3 PROTOCOLO DE COMUNICAÇÕES CAN

3.1 SISTEMAS EM REDE *vs* SISTEMAS CENTRALIZADOS

Com o aumento da utilização de sistemas electrónicos compostos por vários microcontroladores e periféricos inteligentes, como é o caso do controlo em veículos automóveis, ganharam bastante importância protocolos de rede entre estas unidades em relação a sistemas que utilizam um processador central responsável pelas tarefas de controlo.

A escolha por uma solução em rede permite um conjunto de benefícios, em relação a uma solução centralizada, tais como [20]:

- Menos fios no sistema o que torna a cablagem mais simples e barata;
- Ligações curtas a sensores analógicos sensíveis a ruído eléctrico, antes dos sinais serem convertidos em mensagens digitais "imunes" a ruído;
- Sistema flexível. Unidades funcionais podem ser adicionadas ou removidas de forma simples;
- Manutenção do sistema. A electrónica de várias unidades pode ser idêntica, permitindo troca entre elas;
- Cada unidade pode ser desenvolvida e testada individualmente de acordo com requisitos exigidos pelo sistema;
- Partilha de dados entre as várias unidades pode eliminar redundância de informação.

Após apresentar os motivos que levam à adopção de uma arquitectura em rede, serão abordadas, no decorrer das restantes secções deste capítulo, as principais propriedades do protocolo de comunicações em rede que serviu de base ao sistema implementado, o protocolo CAN.

3.2 INTRODUÇÃO

3.2.1 Um pouco de história

O protocolo CAN foi desenvolvido por Robert Bosh em 1986 para aplicação na indústria automóvel, com o objectivo de simplificar os complexos sistemas de fios em veículos com sistemas de controlo compostos por múltiplos microcontroladores/micro-computadores para gestão do motor, sistema ABS, controlo da suspensão, etc. [21]. A sua especificação base anunciava elevada taxa de transmissão, grande imunidade a interferências eléctricas e capacidade de detectar erros [22].

A aplicação da tecnologia CAN, para partilha de dados e controlo em tempo real, tem vindo a tornar-se cada vez mais popular. Ao longo dos anos, o CAN evoluiu de aplicações dedicadas à indústria automóvel para outras de uso industrial e produtos envolvendo microcontroladores, com ligação por fio, e não só. Um dos benefícios do

CAN é suprimir a necessidade de sistemas complexos de fios substituindo-os por um simples cabo. É também considerado uma solução para implementar comunicação em rede de uma forma simples, barata e robusta, nomeadamente ao ruído electromagnético [23].

A utilização da electrónica, na melhoria da eficiência e funcionalidade dos sistemas actuais tornou, a multiplexagem uma necessidade para além de uma realidade, atingindo um nível onde a técnica de ligação ponto-a-ponto, tradicional, não consegue competir, economicamente, com o crescimento dos sistemas electrónicos. Actualmente utilizam-se sistemas interactivos onde a informação é partilhada através de um barramento comum. Uma vez que estas ligações aumentaram, a necessidade de interfaces série e de um protocolo de barramento aberto adquiriram bastante importância. O protocolo de comunicações CAN, descreve o método como a informação é transferida entre dispositivos e é assunto de *standards* internacionais aprovados pela *International Standard Organization* (ISO). Esta organização aprovou-o como barramento *standard* para redes de elevada velocidade (≥125 Kbit/s) na partilha de informação em tempo real em veículos automóveis [4] e para taxas de transmissão menores (≤ 125 Kbit/s) [24], estando em conformidade com o modelo de referência de sete camadas *Open Systems Interconnection* (OSI) para os dois níveis inferiores do mesmo: o nível de ligação de dados (nível 2) e o nível físico (nível 1).

Com a ISO, o CAN foi adoptado pela indústria automóvel bem como por outros tipos de indústrias, devido à sua robustez e flexibilidade. A disponibilidade de circuitos integrados colocados no mercado por vários fabricantes, de semicondutores, encoraja a sua utilização devido ao seu baixo custo.

3.2.2 Características do CAN

O CAN é um protocolo de comunicações série, que permite controlo distribuído em tempo real, com elevado nível de segurança [4]. É um sistema em barramento com capacidades multi-mestre, isto é, vários nós podem pedir acesso ao meio de transmissão em simultâneo. Este protocolo comporta também o conceito de *multicast*, isto é, permite que uma mensagem seja transmitida a um conjunto de receptores simultaneamente.

Nas redes CAN não existe o endereçamento dos destinatários no sentido convencional, em vez disso são transmitidas mensagens que possuem um determinado identificador. Assim, um emissor envia uma mensagem a todos os nós CAN e cada um por seu lado decide, com base no identificador recebido, se deve ou não processar a mensagem. O identificador determina também a prioridade intrínseca da mensagem, ao competir com outras pelo acesso ao barramento.

O CAN é considerado um sistema de barramento série, bom para ligar em rede subsistemas inteligentes, tais como sensores e actuadores. A informação transmitida possui tamanho curto. Assim, cada mensagem CAN pode conter um máximo de 8 *bytes* de informação útil, sendo no entanto possível transmitir blocos maiores de dados recorrendo a segmentação.

A taxa máxima de transmissão especificada é de 1 Mbit/s, correspondendo este valor a sistemas com comprimento de barramento até 40 m [4]. Para distâncias superiores a taxa de transmissão, recomendada, diminui. Alguns dos valores recomendados [25] são: 50 Kbit/s para distâncias até 1 Km e 125 Kbit/s para distâncias até 500 m. Se a distância

do barramento for superior a 1 Km pode ser necessária a utilização de dispositivos repetidores (*repeater*) ou ponte (*bridge*).

O número de elementos num sistema CAN está, teoricamente, limitado pelo número possível de identificadores diferentes. Este número limite é no entanto significativamente reduzido por limitações físicas do *hardware*. Existem no mercado de integrados *transceivers* que permitem ligar pelo menos 110 nós [26]. Refira-se também, neste contexto, que com módulos de entrada/saída (E/S) adequados é possível ter diversos sensores e actuadores por nó.

O CAN permite flexibilidade uma vez que podem ser adicionados novos nós a uma rede CAN sem requerer alterações do *software* ou *hardware* dos restantes nós, se o novo nó não for emissor, ou se o novo nó não necessitar da transmissão de dados adicionais [4].

Outra característica importante é o facto de o controlador CAN de cada estação, registar os erros, avaliando-os estatisticamente, por forma a desencadear acções com eles relacionadas. Estas acções podem corresponder ao desligar, ou não, da estação que provoca os erros, tornando este protocolo eficaz em ambientes ruidosos.

Utilizando o protocolo CAN, a ligação entre o nível físico (nível 1 OSI) e o de aplicação (nível 7 OSI) é feita utilizando vários protocolos emergentes ou através de *software* desenvolvido pelo utilizador [22]. O exemplo de um protocolo, baseado em CAN, *standard* industrial ao nível de aplicação é o DEVICEnet™ da Allen-Bradley o qual é utilizado para ligar em rede controladores lógicos programáveis e sensores inteligentes.

Em resumo, o grande interesse pelo CAN por parte dos círculos da engenharia de automação industrial reside em diversas das suas características, nomeadamente as seguintes [27]:

- Ser um *standard* ISO;
- Considerável imunidade ao ruído;
- Capacidade multi-mestre;
- Capacidade *multicast*;
- Capacidade eficaz de detectar e sinalizar erros;
- Simplicidade;
- Retransmissão automática de mensagens “em espera” logo que o barramento esteja livre;
- Reduzido tempo de latência;
- Atribuição de prioridade às mensagens;
- Flexibilidade de configuração;
- Distinção entre erros temporários e erros permanentes dos nós;
- Elevadas taxas de transferência (1 Mbit/s);
- Redução de cabo a utilizar;
- Baixo preço;
- *Hardware standard*.

Nas próximas secções deste capítulo serão abordadas algumas regras que permitem ao protocolo CAN possuir as características anteriormente referidas e que o tornam actualmente num dos protocolos de comunicações com maior aceitação.

3.3 MÉTODO DE ENDEREÇAMENTO

Quando são transmitidos dados utilizando o CAN, não existe endereço fonte ou destino numa mensagem. Os identificadores únicos, das mensagens, servem para caracterizar o conteúdo da mensagem (ex. rpm ou temperatura do motor, no caso do controlo de veículos) sendo da competência de cada nó da rede decidir se a mensagem é ou não válida, para esse nó, realizando para isso um teste de aceitação ao identificador da mesma [28]. Este teste é designado por filtragem, "*Frame acceptance filtering*" [4], existindo vários dispositivos controladores que permitem diferentes níveis de sofisticação desta filtragem.

