
بسم الله الرحمن الرحيم

سرشناسه	:	قربانعلی افجه، محسن، ۱۳۶۳ -
عنوان و نام پدیدآور	:	سیستم کنترل و مانیتورینگ قطار شهری/ محسن قربانعلی‌افجه.
مشخصات نشر	:	مشهد: مینوفر،۱۳۹۵.
مشخصات ظاهری	:	۱۸۶ص.: مصور ،جدول ،نمودار.
فروست	:	علوم پایه و مهندسی؛۴۱۱۴.
شابک	:	۱۶۰۰۰۰ ریال:۶-۹۵-۸۰۶۰-۶۰۰-۹۷۸
وضعیت فهرست نویسی	:	فیپا
یادداشت	:	کتابنامه
موضوع	:	راه‌آهن شهری
موضوع	:	Street-railroads
موضوع	:	سیستم‌های کنترل رقمی
موضوع	:	Digital control systems
موضوع	:	شبکه‌های کامپیوتری
موضوع	:	Computer networks
رده بندی کنگره	:	TFV۰۵/ق۴س۹ ۱۳۹۵
رده بندی دیویی	:	۳۸۸/۴۶
شماره کتابشناسی ملی	:	۴۳۳۲۹۱۰

سیستم کنترل و مانیتورینگ

قطار شهری

محسن قربانعلی‌افجه

انتشارات مینوفر
Minufar.ir

مشهد مقدّس - ۱۳۹۵

سیستم کنترل و مانیتورینگ قطار شهری

محسن قربانعلی‌افجه

طراح جلد: مینوفر

نوبت چاپ: اول - ۱۳۹۵

شمارگان: ۱۰۰۰ نسخه

قطع: وزیری

چاپ و صحافی: دقت

قیمت: ۱۶،۰۰۰ تومان

شابک: ۹۷۸-۶۰۰-۸۰۶۰-۹۵-۶

انتشارات مینوفر
Minufar Publications

مشهد مقدس، بلوار جلال‌آل‌احمد ۶۹ پ ۱۱۶، ص.پ: ۱۷۵۵-۹۱۸۹۵

همراه: ۰۹۳۵۲۱۶۲۷۵۵ - **تلفن و دورنگار**: ۳۸۳۲۳۵۵۳-۰۵۱

سایت: www.minufar.ir - **ایمیل**: minufar@yahoo.com

فهرست مطالب

فصل اول: معرفی استانداردهای سیستم کنترل و مانیتورینگ قطار ۹

مقدمه ۱۱

استاندارد TCN ۱۳

شرح وظایف هر یک از لایه‌ها در OSI ۱۴

کاربرد و نقش WTB ۱۷

پروتکل ارسال WTB ۱۸

مدهای کار HDLC ۱۸

فریم‌های HDLC ۱۹

روش منچستر کدینگ ۲۳

کاربرد و نقش MVB ۲۴

ارتباطات در حالت ESD با ایزولاسیون الکتریکی ۲۶

کلاس‌های تجهیزات دارای MVB ۳۲

فریم پروتکل MVB ۳۵

انواع فریم پروتکل MVB در حالت Slave و Master ۳۷

افزایش قابلیت اطمینان ۴۱

پروتکل MVB به‌طور خلاصه ۴۷

پروتکل MVB در محصولات Bombardier ۴۷

تفاوت‌های WTB و MVB: ۴۹

استاندارد LONWORKS ۵۱
معرفی اجزای فن‌آوری Lonworks ۵۲
پروتکل Lonworks ۵۷
پیاده سازی Lonworks در قطار ۶۰
مقایسه استانداردهای TCN و LONWORKS ۶۱
شبکه ETHERNET ۶۳
مقدمه ۶۳
تاریخچه Ethernet ۶۴
مشخصات استاندارد Base T۱۰ ..... ۶۴
مشخصات استاندارد Base T۱۰۰ ..... ۶۷
سیم بندی شبکه، Base T۱۰ Base T۱۰۰ ..... ۶۷
توپولوژی شبکه Base T۱۰ و Base T۱۰۰ ..... ۷۰
پیاده سازی لایه سخت‌افزاری Base T۱۰ و Base T۱۰۰: ۷۲
ساختار فریم اترنت ۷۳
پروتکل ارتباطی CAN ۷۷
مقدمه ۷۷
معرفی شبکه‌ی CAN ۷۸
حالات سیگنال الکتریکی در بأس CAN ۸۱
شبکه CAN فقط از دولایه‌ی مدل مرجع استفاده می‌کند ۸۴
مراحل تبادل داده‌ها ۸۵
فرمت فریم‌ها در شبکه CAN ۸۶
تخصیص بأس ۹۲
روش برقراری ارتباط بین تجهیزات و سیستم CAN ۹۲
پروتکل CANOPEN ۹۷
COB-ID ۱۰۰
مدل‌های ارتباطی ۱۰۱
پروتکل SDO (Service Data Object) ۱۰۳
خواندن مقدار منقطع CANopen_SDOC_Exp_Read ۱۰۵

