou, ter que projetar totalmente as ECUs, considerando a leitura das entradas, seus devidos processamentos e atuações nas saídas, além da troca de dados através da rede propriamente dita.

Se tomarmos como ponto de partida uma aplicação onde as ECUs já estiverem prontas, nossa responsabilidade será fundamentalmente disponibilizar as informações de cada ECU no formato determinado pelo Protocolo CAN, além de estabelecer as conexões necessárias à comunicação de dados entre as próprias ECUs. Este cenário pode ser complexo de se lidar uma vez que nem todas as ECUs, da forma como foram originalmente projetadas, serão capazes de, facilmente, fornecer as informações sob sua responsabilidade para que o devido empacotamento no formato CAN seja realizado. De qualquer forma o trabalho é possível.

Se considerarmos uma aplicação onde somente o escopo do sistema que se deseja controlar estiver disponível, apesar de “partir do zero”, poderemos projetar as ECUs já considerando os controladores capazes de, facilmente, estabelecer comunicação fundamentada no protocolo CAN. Seguiremos então por esta linha de raciocínio.

A aplicação sobre a qual trabalharemos será simples. A Figura 1 apresenta a arquitetura proposta.



Esta arquitetura mostra-se extremamente simples. De qualquer forma vale destacar:

· Serão duas ECUs conectadas por uma rede de comunicação CAN Bus, onde ambas terão as mesmas responsabilidades:

· Ler algumas entradas digitais;

· Empacotar estes dados no formato determinado pelo CAN;

· Transmitir estes dados pela rede CAN à outra ECU;

· Receber dados da outra ECU, enviados pela rede CAN, e

· Processar estes dados, comandando as saídas necessárias.

· Através de uma linha de comunicação serial RS232 e um PC, poderemos programar e monitorar o funcionamento de cada ECU.

ANÁLISE DA NORMA RELACIONADA À APLICAÇÃO

Como mencionado na segunda parte desta matéria, existem diversas normas baseadas no protocolo CAN Bus, cada qual relacionada a uma classe de aplicações. Vamos recordar algumas destas normas:

· NMEA 2000: Utilizada em aplicações navais e aéreas.

· SAE J1939: Utilizada em aplicações automotivas.

· DIN 9684 – LBS: Utilizada em aplicações agrícolas.

· ISO 11783: Utilizada em aplicações agrícolas.

Estas normas trazem informações específicas de cada classe de aplicações, como o tipo de cabeamento e conectores necessários e os tipos de mensagens requeridas, entre outras.

Neste nosso desenvolvimento, não nos preocuparemos com uma norma específica. Trataremos simplesmente de projetar uma rede CAN considerando seu padrão de empacotamente de dados e transmissão e, quando necessário, estabelecendo alguns conceitos específicos para a nossa aplicação. O primeiro conceito que deveremos estabelecer é o chamado Dicionário de Dados, já explicado na segunda parte desta matéria.

DETERMINAÇÃO DO DICIONÁRIO DE DADOS

O Dicionário de Dados (D.D.) de uma aplicação pode ser resumidamente descrito como uma tabela que relaciona as mensagens existentes nesta aplicação (seus identificadores e dados) e as ECUs responsáveis por sua transmissão e recepção.

Vamos então estabelecer um D.D. para a nossa aplicação. A Tabela 1 traz as informações importantes ao nosso trabalho.

Nome da Mensagem	Matriz TX – RX		Bytes do Identificador (ID)	Bytes de dados							
	ECU #1	ECU #2		D #1	D #2	D #3	D #4	D #5	D #6	D #7	D #8

1	Estado das Entradas da ECU #1	TX	RX	12	34	56	78	**XX**	00	00	00	00	00	00	00
2	Estado das Saídas da ECU #1	TX	RX	11	22	33	44	00	00	**XX**	00	00	00	00	00
3	Estado das Entradas da ECU #2	RX	TX	87	65	43	21	**XX**	00	00	00	00	00	00	00
4	Estado das Saídas da ECU #2	RX	TX	55	66	77	88	00	00	00	00	00	**XX**	00	00

Analisando a Tabela 1 percebemos que, no nosso Dicionário de Dados:

- A mensagem “1” é chamada de “Estado das Entradas da ECU #1”;
- Ela é transmitida pela ECU #1 (TX) e recebida pela ECU #2 (RX);
- Seu Identificador (ID) tem valor igual a “12345678 hex”;
- Seus Bytes de Dados, excluindo-se o Byte “D #1”, são iguais a “00 hex”;
- Seu Byte de Dados “D #1” depende de algumas regras específicas, onde:
- “XX” será “31 hex” caso uma determinada entrada digital seja igual a “1”
- “XX” será “30 hex” caso esta determinada entrada digital seja igual a “0”

Para as demais mensagens, vale a mesma análise. A única observação que gostaríamos de fazer é que cada Byte de Dados da tabela que tiver seu valor igual a “XX” possuirá uma regra específica, assim como explicado para o Byte de Dados “D #1” da mensagem “1”.

Apesar do nosso D.D. parecer simples, e realmente é, ele demonstra o que efetivamente ocorre em uma aplicação real. Se analisarmos uma aplicação automotiva por exemplo, teremos para ela, mensagens relacionadas ao funcionamento do motor, dos freios ABS e dos sistemas de travamento e alarme, entre outras. Além disso, pela complexidade dos sistemas envolvidos, não somente um byte de dados por mensagem será utilizado (indicando a alteração do estado de uma determinada entrada). Teremos a utilização de todos os oito bytes existentes.

Estabelecido o Dicionário de Dados pertinente à nossa aplicação, resta agora a sua implementação efetiva, que se dá através de um software instalado dentro de cada ECU. Este software é conhecido como firmware (exatamente por trabalhar dentro da ECU).

Antes de abordarmos os softwares relacionados, falaremos sobre o projeto do hardware das ECUs.

PROJETO DO HARDWARE DAS ECUs

Consideraremos ambas as ECUs baseadas no mesmo projeto de hardware – com a mesma quantidade de Entradas e Saídas e uma porta de comunicação serial RS232.

Quando projetamos uma ECU para a sua utilização em uma rede de comunicação de dados baseada no Protocolo CAN Bus, temos duas alternativas tecnológicas em relação à execução do processamento: utilizar um Micro-Controlador (sem CAN) conectado a um Controlador CAN (o que caracteriza dois CIs distintos e interligados) ou utilizar um Micro-Controlador com CAN incorporado.

Quando utilizamos um Micro-Controlador com CAN incorporado, este CI passa a ser responsável não só pela leitura e tratamento das entradas e o acionamento das saídas, como também pelo empacotamento dos dados no formato CAN (além da sua transmissão e recepção).

Quando utilizamos um Micro-Controlador sem CAN (conectando-o a um Controlador CAN específico), este ficará responsável pelo tratamento das entradas e saídas, trocando informações por portas de comunicação específicas com o Controlador CAN, responsável pelo empacotamento dos dados no formato do Protocolo (além da sua transmissão e recepção).

O que determina a utilização de um conceito ou do outro é, basicamente, a disponibilidade e o custo dos componentes. Do ponto de vista da implementação, acreditamos que a utilização de um Micro-Controlador com CAN incorporado seja mais simples, rápida e segura do ponto de vista técnico (especialmente em relação à Compatibilidade Eletro-Magnética).

Existem diversos fabricantes de CIs com vários tipos de Micro-Controladores disponíveis (com CAN e sem CAN incorporado). As Tabelas 2 e 3 mostram alguns fabricantes e seus componentes disponíveis. *(ver data sheets para maiores informações)*

Micro-Controladores com Controlador CAN

Fabricante	Componente
Infineon	C167CR-LM
Intel	87C196CA
Microchip	18Fxx8
Motorola	68376
Philips	P8xC591

Controladores CAN

Fabricante	Componente
Infineon	81C90
Intel	82527
Microchip	MCP2510
Philips	SJA1000

Nosso projeto do hardware considerará o Micro-Controlador com CAN incorporado P87C591 (Philips).

Além do Micro-Controlador, uma ECU com capacidade de comunicação via CAN precisa ter o chamado *Transceiver* ou Transmissor-Receptor. Este componente é responsável pela compatibilização dos níveis elétricos requeridos pela rede CAN com os níveis elétricos necessários ao trabalho do Micro-Controlador e vice-versa.