Outra característica importante do identificador, para além de definir o conteúdo da mensagem, é a de este estabelecer, também, a prioridade da mensagem. Isto é importante para a atribuição do barramento quando várias estações competem pelo acesso ao barramento.

O processo de transmissão e recepção de mensagens CAN, ilustrado na Figura 3.1, consiste no seguinte: se a unidade central de processamento, *Central Processing Unit* (CPU), de um dos nós desejar enviar uma mensagem para um ou mais nós da rede, este "passa" os dados a serem transmitidos e o respectivo identificador para o controlador CAN ("Preparar") desse nó. Isto é tudo o que o CPU necessita realizar para iniciar a transferência de informação. A mensagem é então composta e transmitida pelo controlador CAN. Logo que o controlador consiga acesso ao barramento ("Enviar mensagem") todos os outros nós na rede CAN tornam-se receptores da mesma ("Receber mensagem"). Cada estação na rede CAN, tendo recebido correctamente uma mensagem, realiza um teste de aceitação para determinar se os dados recebidos são ou não relevantes para essa estação ("Selecção"). Se os dados tiverem significado são processados ("Aceite"), caso contrário são rejeitados ("Não Aceite").

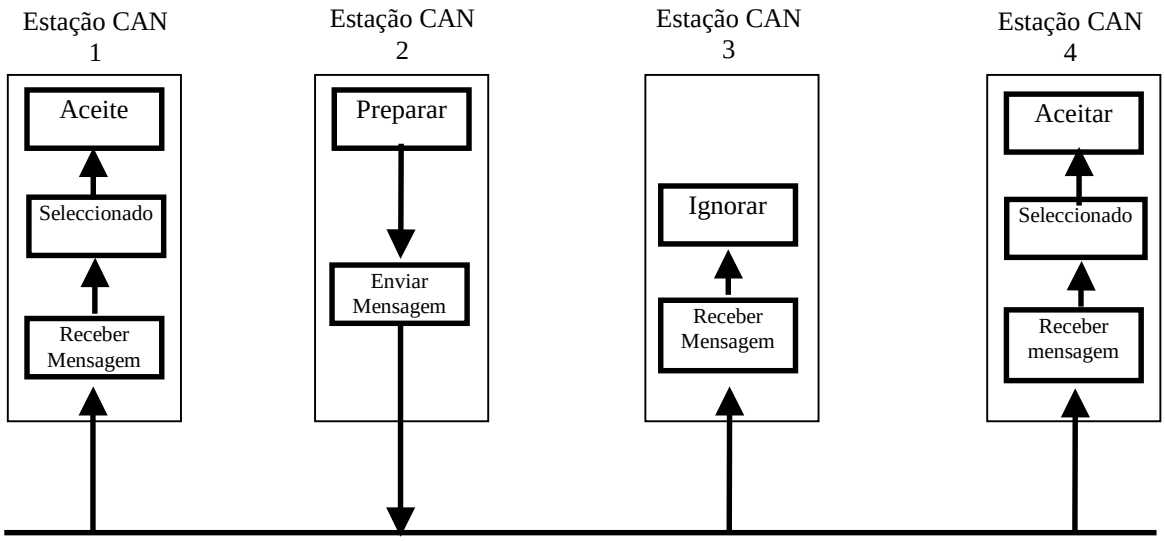


Figura 3.1 - Método de endereçamento.

É atingido, desta forma, um nível superior de flexibilidade do sistema e de configuração devido ao esquema de endereçamento orientado à contenção. É fácil adicionar estações

à rede CAN existente sem necessidade de alterações de *hardware* ou *software* dos nós existentes, desde que as novas estações sejam apenas receptoras. Uma vez que o protocolo de transmissão de dados não requer endereços físicos para os componentes individuais, suporta o conceito de electrónica modular e ao mesmo tempo permite recepção múltipla (*multicast*, *broadcast*) e sincronização de processos distribuídos. Medições, necessárias como informação por vários microcontroladores, podem ser transmitidas pela rede, de tal forma que não é necessário que cada microcontrolador possua o seu sensor, evitando redundância de sensores no caso de redes de aquisição.

3.4 PROCESSO DE ARBITRAGEM NÃO DESTRUTIVA

Para que os dados sejam processados em tempo real estes devem ser transferidos rapidamente. Isto exige um meio físico que permita elevada taxa de transmissão e chamadas rápidas à alocação do barramento quando várias estações tentam transmitir simultaneamente. No processamento em tempo real a urgência da troca de mensagens pela rede pode ser significativamente diferente: uma grandeza (ex. carga do motor) que varie rapidamente deve ser transmitida com maior frequência e menores atrasos do que outras que variem menos (ex. temperatura do motor).

No CAN, a prioridade com que uma mensagem é transmitida relativamente a outra é especificada pelo identificador da respectiva mensagem. A prioridade das mensagens é definida durante a fase de projecto do sistema sob a forma de valores binários. Para esta definição de prioridades é considerado que o identificador de menor valor numérico detém maior prioridade [22].

Por outro lado, a singularidade dos identificadores dos objectos de comunicação, mensagens, também é utilizada para arbitrar os pedidos de acesso ao barramento por parte dos nós concorrentes. O CAN é uma rede *Carrier Sense Multi-Acess with Determinstic Collision Resolution* (CSMA/DCR) [29], ou seja: os nós atrasam a transmissão se o barramento estiver ocupado; quando a condição de barramento livre for detectada, qualquer nó pode iniciar a transmissão; os conflitos de acesso ao barramento são solucionados por comparação orientada ao bit, *bitwise*, dos identificadores afectos aos objectos de comunicação, mensagens, e funciona da seguinte forma:

- Enquanto transmite o identificador do objecto de comunicação, cada nó monitora o barramento série;
- Se o *bit* transmitido for “recessivo”, nível lógico ‘1’, e for monitorizado um *bit* “dominante”, nível lógico ‘0’, o nó desiste da transmissão e inicia a recepção dos dados que chegam;
- O nó que transmite o objecto com o menor identificador, ganha acesso ao barramento e continua a transmissão.

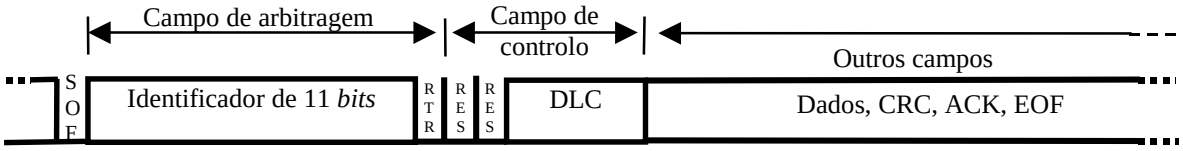
Isto significa que: a arbitragem é não destrutiva, uma vez que a transmissão do objecto de menor identificador não sofre atraso; o acesso ao barramento obedece a prioridades, permitindo que a informação mais urgente seja atendida em primeiro lugar. A retransmissão automática de um objecto de comunicação é tentada após uma perca no processo de arbitragem.

3.5 FORMATO DAS *FRAMES*

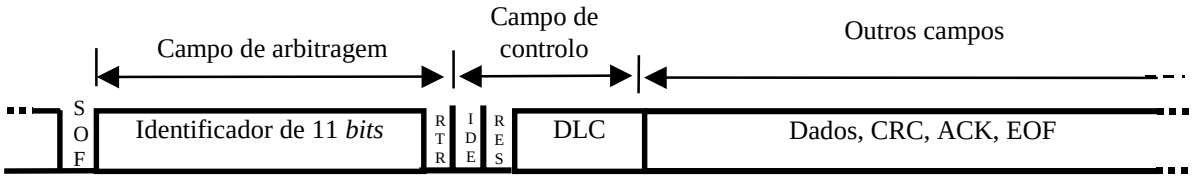
Originalmente as *frames* do protocolo CAN possuíam um identificador de 11 *bits*. Com a especificação 2.0 [27] foi definido um novo formato, o qual se baseia num identificador de 29 *bits* de comprimento, aumentando consideravelmente o número possível de identificadores únicos. Este novo formato é opcional, o que significa que as mensagens com identificadores de 11 *bits* continuarão a ser as mensagens *standard*. De acordo com a especificação 2.0 as *frames* com 11 *bits* identificadores designam-se por *frames standard* e as *frames* com 29 *bits* identificadores designam-se por *frames* estendidas.