نوشتن منقطع CANopen_SDOC_Exp_Write ۱۰۷
خواندن رشته بلند از دیکشنری CANopen_SDOC_Seg_Read ۱۰۹
نوشتن رشته بلند در دیکشنری CANopen_SDOC_Seg_Write ۱۱۲
پروتکل‌های مدیریت شبکه(NMT) ۱۱۵
پروتکل Heartbeat ۱۱۹
مثالی از هارت بیت ۱۲۱
بررسی توابع اصلی CANopen برای تراشه CXX۱۱LPX ۱۲۵
پوینتر به عنوان درایور در ROM ۱۲۵
توابع موجود در canopen_driver ۱۲۶
توابع درایور ۱۲۸
تابع تنظیمات اولیه CAN (init_can) ۱۲۹
تابع تنظیمات در دریافت پیام‌ها config_rxmsgobj ۱۳۲
تابع دریافت CAN ۱۳۴
تابع ارسال CAN ۱۳۴
تنظیمات CANopenCFG ۱۳۴
تابع فرآخوانی تمام توابع APIها (CALLBACKS) ۱۴۱
اطلاعات موجود در CAN_Node_Def.h ۱۴۲
تابع MAIN.C مستر ۱۴۵
تابع MAIN.C اسلیو ۱۵۵
معرفی تراشه ۱۱C۲۴LPC ۱۵۷
فصل دوم: ارتباط سیستم کنترل و مانیتورینگ با سیستم‌های سیگنالینگ ۱۶۹
مقدمه ۱۷۱
سیستم ATP ۱۷۲
سیستم ATO ۱۷۶
سیستم ATS ۱۷۸
طراحی ارتباط بین سیستم کنترل قطار و تجهیزات ON-BOARD سیستم سیگنالینگ. ۱۷۹
سیگنال‌های دیجیتال ورودی به سیستم ATC و خروجی TCMS ۱۸۱

سیگنال‌های دیجیتال خروجی از سیستم ATC و ورودی به TCMS ۱۸۳
سیگنال‌های آنالوگ خروجی از ATC (ورودی به TCMS) ۱۸۵

فصل اول

معرفی استانداردهای سیستم کنترل و مانیتورینگ قطار

مقدمه

آنچه در زیر مشاهده می‌فرمایید، نتیجه‌ی انجام مطالعات بر روی استانداردهای سیستم کنترل و مانیتورینگ است. دو استاندارد مختلف در دنیا در سیستم‌های کنترل و مونیتورینگ مورد توجه قرار گرفته است و سازندگان معتبر ریلی دنیا و تأمین کنندگان تجهیزات از این استانداردها پیروی می‌کنند.

اولین استاندارد، استاندارد TCN یا Train Communication Network می‌باشد که با همکاری IEC و UIC و گروه کاری ۲۲WG اولین پیش نویس این استاندارد در سال ۱۹۹۹ به تصویب رسید و به یک استاندارد بین‌الملل (IS) به شماره‌ی ۶۱۳۷۵IEC تبدیل شد و بسیاری از کارخانجات آمادگی خود را برای تطبیق با این استاندارد اعلام کردند. کارخانجاتی از قبیل:

ABB, EKE, ، Siemens ، Firema ،CAF ، Ansaldo، Alstom،Adtranz
...Unicontrol, Bombardier, MEDCOM

تا سال ۱۹۹۹، TCN در بیش از ۱۰۰۰ وسیله نقلیه ریلی دنیا پیاده سازی شده بود و این تعداد در مدت زمان کوتاهی بعد از تصویب استاندارد رشد زیادی کرد.

