Local Interconnect Network

O LIN-Bus (Local Interconnect Network) é um padrão de comunicação veicular utilizada em arquitetura de redes inseridas em carros. A especificação do LIN é mantida pelo LIN-consortium, o qual teve sua versão 1.1, lançada em 1999.

O LIN bus é um sistema de rede pequeno e lento utilizado como uma opção barata para subredes CAN, utilizada para integrar sensores inteligentes ou atuadores em carros. Recentemente o LIN pode ser utilizado sobre a linha de força da bateria veicular com um transmissor especial, DC-LIN.

O protocolo LIN é composto por LIN-Mestre e LIN-Escravo. O LIN-Mestre utiliza um tabela de horários pré-definidos para enviar e receber dados no barramento LIN. As tabelas contém pelo menos o tempo relativo e onde a mensagem é iniciada. Um LIN Frame consiste de duas partes um cabeçalho e uma resposta. O cabeçalho sempre é enviado pelo mestre e a resposta sempre é enviada pelo escravo.

Os dados são transmistidos através do LIN serialmente, com oito bits de dados e com um bit de início e de parada, porém sem bit de paridade. A taxa de bit varia na faixa de 1kbaud até 20kbaud. Os dados no barramento são divididos em recessivos (logicamente altos) e dominantes (logicamente baixos). O tempo normal é considerado pela fonte estável do clock do LIN master, a menor entidade possivel é um Bit Time (52 µsec @ 19.2 kbaud).

O barramento pode atingir dois estados — Sleep-mode e ativo — utilizados pelo protocolo LIN. Enquanto o dado está no barramento todos os nodos LIN são requisitados para estarem no estado ativo. Depois de um timeout especificado, o nodo entra no Sleep-mode e será colocado em ativo quando receber um WAKEUP frame. Este frame pode ser enviado por qualquer nodo requisitando a utilização do barramento, o que pode acontecer tanto se o LIN Mestre seguir a programação interna, quanto se um dos LIN Escravos forem ativados por seu software interno. Depois de todos os nodos serem “acordados”, o Mestre continua enviando as mensagens programas.

A LIN API (Application Programmers Interface) provê um conjunto de chamadas de funções (que tem base na linguagem C) que devem ser implementada por cada LIN driver. Utilizando as rotinas pré-definidas do driver, todos as funções LIN podem ser acessadas.

A utilização da API facilita a implementação de padrões na criação dos drivers. E também aumenta a velocidade dos testes.

O Protocolo LIN

O protocolo LIN1 (do inglês *Local Interconnect Networks*) é um protocolo de comunicação serial projetado para controlar componentes eletrônicos mais simples do veículo (UPLAP et al., 2004), como por exemplo: vidro elétrico, retrovisor, travamento das portas,
etc. 1http://www.lin-subbus.org
As principais vantagens do protocolo LIN segundo (DENUTO et al., 2001) são:

- Padronização;
- Baixo custo;
- Interface com único fio de 12V;

- Sincronização própria sem necessidade de cristal;
- Tempo de latência conhecido;
- Velocidade de até 20 Kbit/s.

O protocolo LIN implementa o conceito de único-mestre/múltiplos-escravos (*singlemaster/multiple-slave*). Apenas o nó mestre poderá transmitir o cabeçalho da mensagem e somente um nó escravo deverá responder a este cabeçalho. O cabeçalho é composto pelo sinal de *break*, seguido pelo campo de sincronização e pelo campo do identificador. O escravo responde enviando o campo de dados.

O nó escravo executa apenas a tarefa escrava (*slave task*), mas o nó mestre executa a tarefa escrava e a tarefa mestre (*master task*). A tarefa mestra é responsável por gerar a tabela de escalonamento das mensagens e por produzir o cabeçalho do frame. A tarefa escrava é responsável por responder ao cabeçalho do frame e de repassar as mensagens recebidas para a camada de aplicação.

O protocolo LIN é baseado na interface serial UART (*Universal Asynchronous Receiver/Transmitter*), portanto a codificação do frame é formada por um bit de início (*start-bit*, nível baixo), 8 bits de mensagem codificados em *non-return-to-zero* (NRZ) e um bit de parada (*stop-bit*, nível alto). O LIN não utiliza o bit de paridade, uma vez que este bit é opcional para o frame UART, portanto são utilizados 10 bits para codificar um byte de dados.

Uma incompatibilidade com o padrão UART existe na transmissão símbolo de *break*, o protocolo LIN utiliza um sinal de *break* com no mínimo 13 bits. Existem duas técnicas que tornam possíveis a um microcontrolador com uma UART padrão conseguir transmistir 13 ou mais bits para simbolizar um *break* do LIN.

A primeira consiste em: 1) diminuir 30% o *bitrate* da UART em relação ao *bitrate* usado no LIN; 2) transmitir os 10 bits do símbolo *break*, uma vez que o *bitrate* está mais lento estes 10 bits corresponderão ao tempo de bit dos 13 bits do LIN; 3) reconfigurar a UART para transmitir o restante do frame no bitrate padrão do LIN. A desvantagem dessa técnica é a sobrecarga de processamento no microcontrolador, pois para cada mensagem transmitida é necessário repetir este processo.