Seguidamente serão abordadas as diferenças, mais significativas, entre os diversos formatos actualmente conhecidos [30].

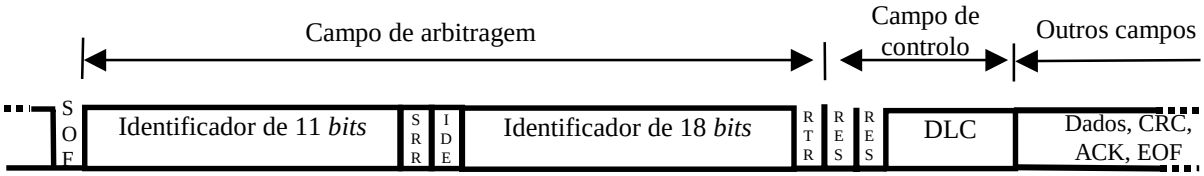
A Figura 3.2 ilustra os diferentes tipos de formato possíveis, onde se pode verificar que em todos os formatos, as mensagens começam com o *bit* de início de *frame*, *start* of *frame* (SOF), seguido dos *bits* do identificador. Com o identificador existem um ou três *bits* de controlo (RTR, SRR e IDE), no campo de arbitragem. Estes *bits* definem quando se trata de uma *frame standard* ou estendida e quando se trata de uma *frame* de dados ou remota.



(a) Formato duma *frame* de dados segundo as versões 1.0, 1.1, 1.2 e 2.0A.



(b) Formato duma *frame* de dados segundo a versão 2.0B (Formato *standard*).



(c) Formato duma *frame* de dados segundo a versão 2.0B (Formato estendido).

Figura 3.2 – Formato dos diferentes tipos de *frames.*

A Figura 3.2-a) ilustra uma *frame* de dados, de acordo com as especificações 1.0, 1.1, 1.2 e 2.0A do protocolo CAN. Totalmente compatível com aquele formato é o formato *standard* conforme é definido na especificação 2.0B, Figura 3.2-b).

Em contraste com o referido anteriormente, a Figura 3.2-c) representa o formato

estendido, definido na versão 2.0 B, o qual diverge dos formatos anteriores no número de *bits* do campo de arbitragem e valor dos *bits* de controlo.

O significado dos três *bits* de controlo é o seguinte:

- O *bit Remote Transmit Request* (RTR), distingue entre *frames* de dados e remotas. Em *frames* de dados este *bit* é "dominante", sendo "recessivo" em *frames* remotas.
- O *bit Substitute Remote Request* (SRR), é um *bit* "recessivo". Este *bit* é transmitido em *frames* no formato estendido na posição que o *bit* RTR ocupa em *frames standard*.
- O *bit Identifier Extension* (IDE), faz a distinção entre *frames standard* e estendidas. Em *frames standard* este *bit* é "dominante", enquanto que em *frames* estendidas é "recessivo".

Assim, num sistema onde vários nós iniciem a transmissão simultânea de *frames* com o mesmo identificador, as seguintes regras são aplicadas: *frames* de dados possuem maior prioridade do que *frames* remotas, e *frames standard* possuem maior prioridade do que *frames* estendidas. Isto significa que uma *frame standard* remota "vence", no processo de arbitragem, uma *frame* de dados estendida, se os 11 *bits* mais significativos do identificador forem iguais [30]. No entanto, em sistemas onde tenham que coexistir os dois formatos é necessário ter alguns cuidados em termos de implementação.

A maioria dos novos controladores respeita a versão 2.0. Assim para permitir *frames* estendidas um controlador pode ser: CAN 2.0B passivo ou CAN 2.0B activo. Se for passivo, ignora *frames* estendidas ao contrário dos controladores 1.0 que transmitem *frames* de erro quando detectam *frames* com identificadores de 29 *bits*. Se for activo, o controlador permite a transmissão e recepção de *frames* estendidas. Existem portanto regras de compatibilidade que devem ser consideradas para transmitir e receber os dois tipos de *frames*:

- Controladores CAN 2.0B activos, transmitem e recebem ambos os formatos de *frames* estendidas ou *standard*;
- Controladores 2.0B passivos, transmitem e recebem *frames standard*, ignorando *frames* estendidas sem erros;
- Controladores CAN 1.0 geram erros quando detectam *frames* estendidas.

Assim, numa rede com controladores CAN 1.0 não podem existir mensagens com formato estendido obrigando a que todos os controladores utilizem o formato *standard*.

3.6 TRANSFERÊNCIA DE MENSAGENS

Conforme foi referido na secção anterior, as *frames* do protocolo CAN possuem dois formatos que diferem no tamanho do identificador:

- *Frames standard*, cujo identificador possui 11 *bits*;
- *Frames* estendidas, cujo identificador possui 29 *bits*.

Embora haja estes dois tipos de formato, só se irá considerar o formato de *frame standard* porque foi o utilizado nesta dissertação. A transmissão e recepção de informação, num sistema CAN são efectuadas e controladas através de quatro tipos

diferentes de *frames*:

- *Frame* de Dados;
- *Frame* Remota;
- *Frame* de Erro;
- *Frame* de Sobrecarga.

3.6.1 *Frame* de Dados

Uma *frame* de dados, ilustrado na Figura 3.3, contém os dados do emissor para o receptor. Os sete campos que compõem este tipo de *frames* são os seguintes:

	Início de *frame*	Campo de arbitragem	Campo de controlo	Campo de dados	Campo CRC	Campo ACK	Fim De *frame*	

Figura 3.3 - *Frame* de dados.

- Início de *frame*, SOF. Este *bit* marca o início da transmissão de dados ou remotas. É um *bit* "dominante". Todos os nós têm de se sincronizar, com a transição provocada pelo *SOF* do nó que iniciou primeiro a transmissão;
- Campo de arbitragem. Este campo depende do formato da *frame*, sendo no formato *standard*, constituído pelo identificador e pelo *bit* RTR. O *bit* RTR tem o valor "dominante" para *frames* de dados;
- Campo de controlo, o qual é composto por seis *bits*, sendo 2 *bits* reservados para futura expansão seguidos pelo campo *Data Length Code* (DLC). Os *bits* reservados são transmitidos como "dominante". O campo DLC possui 4 *bits* que indicam o número de *bytes* do campo de dados;
- Campo de dados. Este campo indica o número admissível de *bytes* de dados para uma *frame* de dados varia num intervalo que pode ir de 0 a 8 [4, 27];
- Campo *Cyclic Redundancy Code* (CRC). Contém a sequência CRC seguida do delimitador CRC, o qual consiste num único *bit* "recessivo". O cálculo da sequência CRC é feito tendo em conta uma polinomial geradora [4, 27];
- Campo *Acknowledge* (ACK), reconhecimento. Este campo é constituído por dois *bits*, o *slot* ACK e o delimitador ACK, sendo este ultimo "recessivo". Os nós receptores ao receberem correctamente a sequência CRC, enviam o reconhecimento substituindo o *bit* "recessivo" por um *bit* "dominante" na *slot* ACK. Como consequência do anteriormente exposto note-se que o *slot* ACK é rodeado por dois *bits* 'recessivos';
- Fim de *frame*, *end of frame* (EOF). Este campo serve de *flag* delimitadora das *frames* de dados ou remotas, sendo constituído por sete *bits* "recessivos".

3.6.2 *Frame* Remota

Um nó que seja receptor de determinados dados, pode iniciar a transmissão dos mesmos, através de pedido ao nó de origem, enviando uma *frame* remota, ou seja, um pedido de dados. A *frame* remota é constituída por seis campos, conforme ilustrado na Figura 3.4.

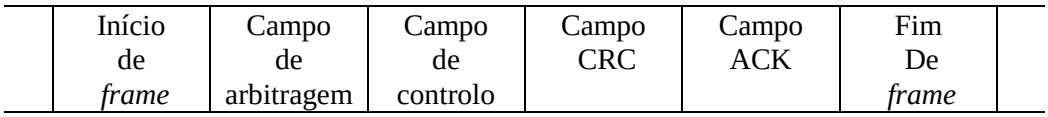


Figura 3.4 - *Frame* remota.

Os campos constituintes duma *frame* remota são idênticos aos duma *frame* de dados, com a excepção do valor do *bit* RTR, do campo de arbitragem, que agora é "recessivo" e da inexistência de campo de dados. Os *bits* DLC do campo de controlo da *frame* remota devem possuir valor igual ao da *frame* de dados correspondente.