علاوه بر IEC و UIC، انجمن مهندسین برق و الکترونیک (IEEE) نیز در سال ۱۹۹۹ استاندارد ۱۷۷۳IEEE را به عنوان بستر انتقال اطلاعات در واگن‌ها معرفی کرد. این استاندارد شامل دو استاندارد L۱۴۷۳IEEE- و T۱۴۷۳IEEE- می‌باشد که به ترتیب پروتکل‌های Lonwork و TCN را به عنوان استاندارد جمع‌آوری و انتقال اطلاعات در واگن‌ها معرفی می‌کنند.

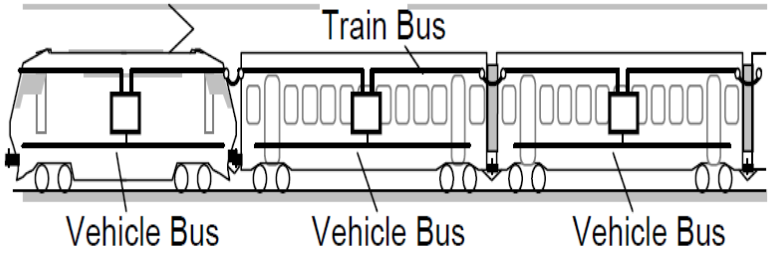
تاکنون بیش از ۳۰۰ شرکت اعم از سازندگان و تأمین کنندگان قطعات و شرکت‌های مسافری ریلی در دنیا به عضویت انجمن LonTalk که معرف پروتکل Lonworks برای استفاده در قطارها می‌باشند درآمده‌اند و تبعیت خود را از این استاندارد اعلام نموده‌اند. تعدادی از این شرکت‌ها عبارت‌اند از:

Alstom, Bombardier, Siemens, CAF, AnsaldoBreda, Kwasaki, Bach simpson, EKE Electronic Ltd, Echelon Corporation, Control Network Solutions, Curtis Door Systems,

با توجه به بررسی‌های انجام شده به نظر می‌رسد هر یک از استانداردهای TCN و Lonworks می‌تواند جهت پیاده سازی سیستم TCMS به کاربرده شود.

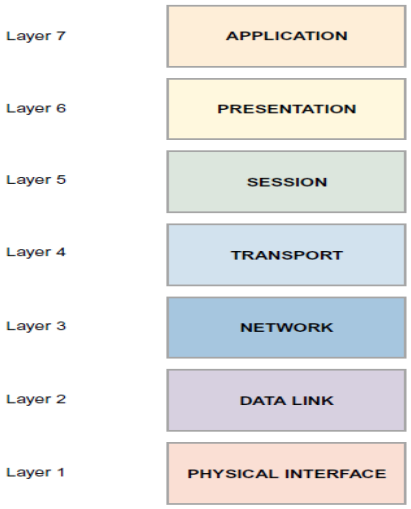
استاندارد TCN

این استاندارد یک شبکه سلسله مراتبی دو سطحی را برای تکمیل شبکه موردنیاز در قطار در نظر گرفته است. سطوح معماری TCN شامل ۲ بأس Vehicle_Bus در سطح واگن و Train_Bus در سطح قطار می‌باشد.



شکل ۱-۲-۱ نمای از ارتباطات بین Vehicle_Bus و Train_Bus

دو سطح معماری TCN از طریق NODEهای واقع در Train-bus به یکدیگر متصل می‌شوند. این گره‌ها نقش یک Gateway برای انتقال اطلاعات بین دو واگن را ایفا می‌کنند. شبکه TCN از مدل ۷ لایه‌ای (کاربردی، نمایش، نشست، انتقال، شبکه، پیوند داده‌ها، فیزیکی) (Open System Information) OSI پیروی می‌کند.



شکل ۱-۲-۲ لایه‌های هفت گانه مدل مرجع OSI

شرح وظایف هر یک از لایه‌ها در OSI

۱-لایه فیزیکی (Physical):

اولین لایه مدل بوده و در پایین‌ترین سطح قراردارد و به‌طور مستقیم با شبکه در ارتباط است. این لایه فقط در تبادل داده‌ها بین فرستنده و گیرنده نقش دارد.