A segunda técnica consiste em manter o bit que configura a transmissão do símbolo de *break* ativado por mais tempo que o necessário, isto acarretará a transmissão de duas sequências de 10 bits, ou seja, 20 bits de break. A desvantagem dessa técnica é que será desperdiçada mais largura de banda transmitindo uma quantidade maior de bits break.

Este sinal de break juntamente com o campo de sincronismo são usados para que os nós escravos do barramento LIN possam sincronizar seus relógios. Isto permite que sejam utilizados microcontroladores de baixo custo que utilizam apenas o oscilador interno (RC - resistor/capacitor). Somente o modo mestre precisará de um relógio mais preciso, utilizando cristal de quartzo ou ressonador cerâmico, para realizar a sincronização dos nós escravos.

O fato do protocolo LIN ser simples, permitir a utilização de microcontroladores de baixo custo e utilizar apenas um fio para transmissão de dado torna-o muito atrativo para a indústria automotiva. Porém o protocolo vem

sofrendo modificações para simplificar a integração de novos componentes nos veículos de forma rápida e transparente.

De fato a especificação LIN teve um aumento drástico de complexidade na segunda maior revisão (Revisão 2.0) (RUFF, 2003). Nesta nova versão foram incorporadas características que tornaram o LIN mais flexível, porém a implementação do sistema tornou-se mais complexa. Foram adicionados recursos como: detecção automática de bit-rate, diagnóstico, suporte a mensagens esporádicas, entre outros.

Um nó mestre LIN 2.0 é compatível com escravos LIN 1.3, porém o oposto não é verdadeiro, ou seja, não é possível a um mestre LIN 1.3 comunicar-se com escravos LIN 2.0. Neste trabalho será apresentado as características do LIN baseando-se na última especificação disponível, revisão 2.1.

- Formato do frame LIN

O frame padrão do protocolo LIN é apresentado na Figura 3.

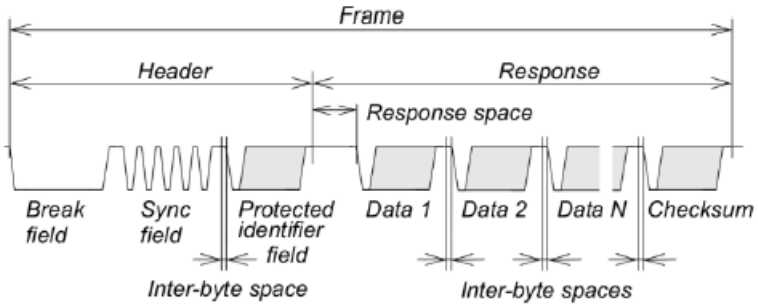

Figura 3: Frame de Protocolo LIN

O frame do LIN é formado pelo subframe de Cabeçalho e pelo subframe de Resposta, sendo o primeiro construído pelo nó mestre e o segundo pelo nó escravo.

O Cabeçalho do LIN é composto pelos campos *Break*, *Sync* e *Identificador Protegido*. O campo Break é identificado por 13 bits dominantes (nível lógico 0) e um bit delimitador recessivo (nível lógico 1). O campo Sync é formado por um byte 0x55 (além do *start-bit* e *stop-bit*). O campo Identificador Protegido é formado por dois subcampos: Identificador do Frame e Paridade.

O Identificador do Frame é formado por 6 bits, portanto valores de 0 a 63 podem ser usados. Já o subcampo Paridade é formado por 2 bits (P0 e P1).

Os identificadores do frame são divididos em três categorias:

- Valores de 0 a 59 (0x3B) são usados para frames carregando sinais;

- 60 (0x3C) e 61 (0x3D) são usados para carregar dados de diagnóstico e configuração;

- 62 (0x3E) e 63 (0x3F) são reservados para uso em futuras aplicações do protocolo.

As paridades são calculadas a partir dos bits do Identificador do Frame, segundo as equações:

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \quad (1)$$
$$P1 = \neg(ID1 \oplus ID3 \oplus ID4 \oplus ID5) \quad (2)$$

O subframe de Resposta é composto de 1 a 8 bytes de dados seguido pelo campo Checksum. Para entidades representadas por mais de um byte, o byte menos significativo é enviado primeiro (*little-endian*). Os campos de dados são nomeados Data1, Data2, ...até Data8.

O campo Checksum é gerado pelo resto da divisão, da soma de todos os valores, por 256. O checksum é calculado sobre os bytes de dados e sobre o campo Identificador Protegido no caso do LIN 2.x ou apenas sobre os bytes de dados para o LIN 1.x. No primeiro caso o checksum é chamado de checksum melhorado e no segundo de checksum clássico.