3.6.3 *Frame* de Erro

A *frame* de erro, cujo formato está ilustrado na Figura 3.5, [7], é transmitida por qualquer nó quando é detectado um erro no barramento e é constituída por dois campos distintos. O primeiro campo é dado pela sobreposição de *flags* de erro provenientes de diferentes estações. O segundo campo é o delimitador de erro.

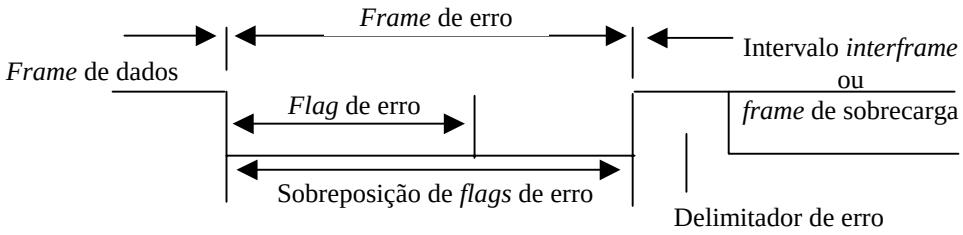


Figura 3.5 – *Frame* de erro.

Um nó "activo-ao-erro" ao detectar um erro assinala-o pela transmissão de uma *flag* "activa-ao-erro". O formato desta *flag* de erro viola a lei da inserção de *bits* aplicada a todos os campos desde o SOF até ao delimitador CRC. Como consequência, todos os outros nós detectam também uma condição de erro e iniciam por si a transmissão de uma *flag* de erro. A sequência de *bits* "dominantes", monitorizada no barramento, resulta da sobreposição das diferentes *flags* de erro transmitidas por nós individuais. O comprimento total desta sequência varia entre um mínimo de seis e um máximo de doze *bits*.

Uma estação 'passiva-ao-erro' detectando uma condição de erro tenta sinalizar este facto através da transmissão de uma *flag* "passiva-ao-erro". A estação "passiva-ao-erro"

tem que aguardar sempre por seis *bits* subsequentes iguais após detectar uma *flag* "passiva-ao-erro". Esta *flag* passiva ao erro é concluída quando forem detectados estes 6 *bits* iguais.

O segundo campo é o delimitador de erro, que consiste em oito *bits* "recessivos".

Após transmissão de uma *flag* de erro, cada nó envia *bits* "recessivos" e monitoriza o barramento até detectar um *bit* "recessivo". Após o que inicia a transmissão de mais sete *bits* "recessivos".

3.6.4 *Frame* de Sobrecarga

A *frame* de sobrecarga, *overload*, contem dois campos de *bits*: *flag* de sobrecarga e delimitador de sobrecarga. O formato da *flag* de sobrecarga corresponde à da *flag* activa-ao-erro. O delimitador de sobrecarga tem formato idêntico ao do delimitador de erro [4, 27].

A *frame* de sobrecarga, ilustrado na Figura 3.6, é utilizada para provocar um atraso extra entre uma *frame* de dados ou remota e a *frame* posterior [7].

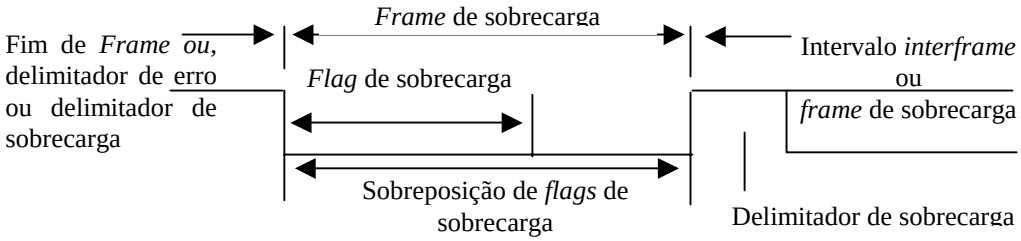


Figura 3.6 – *Frame* de sobrecarga.

A *flag* de sobrecarga consiste em seis *bits* "dominantes". Esta *flag* destrói o formato fixo do campo de Intermissão, pelo que todos os outros nós também detectam uma condição de sobrecarga e iniciam a transmissão de uma *flag* de sobrecarga. As condições que originam a transmissão de uma *flag* de sobrecarga são [27]:

- Condições internas de um receptor, que requer um atraso da próxima *frame* de dados ou remota. Nestas condições a *frame* de sobrecarga só pode ter início durante o primeiro *bit* de uma intermissão;
- A detecção de um *bit* "dominante" durante a intermissão. Neste caso a *frame* de sobrecarga tem início depois de ser detectado o *bit* "dominante".

Depois da transmissão de um *flag* de sobrecarga, qualquer nó "escuta" o barramento até detectar um *bit* "recessivo".

O delimitador de sobrecarga

O delimitador de sobrecarga é composto por oito *bits* "recessivos". Após a transmissão de uma *flag* de sobrecarga, todos os nós monitorizam o barramento até detectar uma transição de "dominante" para "recessivo". Nesse instante todos os nós terminaram o

envio da sua *flag* de sobrecarga e todos iniciam simultaneamente a transmissão de mais sete *bits* "recessivos" em simultâneo, para completar o delimitador de sobrecarga com o comprimento de oito *bits* [4].

3.6.5 Intervalo *interframe*

As *frames* de dados e as *frames* remotas são separadas de *frames* precedentes independentemente do tipo destas (dados, remota, erro ou sobrecarga), por um campo de *bits* designado por "intervalo *interframe", interframe space*.

Ao contrário das anteriores, as *frames* de erro e de sobrecarga não são precedidas por um intervalo *interframe*.

O intervalo *interframe* contém os campos de Intermissão, de barramento livre, *Idle*, e ainda o campo de Suspensão de Transmissão para nós "passivos-ao-erro" que tenham sido emissores de *frames* anteriores. A Figura 3.7 e a Figura 3.8 ilustram o intervalo *interframe* para estações que não sejam "passiva-ao-erro" ou que tenham sido receptoras da mensagem anterior e para estações "passivas-ao-erro" que tenham transmitido a mensagem anterior, respectivamente.

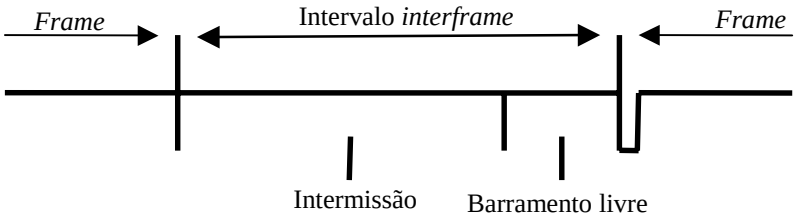
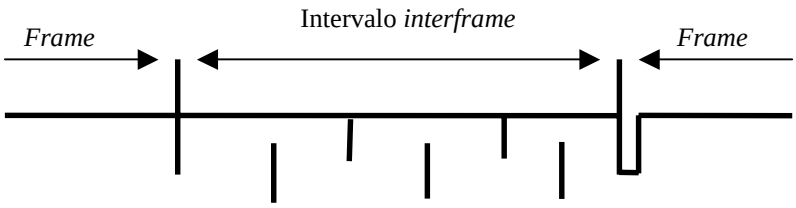


Figura 3.7- Intervalo *interframe* para estações que não sejam "passivas-ao-erro" ou que tenham sido receptoras da mensagem anterior.

O campo de intermissão consiste em três *bits* "recessivos". Durante a intermissão, não é permitido a nenhum nó iniciar a transmissão de uma *frame* de dados ou remota. A única acção que pode ter é assinalar uma condição de sobrecarga.

O período de barramento livre (*idle*) pode ter um tamanho arbitrário. Uma *frame* que esteja pendente para transmissão é iniciada no primeiro *bit* posterior à intermissão. A detecção de um *bit* "dominante" durante o estado livre do barramento é interpretada como início da *frame*, SOF.

A suspensão de transmissão verifica-se quando um nó "passivo-ao-erro" após transmitir uma *frame*, enviar oito *bits* "recessivos" a seguir à intermissão, antes de lhe ser permitido iniciar a transmissão da *frame* posterior ou reconhecer o barramento como livre. Se entretanto outro nó iniciar uma transmissão, o nó tornar-se-á um receptor dessa mensagem.



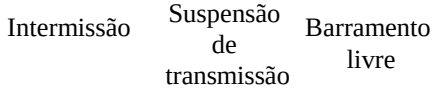


Figura 3.8 - Intervalo *interframe* para estações "passivas-ao-erro" que tenham transmitido a mensagem anterior.