۲-لایه پیوند داده‌ها (Data Link):

وظیفه این لایه این است که اطلاعات را برای ارسال آماده کند و درواقع اطلاعاتی را که از لایه بالاتر یعنی لایه شبکه دریافت می‌کند که به واحدهای کوچک‌تری تبدیل کرده و آنها را ارسال کند. همچنین این لایه وظیفه دارد که اطلاعات را برای ارسال صحیح و بدون خطا کنترل کرده و به فرستنده صحت

اطلاعات را اعلام کند، این لایه خود از دو زیرلایه به نام‌های LIC وMAC تشکیل شده است.

زیر لایه LIC وظایفی برعهده دارد که عبارت‌اند از: برقراری ارتباط نظیر به نظیر بین فرستنده و گیرنده، ایجاد قاب‌ها و کنترل خطاهایی که در اثر عوامل محیطی بر رسانه به وجود می‌آید. این زیرلایه عمل کنترل خطا را به این صورت انجام می‌دهد که هر قاب را ساخته و مرزهای ابتدا و انتهای آن را مشخص کرده سپس قاب‌ها را شماره‌گذاری و ارسال می‌کند. گیرنده قاب‌های ارسال شده را دریافت کرده و به ترتیب شماره، پشت سرهم قرار می‌دهد و اطلاعات را دوباره بازسازی می‌کند. زیرلایه LIC درگیرنده پس از دریافت هر قاب یک پاسخ برای فرستنده می‌فرستد. به این پاسخ Acknowledge گفته می‌شود. فرستنده اطلاعات با دریافت این پاسخ Acknowledge متوجه می‌شود که قاب مذکور به‌طور صحیح به مقصد رسیده است، فرستنده تا مدتی منتظر می‌ماند تا برای تمامی قاب‌های ارسال شده Acknowledge دریافت نماید. درصورتی‌که LIC برای قابی Acknowledge دریافت نکرد، متوجه می‌شود که قاب مذکور آسیب‌دیده و به مقصد نرسیده در این صورت قاب را دوباره ساخته و برای مقصد ارسال می‌کند. این زیرلایه با این روش سالم رسیدن اطلاعات به مقصد را تضمین می‌کند.

زیرلایه دیگری که در لایه پیوند داده‌ها قرار دارد، زیرلایه MAC است. یکی از وظایف آن کنترل دسترسی به خطوط انتقال و همچنین کنترل آدرس فیزیکی فرستنده و گیرنده است.

۳-لایه شبکه (Network):

یافتن تجهیزات مبدأ و مقصد و ایجاد یک مسیر ارتباطی بین مبدأ و مقصد وظیفه اصلی این لایه است.

۴-لایه انتقال (Transport):

وظیفه اصلی لایه انتقال، دریافت داده‌ها از لایه جلسه، در صورت نیاز، شکستن داده‌ها به واحدهای کوچک‌تر، انتقال آنها به لایه شبکه و حصول اطمینان از دریافت صحیح داده‌ها در انتهای دیگر (مقصد) است.

از وظایف دیگر لایه انتقال این است که این لایه باید مراقب برقراری و قطع اتصال در شبکه باشد. همچنین این لایه مکانیزمی برای کنترل جریان ارسال داده‌ها در اختیار دارد، به‌طوری‌که این مکانیزم سبب می‌شود فرستنده، داده‌ها را با سرعتی ارسال کند که گیرنده قادر به دریافت آنها باشد.

۵- لایه جلسه (Session):

ارسال معمولی داده‌ها را فراهم می‌کند اما خدمات پیشرفته‌ای را نیز ارائه می‌کند که کاربردهای مفیدی دارد. این لایه می‌تواند درحفظ نوبت کمک کند. یکی دیگر از خدمات مدیریت Token است. در بعضی از پروتکل‌ها لازم است هیچ‌کدام از طرفین کاری را هم‌زمان شروع نکنند برای مدیریت بر فعالیت‌های جلسه، Token هایی تهیه می‌شود که بین مبدأ و مقصد قابل مبادله‌اند در این شرایط فقط طرفی که Token را در اختیار دارد می‌تواند فعالیت کند و طرف مقابل باید منتظر باشد تا نوبت او برای استفاده از Token فرا برسد. یکی دیگر از اعمال لایه جلسه این است که روی قسمت‌هایی از رشته داده‌ها را

علامت‌گذاری می‌کند؛ درصورتی‌که بسته‌ای هنگام ارسال مفقود یا خراب شود این لایه بسته را از روی کدهای آن شناسایی و دوباره ارسال می‌کند.

۶-لایه نمایش (Presentation):

لایه نمایش داده‌ها ر ا به روش استاندارد کدگذاری می‌کند.