- Tipos de Frames

Alguns tipos de frames são apenas para propósitos específicos, como será visto nas subseções abaixo. Nem todos os tipos de frames serão suportados por determinados nós, isto depende da função deste nó.

- *Frame incondicional*

Os frames incondicionais carregam sinais e seus identificadores de frame podem assumir os valores entre 0 e 59 (0x3B). Neste tipo de frame é utilizado o modelo publicador/assinante (*publisher/subscriber*). O publicador de um frame incondicional (tarefa escrava) sempre deverá responder ao cabeçalho produzido pela tarefa mestre.

Todos os assinantes de um frame incondicional deverão receber a mensagem publicada e repassá-la para a aplicação. As formas como estas comunicação podem ocorrer entre os nós escravos e o nó mestre são apresentadas na Figura 4.

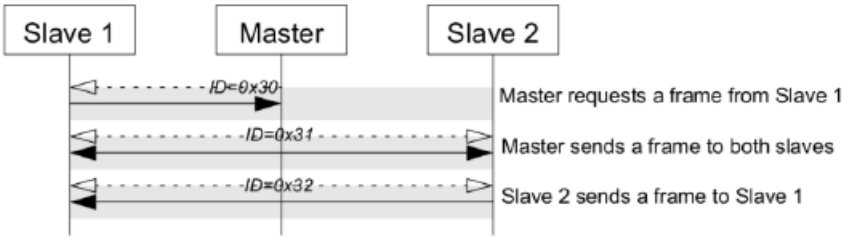

Figura 4: Três transferências de dados via frame incondicional

- *Frame Event Triggered*

O frame Event-Triggered foi adicionado ao LIN para permitir que nós escravos respondam apenas quando os mesmos possuírem dados disponíveis. Um frame Event-Triggered é associado aos sinais definidos para um ou mais

Frame Incondicional, por exemplo, um frame Event-Triggered com ID=0x12 pode estar associado ao Frame Incondicional ID=0x13 do nó escravo 1 e ao Frame Incondicional ID=0x14 do nó escravo 2.

Neste exemplo citado, sempre que o mestre enviar o frame Event-Triggered com ID=0x12 os nós 1 ou 2 poderão responder caso algum deles tenha um dado atualizado para enviar. Se o nó 1 não possuir dado para transmitir, mas o nó 2 possuir, ele irá criar um frame de Resposta com o primeiro byte de dados contendo seu ID (nó 2: ID=0x14) e os demais bytes contendo o dado do sinal disponível.

Se acontecer de dois ou mais nós responderem ao mesmo tempo a um frame Event-Triggered uma colisão será detectada pelo nó mestre. Neste caso o nó mestre consultará uma tabela de resolução de colisão para saber qual nó é mais prioritário. No exemplo citado, suponhamos que o nó 1 seja o mais prioritário, então o nó mestre enviará um frame Incondicional contendo o ID do nó 1 (0x13) para que o mesmo possa transmitir seu dado, em seguida enviará outro frame com o ID do nó 2.

O frame Event-Triggered evita que o nó mestre fique consultando cada nó individualmente para saber se eles têm dados atualizados para enviar. Isto melhora o tempo de resposta do sistema.

- *Frame Esporádico*

O objetivo do frame esporádico é adicionar um comportamento dinâmico no escalonamento do protocolo LIN, mas sem perder o determinismo do sistema como um todo.

Um frame Esporádico é associado a um grupo de frames Incondicionais e é transmitido pelo nó mestre, num slot destinado ao frame Esporádico, quando houver uma atualização de um sinal de um frame Incondicional do grupo. Caso sinais de diferentes frame Incondicionais sejam atualizados ao mesmo tempo, o nó mestre transmitirá o mais prioritário no slot de frame Esporádico atual e os demais serão transmitidos nos próximos frames Esporádicos ou quando seus frames Incondicionais forem escalonados. Normalmente os slots dos frames Esporádicos ficam vazios, uma vez que frames Esporádicos só são enviados quando ocorre uma atualização de um sinal fora da fase em que normalmente este é transmitido no frame Incondicional.

Apenas o nó mestre pode enviar frames Esporádicos, uma vez que apenas a tarefa mestre sabe quando um determinado frame Incondicional está pendente para transmissão.

- *Frame de Diagnóstico*

Os frames de diagnósticos possuem o Identificador de Frame igual a 60 (0x3C), chamados de frame de requisição do mestre, ou 61 (0x3D), chamados de frame de requisição do escravo. Um frame de diagnóstico sempre contém 8 bytes de dados.

Antes de transmitir um frame de requisição o nó mestre consulta o módulo de diagnóstico para saber se a mensagem de diagnóstico deverá ser transmitida ou se o barramento deverá ficar em silêncio. O nó escravo deverá enviar um frame de resposta incondicionalmente.

O frame de Diagnóstico utiliza o checksum clássico.

- *Frame Reservados*

Frames Reservados são identificados pelo Identificador de Frame igual a 62 (0x3E) e 63 (0x3F). Frames Reservados não devem ser usados numa rede LIN 2.x.