3.7 CODIFICAÇÃO

Os campos das *frames* de dados e remotas (SOF, arbitragem, controlo, dados e sequência CRC), são codificados pelo método da inserção do *bit* (*bit stuffing*) de comprimento cinco. Assim, sempre que um emissor detecte cinco *bits* consecutivos, de igual valor, na série de *bits* a transmitir, ele insere automaticamente um *bit* complementar na sequência de *bits* a transmitir. Os restantes campos têm forma fixa e não são sujeitos a este método. As *frames* de erro e sobrecarga possuem forma fixa pelo que também não são codificadas pela regra da inserção do *bit*. Este processo é representado na Figura 3.9.

Sequências a transmitir	**100000abc**	**011111abc**
Sequências transmitidas	**1000001abc**	**0111110abc**
a, b, c ∈ { 0 , 1 }		

Figura 3.9 – Método de inserção de *bit.*

3.8 ORDEM DE TRANSMISSÃO DE *BITS* NUMA *FRAME* CAN

Uma *frame* é transferida campo a campo, iniciando no campo SOF. Dentro de um campo, o *bit* mais significativo é transmitido em primeiro lugar, conforme ilustrado na Figura 3.10, para uma *frame* de dados no formato *standard*.

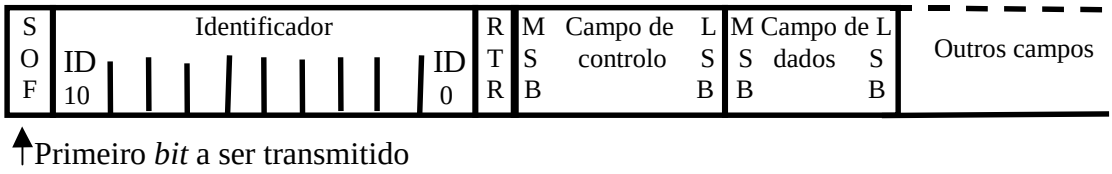


Figura 3.10 - Ordem de transmissão dos *bits*.

3.9 PRIORIDADE

Conforme foi referido na secção 3.4, as prioridades são definidas durante a fase de

projecto do sistema, sendo tanto maior a prioridade de determinada mensagem, quanto menor for o valor numérico do seu identificador.

Utilizando a especificação 2.0 B [27], o identificador é constituído por 29 *bits* o que significa que o número possível de identificadores diferentes (mais de 500 milhões) para as mensagens é consideravelmente superior ao número de identificadores permitidos se for utilizada a especificação 2.0 A (2032). Como consequência disto, o número de níveis de prioridades também é significativamente superior. Não existe, no entanto, na maioria dos casos, necessidade de utilizar o CAN 2.0 B, uma vez que 2032 níveis de prioridade são suficientes para a generalidade das aplicações. Utilizando o CAN 2.0B acaba por se desperdiçar largura de banda na maioria das aplicações.

Em termos de implementação existem controladores CAN que permitem apenas o formato 2.0A (Philips 80C200), enquanto que outros (Intel 82527) permitem os formatos CAN 2.0 A e CAN 2.0 B. A escolha do tipo de controlador a utilizar, deve ter em consideração, conforme foi referido na secção 3.5, a compatibilidade entre os diferentes formatos de *frame*.

3.10 TAXAS DE TRANSMISSÃO

No CAN a taxa de transmissão depende do comprimento de barramento e vice-versa. Esta limitação surge devido aos processos de arbitragem e *error-recovery*, o tempo de *bit* nunca deve ser inferior ao dobro do atraso de propagação no barramento. Segundo a especificação ISO 11989 [4], a taxa máxima corresponde a 1 Mbit/s para um tamanho de 40 metros. Para tamanhos de barramento maiores as recomendações segundo a CiA [25], CAN *in Automation*, são:

- 500 Kbit/s para distâncias até 100 metros;
- 250 Kbit/s para distâncias até 250 metros;
- 125 Kbit/s para distâncias até 500 metros ;
- 50 Kbit/s para distâncias até 1Km.

3.11 TAMANHO DUMA REDE CAN

O número de nós que podem existir numa única rede é, teoricamente, ilimitado [27]. No entanto, o número de identificadores e a capacidade dos *transceivers* existentes impõem restrições. Assim, dependendo do tipo de *transceiver*, até 32 ou 64 nós por rede é normal, existindo no entanto *transceivers* que permitem ligar pelo menos 110 nós por rede [26].

3.12 ARQUITECTURA DOS CONTROLADORES CAN

Existem duas versões de controladores CAN, uma com a designação de CAN básico, *Basic* CAN, e uma versão superior, possuindo um "filtro de aceitação" implementado em *hardware*, com a designação de CAN completo, *Full* CAN [22].

Na versão CAN básico, existe uma relação estreita entre o CPU e o controlador CAN pelo que todas as mensagens, que existam no barramento, são verificadas individualmente pelo microcontrolador. Isto resulta na ocupação do CPU para verificar todas as mensagens, em vez de processar apenas as relevantes, o que tende a limitar o *baud rate* a 250 Kbit/s. Por outro lado, na versão CAN completo, a introdução de um filtro de aceitação permite ao controlador recusar mensagens irrelevantes, através da verificação dos identificadores, libertando o CPU para processar apenas mensagens consideradas relevantes o que permite conseguir taxas de transmissão mais elevadas (1 Mbit/s).

3.13 MÉTODOS DE DETECÇÃO E SINALIZAÇÃO DE ERRO

Numa rede CAN há a garantia de que uma *frame* é simultaneamente aceite por todos os nós ou apenas por alguns. Assim a consistência de dados é, uma propriedade do sistema, alcançada através de conceitos de *multicast* e suporte de erros, *error handling* [4].

Para implementar o segundo conceito, estão definidos diversos tipos de erro e respectivos mecanismos de detecção.

Ao contrário de outros sistemas de barramento, o protocolo CAN não utiliza mensagens de confirmação mas detecta e assinala erros que ocorram [28].

3.13.1 Detecção de erros

Para a detecção de erros o protocolo CAN implementa três mecanismos ao nível da mensagem e dois ao nível do *bit*.

Esses mecanismos são [4]:

- Monitorização;
- Verificação da regra de inserção do *bit*;
- Verificação da *frame*;
- Verificação da sequência CRC de 15 *bits*;
- Verificação do reconhecimento (*ACK).*

Estes tipos de mecanismos, não mutuamente exclusivos, são utilizados para verificar a ocorrência dos seguintes tipos de erro:

3.13.1.1 ERROS AO NÍVEL DA MENSAGEM

Erro de CRC

O CRC salvaguarda a informação contida na *frame*, sendo adicionados pelo emissor, no fim de transmissão desta, *bits* redundantes. O receptor recalcula estes *bits* e compara-os com os *bits* recebidos. Se não forem coincidentes é assinalado um erro de CRC.

Erro de formato

Este tipo de erro é detectado quando um campo de formato fixo contiver um ou mais *bits* ilegais.

Erro de reconhecimento

Este tipo de erro é detectado quando o emissor não detecta um *bit* "dominante" durante o *slot* ACK.

3.13.1.2 ERROS AO NÍVEL DO *BIT*

Erro de *bit*

O emissor tem capacidade para detectar erros de *bit* baseado na monitorização dos sinais do barramento. Assim cada nó que transmite também monitora o barramento, detectando, caso existam, diferenças entre o valor do *bit* transmitido e o valor do *bit* monitorizado.

Erro de inserção de *bit*

O erro de inserção de *bit* é detectado sempre que existam seis *bits* consecutivos com nível lógico igual, num campo da *frame* que seja codificado pelo método da inserção do *bit*.

3.13.2 Sinalização de erros

Se for detectado um erro de *bit*, de inserção de *bit*, de formato ou no reconhecimento, por qualquer nó, este inicia a transmissão de uma *flag* de erro durante o *bit* seguinte.

Se um erro na CRC for detectado, é iniciada a transmissão de uma *frame* de erro no *bit* seguinte ao delimitador ACK, a menos que já tenha sido iniciada uma *frame* de erro para qualquer outra condição de erro [4].

3.14 O NÍVEL FÍSICO

O nível físico é responsável pela transferência de *bits* entre os diferentes nós de uma rede CAN; este define o modo como os sinais são transmitidos, lidando, por isso, com parâmetros como temporização, codificação e sincronização das sequências de *bits* a serem transmitidos [29]. Na especificação inicial feita por Robert Bosh [27], nenhum meio foi definido, permitindo diferentes opções para o meio de transmissão e níveis dos sinais. Estas propriedades foram posteriormente contempladas pelo *standard* ISO [4] onde estão definidas características dos sinais.