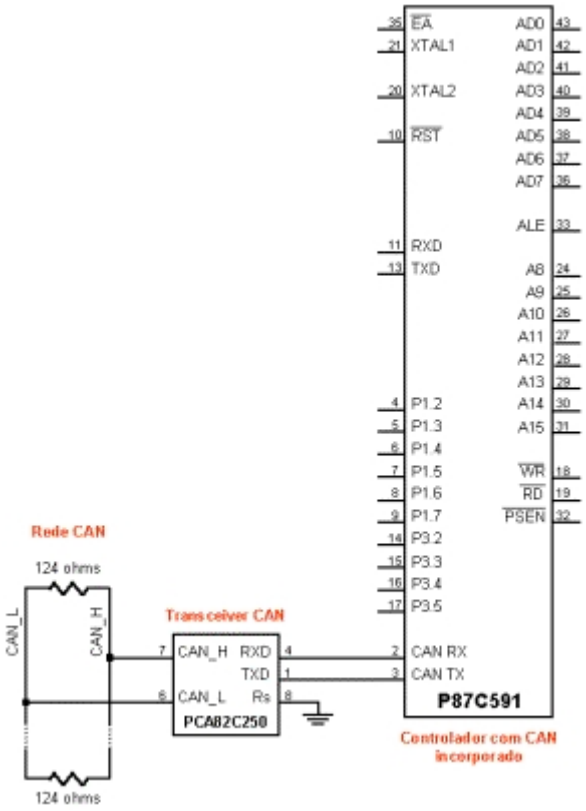
Existem diversos fabricantes de *Transceivers* CAN. A Tabela 4 mostra alguns fabricantes e seus componentes disponíveis. *(ver data sheets para maiores informações)*

Transceivers CAN

Fabricante	Componente
Bosch	CF151
Infineon	TLE6252G
Philips	PCA82C250
	TJA1050

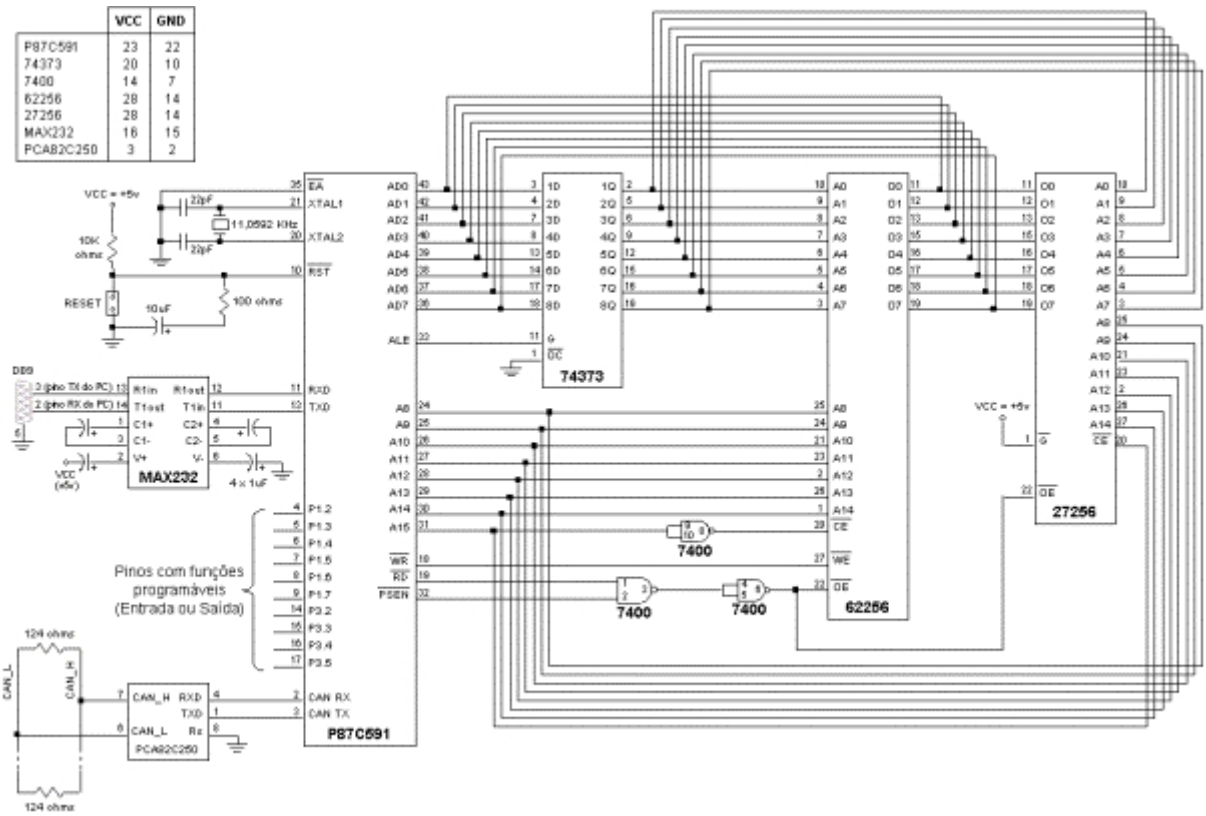
Nosso projeto do hardware considerará o Transceiver PCA82C250 (Philips).