۷-لایه کاربردی (application):

این لایه حاوی پروتکل‌های گوناگون است که نرم‌افزارهای کاربردی برای ارتباط شبکه‌ای از آنها استفاده می‌کنند. این لایه می‌تواند ارتباط برنامه‌های مختلفی را که در محیط شبکه وجود دارند، با یکدیگر برقرار کند.

کاربرد و نقش WTB

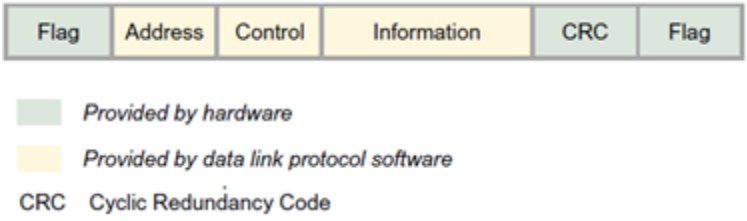
بالاترین سطح در شبکه سلسله مراتبی TCN، سطح Train_ bus می‌باشد که برای اتصال واگن‌های یک قطار به یکدیگر استفاده می‌شود، بأس استاندارد معرفی‌شده در IEC۱-۶۱۳۷۵، در سطح قطار WTB (Wire Train Bus) می‌باشد. بأس WTB با سرعت ۱ Mbit/S و پریود ۲۵ms می‌باشد و در لایه فیزیکی خود از یک زوج سیم به هم تابیده شیلد دار، کابل UIC استفاده می‌کند. با توجه به اینکه حداکثر تعداد گره‌های قابل اتصال به شبکه ۳۲ گره می‌باشد، با استفاده از این شبکه می‌توان حداکثر ۳۲ واگن را در حداکثر فاصله ۸۶۰ متر به هم وصل کرد.

موقعی که در قطار، واگنی جابجا می‌شود و یا واگنی به قطار متصل و یا از آن منفصل می‌شود، WTB به‌طور خودکار مجدداً همه واگن‌ها را متناسب با جایگاهشان در قطار آدرس‌دهی می‌کند به‌طوری‌که همه واگن‌ها دارای آدرس

می‌شوند و همچنین هر واگنی از آدرس سایر واگن‌های قطار مطلع می‌شود و جایگاه خود را در قطار می‌شناسد.

پروتکل ارسال WTB

پروتکل WTB به فرمت HDLC (High-Level Data Link Control) و نحوه کدینگ اطلاعات نیز به‌صورت منچستر کدینگ است. پروتکل HDLC برای حمایت از ارتباط تمام دوطرفه و نیمه دوطرفه بر روی پیوندهای ارتباطی نقطه‌به‌نقطه و چند نقطه‌ای طراحی شده است.



شکل ۱-۲-۳ نمایی از فریم پروتکل HDLC در ساده‌ترین حالت

در پروتکل HDLC برای انتقال اطلاعات که توسط ایزو پذیرفته شده است. پیام‌ها در واحدهایی که قاب نامیده می‌شود انتقال می‌یابد. مقدار داده‌های مختلفی را می‌توان در قاب‌ها ذخیره نمود اما سازمان‌دهی تمام آنها باید یکسان باشد.

مدهای کار HDLC

در HDLC، ایستگاه‌ها می‌توانند نقش اولیه (Primary) یا ثانویه (Secondary) و یا ایستگاه ترکیبی (Combine) را ایفا کنند. در نقش اولیه، ایستگاه می‌تواند با ارسال فرمان شروع، یک ارتباط را آغاز کند که آن را با P نشان می‌دهیم. در نقش ثانویه، ایستگاه تنها به‌فرمان ارسالی جواب می‌دهد و

نمی‌تواند به‌تنهایی یک فرمان را آغاز کند. ما آن را با S نشان می‌دهیم؛ و سرانجام در نقش ترکیبی، ایستگاه هم می‌تواند اولیه و هم ثانویه باشد که آن را با PS نشان می‌دهیم. HDLC، دو مد معروف ارسال را عرضه می‌کند:

مد پاسخ طبیعی یا NRM- Normal Response Mode

مد NRM، یک ایستگاه اولیه و یک یا چند ایستگاه نقش ثانویه را دارا می‌باشند. هر NRM می‌تواند نقطه‌به‌نقطه یا چند نقطه‌ای باشد و ایستگاه اولیه می‌تواند پیام‌ها را ارسال کند و ایستگاه‌های ثانویه تنها می‌توانند پاسخ دهند.