O *standard* DS 102-1 da CiA [25], completa aquelas definições relativamente a especificações do meio físico e conectores. A CiA especifica também diversas ligações mecânicas (cabos e conectores).

A rede CAN opera num modo quase estacionário: por cada transmissão de um *bit* é

dado tempo suficiente para estabilizar o nível do sinal antes que seja feita a amostragem quase simultânea por todos os nós. Isto significa que a capacidade do barramento é de um *bit*. Devido à mencionada estabilidade requerida, o comprimento máximo da rede depende da taxa de transmissão. Alguns valores típicos para as taxas de transmissão foram referidos na secção 3.10.

3.14.1 Meio físico

Com o CAN é possível utilizar diversos meios físicos, tais como: par de fios entrelaçados, fibra óptica, rádio frequência, etc. Actualmente, a maioria das aplicações utiliza um barramento diferencial a dois fios [22].

3.14.2 Transmissão diferencial

No CAN os sinais são transmitidos utilizando tensões diferenciais, derivando daí muita da imunidade ao ruído e tolerância a falhas que o caracterizam. As duas linhas de sinal são designadas por 'CAN_H' e 'CAN_L'. Um '0' corresponde ao sinal CAN_H superior ao CAN_L e como tal designado por *bit* "dominante". A situação contrária, CAN_L superior a CAN_H, corresponde a um *bit* "recessivo" ou '1' conforme ilustra a Figura 3.11.

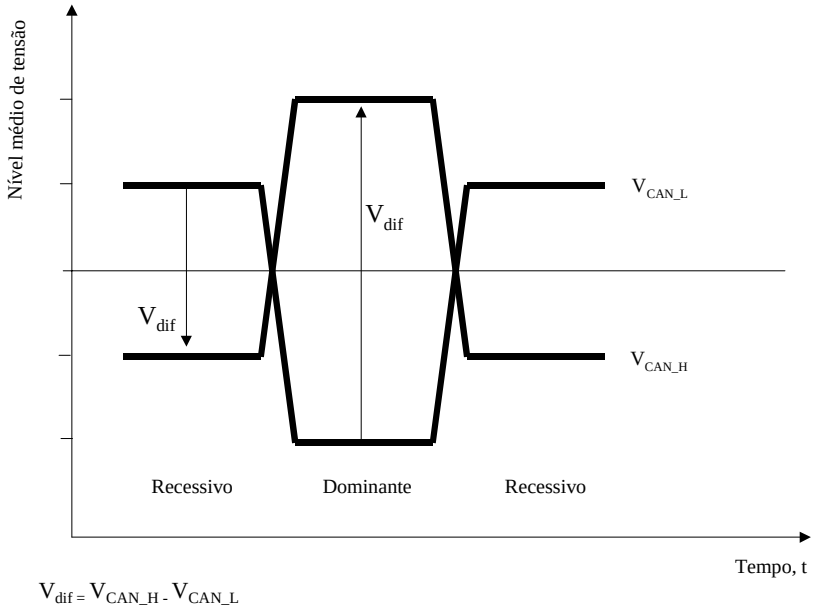


Figura 3.11 - Representação física de *bit.*

A utilização de diferenciais de tensão permite às redes CAN funcionar quando uma das linhas de sinal for danificada, ou em situações extremas de ruído. Recorrendo a um simples par entrelaçado, as entradas CAN diferenciais cancelam o ruído de forma efectiva, assegurando que estão compreendidas na gama de modo comum.

Existem interfaces económicas [26] disponíveis que fazem a translação dos níveis lógicos de 5 V para linha balanceada requerida pelo CAN e vice-versa.

Todos os *bits* são transmitidos de acordo com o método *Non-Return-to-Zero* (NRZ). Isto significa que o nível do *bit* é constante durante a sua duração, sendo "dominante" ou "recessivo". Este método apresenta uma densidade espectral baixa, possibilitando um bom aproveitamento da largura de banda de transmissão.

3.14.3 Tolerância a falhas no barramento

A especificação *standard* [4], define meios próprios para detecção de falhas, que permitem o funcionamento da rede em condições adversas, mesmo com uma relação sinal/ruído reduzida. A tolerância a falhas fundamenta-se na operação com apenas um fio e é capaz de garantir continuidade da operação da rede na presença de:

- Quebra de um fio;
- Curto circuito de um fio à massa, *ground*, ou à bateria de alimentação, V_{CC};
- Curto circuito entre as linhas CAN_L e CAN_H;

A especificação *standard* não prevê meios de tolerância no caso de anomalia simultânea nos dois fios do barramento.

3.15 TEMPO DE *BIT* E SINCRONIZAÇÃO

3.15.1 Representação e definição do tempo de *bit*

O tempo nominal do *bit*, t_B, é definido como sendo a duração de um *bit*, correspondendo ao inverso da taxa nominal de transmissão, número de *bit*/s transmitidos, de um emissor ideal na ausência de ressincronização [4] . Os segmentos que constituem o tempo de *bit* são ilustrados na Figura 3.12.

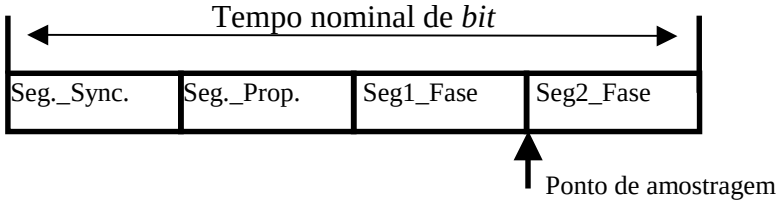


Figura 3.12 - Divisão do tempo de *bit*.

As funções de gestão do barramento executadas durante do tempo do *bit*, tais como o ajuste da sincronização de um nó, atraso de compensação da transmissão da rede e a posição dos pontos de amostragem, são definidas pela lógica programável do tempo de *bit* do controlador CAN [5, 7]. Os segmentos que constituem o tempo *bit* são:

- Segmento de Sincronização, Seg_Sync.. Este segmento é usado para sincronizar os várias nós do barramento, sendo esperada uma transição durante o mesmo;

- Segmento do Tempo de Propagação, Seg_Prop. Este segmento é utilizado para compensar os tempos de atraso físico na rede. Estes atrasos consistem nos tempos de propagação do sinal na linha do barramento e nos tempos de atraso interno aos nós;
- Segmentos 1 e 2 dos *buffers* de fase, Seg1_Fase e Seg2_ Fase. Estes segmentos são utilizados para compensar erros de fase nas transições. Estes segmentos podem ser alongados ou encurtados por ressincronização;
- Ponto de amostragem é o instante no qual é lido e interpretado o nível do barramento como sendo o valor do respectivo *bit*. Ocorre no final do segmento Seg1_Fase.

O tempo de processamento de informação é o segmento de tempo que tem início no ponto de amostragem e é reservado para cálculos do nível de *bit* subsequente.

3.15.2 Sincronização

Existem duas formas de sincronização:

- Sincronização forçada (*Hard synchronization*), no início de uma mensagem;
- Ressincronização, durante a transferência da mensagem.

As regras que regem os processos de sincronização são as seguintes:

- Somente é permitida uma sincronização durante um tempo de *bit*;
- Uma transição só pode ser usada para sincronização se o valor detectado no ponto de amostragem anterior, diferir do valor do barramento imediatamente após a transição;
- A sincronização forçada é realizada durante o estado de barramento livre, sempre que haja uma transição de “recessivo” para “dominante”.

A ressincronização consiste em produzir um alongamento ou estreitar do tempo de *bit*, por forma a permitir que a localização do ponto de amostragem seja correcta.

3.16 CONFIGURAÇÕES BASE DE CIRCUITOS INTEGRADOS

Existem três configurações base no que diz respeito a utilização de integrados CAN [21]:

- Controlador CAN discreto (*Stand alone*);
- Microcontrolador com controlador CAN interno;
- *Serial Linked Input Output (*SLIO).

3.16.1 Configuração utilizando controladores CAN discretos

No primeiro caso, o controlador CAN é ligado ao microcontrolador através dos barramentos de dados e endereços. Esta configuração é útil quando se pretende actualizar sistemas já existentes. Implica que seja possível efectuar a interface do controlador, ou seja, não pode ser utilizado um controlador apenas com E/S.