Definidos os dois componentes principais das nossas ECUs, passemos ao entendimento de como conectá-los. A Figura 2 ilustra esta tarefa:



Através de dois fios conseguimos conector o Micro-Controlador ao Transceiver. Além disso, a ECU é conectada à rede CAN por outros dois fios, CAN_H e CAN_L, já mencionados na segunda parte desta matéria. Perceba também, na Figura 2, os dois Terminadores de 124 Ohms. Eles podem estar conectados diretamente ao chicote ou, no nosso caso, onde a aplicação possui somente duas ECUs, um em cada ECU.

Obviamente, na implementação efetiva de cada uma das ECUs, serão necessários outros componentes além dos dois anteriormente mencionados – Micro-Controlador e Transceiver. A Figura 3 mostra o diagrama elétrico das nossas ECUs, agora completas (lembre-se de que o projeto de hardware de ambas é o mesmo !!!).



Por se tratarem de ECUs destinadas ao aprendizado do Protocolo CAN, optamos pelo projeto apresentado, onde a execução de seus programas ocorre nas memórias EPROM e RAM, ao invés da memória interna do Micro-Controlador (neste caso um OTP – "Gravável somente uma vez"). Na memória EPROM gravamos um programa de uma categoria conhecida como Monitor, enquanto que, na memória RAM, gravamos os programas Principais das ECUs.

Vamos agora falar um pouco mais sobre as memórias, suas responsabilidades e os diversos programas envolvidos.

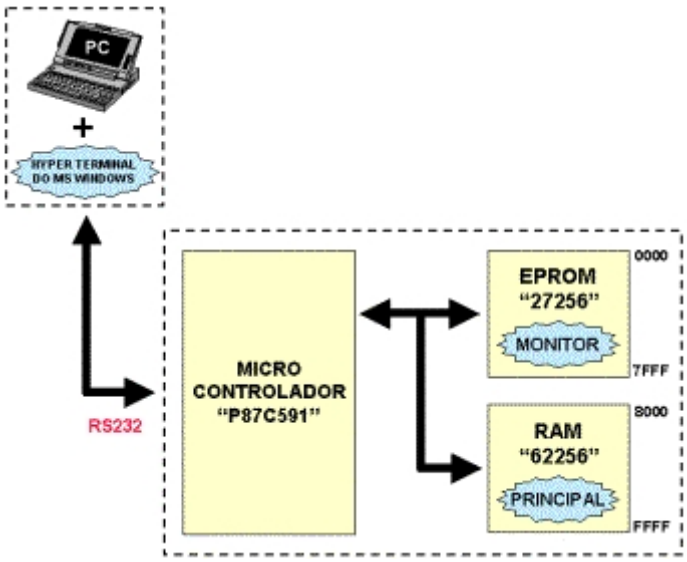
PROJETO DO SOFTWARE DAS ECUs

São dois os tipos de firmware (software executado dentro das ECUs) utilizados no nosso projeto:

· O primeiro é chamado de Monitor. É gravado na memória EPROM, a partir do seu endereço 0000 hex, e passa a ser executado toda vez que a ECU é reinicializada. A função principal deste programa Monitor é possibilitar a gravação e operação do programa Principal da ECU em sua memória RAM. Isto é possível através da operação de comandos do programa Monitor. Para tanto, a ECU precisa estar conectada a um PC via RS232 e, por exemplo, utilizando o aplicativo Hyper Terminal do MS Windows, poderemos transferir o programa Principal da ECU para a sua memória RAM. Qualquer programa Monitor para Micro-Controladores similares ao 8051 poderá ser utilizado neste nosso projeto. Nós utilizamos um dos vários Monitores disponíveis na internet (PAULMON – www.pjrc.com/tech/8051/ – acesso em: 20 Out 2001).

· O segundo é chamado de Principal. É gravado na memória RAM pelo processo descrito anteriormente e passa a ser executado por um dos comandos existentes no programa Monitor, (este comando desvia a execução do Micro-Controlador para a primeira posição de memória ocupada pelo programa Principal – endereço 8000 hex). Este programa Principal é responsável pela leitura e processamento das entradas, ativação das saídas, controle da linha de comunicação serial RS232 e da linha de comunicação CAN Bus.

A Figura mostra a distribuição dos vários programas mencionados e os endereços de início e término das memórias EPROM e RAM.



Ainda sobre o programa Principal, é necessário o desenvolvimento de algumas rotinas que devem realizar as seguintes operações:

· Inicialização da Porta de Comunicação Serial RS232;

· Inicialização da Porta de Comunicação CAN Bus (Baud Rate e os Filtros de Aceitação de Mensagens);

· Rotinas de Transmissão e Recepção via RS232;

· Rotinas de Transmissão e Recepção via CAN Bus;

· Rotina de leitura de uma Entrada Digital;

· Rotina de acionamento de uma Saída Digital.

As rotinas acima mencionadas podem ser escritas em linguagem C e o programa final pode ser compilado com o auxílio de qualquer compilador disponível na internet (por exemplo, o SDCC – Small Device C Compiler – www.sdcc.sourceforge.com – acesso em: 21 Nov 2001).

Alguns exemplos de rotinas programadas em C são mostradas a seguir:

· Início do Programa em C:

```
#include “87C591.h”
#define BYTE unsigned char
#define WORD unsigned int
#define IN P1_2
#define OUT P3_2
```

· **Definição da estrutura da mensagem CAN:**

```
typedef struct {
BYTE INFO; /* Byte com informações relacionadas a mensagem CAN */
BYTE ID[4]; /* 4 bytes de Identificador */
BYTE BUF[8]; /* 8 bytes de dados */
}
PDU;
```

· Inicialização da Porta RS232:

```
void init_serial() {
PCON = 0x80; /* SMOD1=1 e SMOD0=0 => Baud Rate dobrado */
TMOD = 0x20; /* Timer 1 autoload */
TCON = 0x40; /* Inicia o Timer 1 */
TH1 = 0x0FD; /* Taxa de transmissão: 19200*2=38400 */
SCON = 0x52; /* SM0=0 e SM1=1 => Modo 1 */
}
```

· Inicialização do CAN Bus:

```
void init_can (void) /* Baud Rate de 125kbps e XTAL de 11,0592 */
/* Assumindo CAN2.0B (ID de 29bits) e DLC = 8 bytes */
{
BYTE i;
CANMOD = 0x01; /* Força o Modo Reset para inicialização */
P1M2 = P1M2 | 0x02; /* TXDC Port (P1.1) Configuração: */
/* Pin TXDC para push-pull */
/* P1M2.1=’1', P1M1.1 = ‘0’ (default) */

CANSTA = 0x03; /* Interrupções de Recepção e Transmissão habilitadas */

/* BTR0 e BTR1 determinam o Baud Rate e o Sample Point Position */
CANADR = BTR0;
CANDAT = 0x85;
CANADR = BTR1;
CANDAT = 0x2A;
```