3.16.2 Configuração utilizando microcontrolador com controlador CAN

Se um sistema for desenvolvido de raiz, a segunda opção deve ser utilizada relativamente à primeira. Nesse caso, a interface de dados e endereços é feito internamente existindo vários destes dispositivos disponíveis no mercado.

3.16.3 Configuração SLIO CAN

O CAN, apesar das vantagens que permite, continua a ser considerado caro em alguns tipos de aplicações. Além disso, não é justificável utilizar microcontroladores para simples operações liga/desliga, *on/off*, tendo sido introduzida para esses casos, por alguns fabricantes, uma versão simples e barata de dispositivos designados vulgarmente por SLIO CAN. Trata-se de uma forma simples de reforçar a capacidade de E/S de um microcontrolador central através do barramento CAN. Os integrados SLIO detêm pouca "inteligência" necessitando de serem programados e calibrados por um microcontrolador, por eles responsável, funcionando sempre como "satélites" desse nó "inteligente" possuindo microprocessador.

Existe também a possibilidade de ter, no mesmo barramento, os três tipos de configurações anteriormente referidas. Mesmo tratando-se de integrados de diferentes fabricantes, com *software* desenvolvido de forma adequada, é possível interligar esses dispositivos.

Seguidamente, nesta secção, será justificada a utilização da tecnologia SLIO CAN em sistemas de controlo de reduzida velocidade (<125 Kbit/s), uma vez que foi esta a configuração base utilizada no âmbito desta dissertação.

Aplicações, como o caso de controlo de estufas, que envolvem, principalmente, operações liga/desliga de diversos dispositivos actuadores (lâmpadas, motores, electroválvulas, etc.) requerem tráfego reduzido de informação. Para aplicações deste tipo pode tornar-se caro e injustificado utilizar um barramento de dados de alta velocidade para informação de controlo, sendo por isso recomendada a utilização de um barramento de dados de menor velocidade.

Existem assim, duas configurações possíveis, sendo uma vantajosa para aplicações que exijam elevada velocidade e outra para aplicações de menor velocidade de transferência de informação. Do ponto de vista de engenharia é fácil fazer a passagem de elevada para baixa velocidade e vice-versa, se o mesmo protocolo for utilizado. No entanto, quando um sistema CAN é configurado para velocidades reduzidas, geralmente permanece o mesmo conjunto de circuitos integrados CAN. Isto não é económico uma vez que existe um desperdício de largura de banda de transmissão e recursos, utilizando a configuração composta por controladores CAN de elevada velocidade em aplicações de controlo que requerem velocidade reduzida, as quais, na maioria dos casos, envolvem funções simples de liga/desliga. Em resumo, o conjunto de circuitos integrados utilizado para aplicações de reduzida velocidade teria um custo igual ao gasto para aplicações de velocidade elevada.

O SLIO CAN foi desenvolvido para tirar partido da tecnologia CAN em aplicações de baixa velocidade com menor custo. Tipicamente, um integrado SLIO CAN custa cerca

de 1/5 do preço de um microcontrolador com interface CAN.

SLIO (*Serial Link Input/Output*)

Na sua forma mais simples, o SLIO pode ser visto como um porto E/S com controlador CAN, respeitando a especificação 2.0 A e 2.0 B (passivo). Existem SLIOs que permitem uma taxa de transmissão máxima de 125 Kbit/s, utilizando um oscilador interno, sendo no entanto possível operar a 250 Kbit/s utilizando para tal um oscilador externo a cristal, o que aumenta no entanto o preço e complexidade da interface.

As Figuras 3.13 e 3.14 ilustram configurações típicas de sistemas CAN e SLIO CAN utilizadas em aplicações de alta e baixa velocidade, respectivamente.

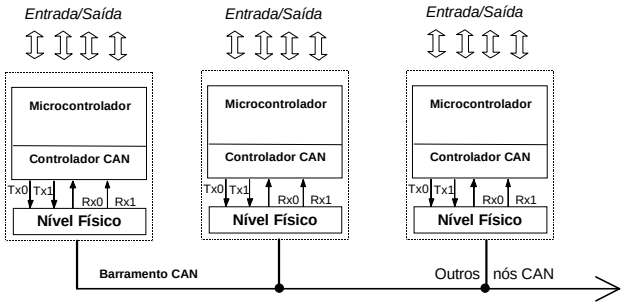


Figura 3.13 – Sistema CAN típico.

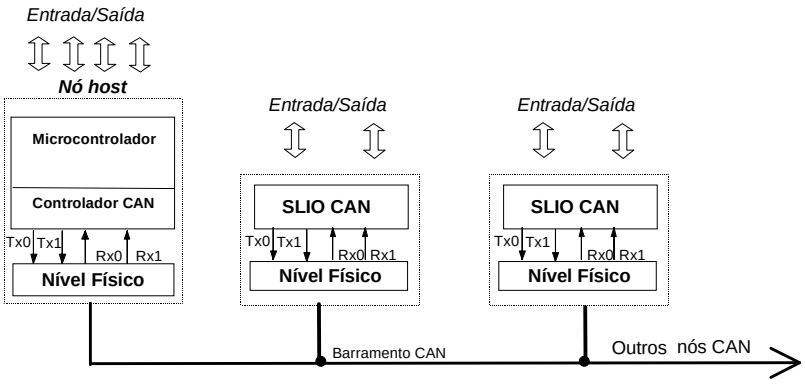
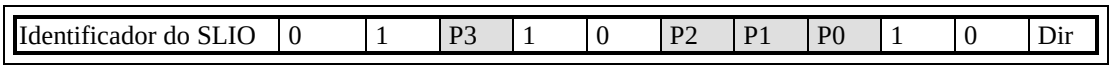


Figura 3.14 – Sistema SLIO CAN.

A diferença fundamental entre as duas configurações é a ausência do "dispendioso" microcontrolador nos nós SLIO CAN.

Conforme foi referido anteriormente, os dispositivos SLIO CAN são programados e calibrados por um nó "inteligente", *host*, o qual consiste num nó possuindo microcontrolador. Um único nó *host* pode controlar um máximo de 16 SLIOs num barramento SLIO CAN, estando este número relacionado com a existência de 4 *bits* configuráveis no identificador em cada SLIO o que limita a 16 o número máximo de identificadores únicos possível, conforme é ilustrado na Figura 3.15.

Identificador *standard*	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0



P0, P1, P2 e P3: pinos para configuração do identificador do SLIO
Dir: *bit* de direcção para mensagens do SLIO CAN

Figura 3.15 – Relação entre o identificador do SLIO P82C150 e o identificador CAN *standard.*

Combinando dois fabricantes (Philips e National) de SLIOs, os quais possuem configurações diferentes para os identificadores (*bit* ID1, da Figura 3.15 diferente), é possível um máximo de 32 nós num sistema SLIO CAN [31]. Em ambos os casos o *bit* ID0 do identificador CAN, indica a direcção da transferência de dados, sendo '0', em mensagens transmitidas do nó *host* para o(s) SLIO(s), e ´1´ para *frames* com "sentido" oposto. O nó *host* pode fazer *polling* aos seus SLIOs transmitindo *frames* remotas para o sistema SLIO CAN verificando, periodicamente, a existência de todos os nós assegurando desta forma a integridade do sistema.

Outra característica importante na comunicação *host vs* nós SLIO CAN é a de o campo de dados do sistema SLIO CAN ser fixo. O primeiro *byte* do campo de dados funciona como registo de *status* e comando, correspondendo os restantes *bytes* de dados aos pinos de E/S do SLIO (16 *bits* E/S disponíveis no caso do PCA82C150 da Philips). Cada pino E/S pode ser programado individualmente.

Endereço físico do SLIO

Devido aos 4 *bits* configuráveis do identificador, o SLIO CAN evidência a técnica de endereço físico na qual a cada nó SLIO CAN é atribuído um identificador único por *hardware* durante o *reset*. Deixando de haver endereçamento funcional como no CAN em geral. Além disso, até certo nível actua como uma configuração mestre/escravo virtual. A relação SLIO-*host* está representada na Figura 3.16. Conforme foi referido anteriormente, todos os 16 SLIOs num barramento CAN devem ser controlados apenas por um *host*. No entanto, em algumas situações, podem ser divididos em grupos e controlados por vários *host*s, não podendo no entanto, o número de SLIOs exceder 16 (ou 32) no mesmo barramento. No ambiente *multi-hosts* apenas é necessário um dos *hosts* para efeitos de calibração [31].

Devido à técnica de *broadcast* do CAN, todos os outros nós CAN (além do *host* e dos SLIOs) podem receber mensagens transmitidas pelos SLIOs (Figura 3.16). Sendo por isso, necessário garantir, através do teste de aceitação das mensagens, que todos os outros nós inteligentes (excepto o *host*) não actuem sobre os dados uma vez que tal situação poderia provocar erros e "confusão" no sistema.

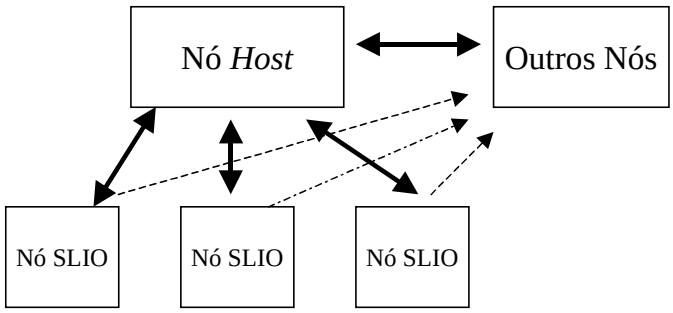


Figura 3.16 – SLIO e outros nós CAN partilhando o mesmo barramento.

Nas próximas secções serão consideradas aplicações utilizando o SLIO P82C150 da Philips uma vez que foi o utilizado nesta dissertação.

Iniciação do sistema SLIO CAN

Num sistema SLIO CAN, utilizando o SLIO P82C150, é necessário um nó *host* que envie uma mensagem de calibração com períodos de tempo iguais ou inferiores a 8000 tempos de *bit* para desta forma sincronizar o oscilador interno dos SLIOs com o tempo de *bit* do barramento [32]. Uma vez calibrados, os nós SLIO podem transmitir e receber mensagens como outros nós CAN. A Figura 3.17 ilustra o exemplo de uma mensagem de calibração [33].

SOF	**Campo de arbitragem**	**Campo de controlo**	***Byte* 1**	***Byte* 2**	**Campo CRC**
0	000 1010 1010 0	000|010	**10**10 1010	0000|0100	000|0 1011 1000 00|**0**

| - *bit* inserido pelo controlador CAN para garantir a regra de inserção de *bit*.

Figura 3.17 – Mensagem de calibração.

A transmissão pelo SLIO é efectuada automaticamente pelo controlador CAN, que o mesmo possui. Durante a iniciação, o SLIO é configurado para realizar funções tais como entrada *event capture*, saída digital ou conversão analógico-digital. A iniciação é feita pelo nó *host* programando os registos dos nós SLIO através do barramento CAN. Do ponto de vista da recepção, o SLIO possui um controlador CAN com uma lógica interna que lhe permite reconhecer automaticamente o identificador único do nó. A *frame* de reconhecimento é composta pelo *status* corrente e pelo valor dos registos do SLIO, permitindo desta forma ao nó *host* uma verificação da mensagem transmitida e do estado do(s) SLIO(s).

Quando é incorporado um novo SLIO numa rede CAN, o novo nó é capaz de se fazer conhecer ao controlador *host* após detectar um mínimo de 3 *frames* CAN, num espaço de 8000 tempos de *bit* [33]. O novo SLIO envia uma mensagem própria de "assinalar presença" confirmando ao *host* a sua presença na rede. O único requisito que o novo elemento deve preencher é ter um identificador diferente de qualquer outro dos nós

activos na rede CAN.

O tamanho da rede SLIO CAN

Uma vez que o SLIO CAN não possui oscilador a cristal, a lógica interna de tempo de *bit* do SLIO é optimizada para tolerância máxima do oscilador [31]. Isto encurta o tamanho efectivo do sistema CAN uma vez que o ponto de amostragem deve estar o mais próximo possível do início do tempo de *bit* o que limita significativamente o tempo permitido para o atraso de propagação na linha de transmissão, resultando em comprimentos de barramento mais reduzidos do que num sistema CAN convencional. Na Tabela 3.1 são comparados os comprimentos de barramento de um sistema CAN com os de um sistema SLIO CAN [31]. Refira-se no entanto que os comprimentos para um sistema SLIO CAN são suficientes para diversas aplicações, como o caso do controlo de estufas agrícolas.

Taxa de transmissão	P82C150 (SLIO CAN)	P8xC592/PCA82C200 (CAN)
125 Kbit/s	80 m	530 m
100 Kbit/s	120 m	620 m
50 Kbit/s	300 m	1300 m
20 Kbit/s	850 m	3300 m

Tabela 3.1– Distância máxima aconselhada, entre dois nós extremos, para uma rede CAN e SLIO CAN.

Futuro

Alguns autores [31] consideram que os semicondutores como o SLIO e circuitos de accionamento, de potência, podem ser incorporados em dispositivos "*plug-and-play*". Esta aproximação modular elimina a cablagem local possibilitando uma melhoria da eficácia do sistema total, uma vez que existem poucos fios, que podem falhar, sendo também mais robusto em termos de compatibilidade electromagnética. Esta visão do futuro é ilustrada pelas Figuras 3.18 e 3.19.

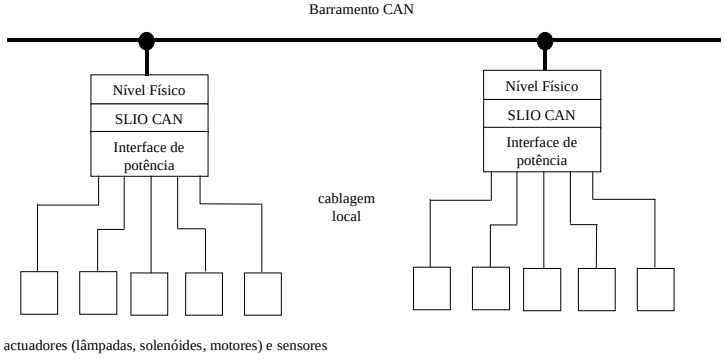


Figura 3.18 - Sistemas actuais.

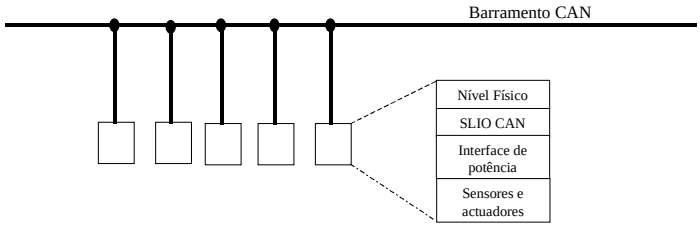


Figura 3.19 – Sistemas futuros.

3.17 PORQUE ESCOLHER O CAN?

O protocolo CAN, devido à sua fiabilidade e baixo custo, tem tido bastante aceitação, existindo actualmente vários dispositivos acessíveis no mercado de circuitos integrados.

Uma das vantagens do protocolo CAN é a capacidade de suportar o erro. Esta capacidade consiste em retransmitir a mensagem que não aparece correctamente no barramento CAN.

O meio de transmissão é simples, sendo o mais comum um par de fios. Um sistema CAN pode trabalhar apenas com um fio. Dependendo da aplicação, existem outros tipos de ligações que podem ser escolhidos, como a via rádio, óptica, etc.

Num sistema de comunicações, quando um nó do sistema falha existe a possibilidade de este nó bloquear todo o sistema, mas com o protocolo CAN isto não sucede porque esse nó pode ser excluído de enviar e receber no barramento CAN.

Utilizando um relógio global é simples implementar aplicações em tempo real, porque é possível sincronizar um sistema e esta sincronização é mantida durante a latência de uma mensagem.

Uma vez que todos os módulos “escutam” todas as mensagens, é fácil implementar aplicações do tipo *event driven.* Assim, um módulo pode ser programado para reagir logo que qualquer mensagem surja no barramento CAN.

O protocolo CAN é bom para implementar sistemas de controlo distribuído. O método de arbitragem para determinar a prioridade das mensagens e a possibilidade de um determinado número de nós tentar o acesso ao barramento (multi-mestre) possibilita a construção de bons sistemas de controlo.

Pode-se concluir que a aceitação do protocolo CAN por parte da indústria de automação se deve a vários factores, designadamente:

- Baixo custo;
- Capacidade de funcionar em ambientes com condições eléctricas adversas;
- Elevado grau de capacidades de tempo real e controlo distribuído;
- Fácil utilização;

- Disponibilidade de componentes, controladores, com o protocolo CAN;
- Existência de *standards*.