مد متوازن غیر هم گام یا ABM- Asynchronous Balanced Mode

در ABM، ارتباط متوازن است و ایستگاه نقش ترکیبی دارد؛ یعنی، می‌تواند آغازکننده‌ی ارتباط باشد و فرمان بدهد و از ایستگاه مقابل پاسخ را دریافت کند.

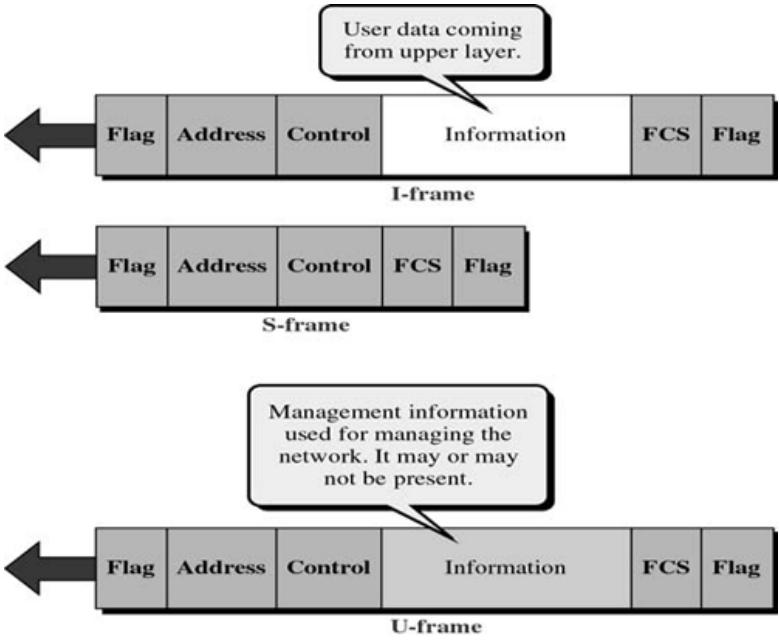
فریم‌های HDLC

جهت فراهم ساختن قابلیت انعطاف لازم برای پشتیبانی از تمام گزینه‌های ممکن در مدها و پیکربندی‌هایی که در بالا وصف شدند، HDLC از سه نوع فریم حمایت به عمل می‌آورد:

فریم‌های اطلاعاتی (I-Frame :Information Frame): فریم‌هایی هستند که برای ارسال اطلاعات به کار می‌روند و اطلاعات مربوط به کنترل جریان و خطا از جمله شماره ترتیب فریم را در یک بایت کنترلی به همراه دارند.

فریم‌های نظارتی (S-Frame :Supervisor Frame): فریم‌هایی هستند که حاوی داده نبوده و وقتی به کار می‌روند که مکانیزم ارسال پیام متوقف شده و پیام باید برای از سرگیری ارسال و دریافت پیام‌ها آغاز گردد.

فریم‌های بدون شماره (U-Frame :Unnumbered Frame): فریم‌هایی هستند که برای انتقال اطلاعات مدیریت پیوند از جمله برای ارسال فریم‌های کنترلی و برپاسازی ارتباط یا قطع آن، ارسال می‌شوند.



شکل ۱-۲-۴ نمایی از انواع فریم پروتکل HDLC

هر فریم HDLC می‌تواند حاوی حداکثر ۶ فیلد باشد: فیلد نمایشگر شروع، فیلد آدرس، فیلد کنترل، فیلد اطلاعات، فیلد شماره ترتیب (Frame check Sequence) و فیلد نمایشگر پایانی. در انتقالات چند فریمی، نمایشگر پایانی

می‌تواند دوتایی شود. لذا، نمایشگر دوم معرف نمایشگر شروع فریم بعدی خواهد بود.

فیلد نمایشگر: یگ الگوی هشت بیتی ۰۱۱۱۱۱۱۰ است که ابتدا و انتهای فریم را مشخص می‌کند. این الگو در هیچ جای دیگر فریم دیده نمی‌شود، لذا در فرستنده و گیرنده، مکانیزم بیت گذاری (گذاشتن یک صفر پس از پنج یک در فرستنده و برداشتن صفر بعد از پنج یک در گیرنده) پیاده سازی شده است.

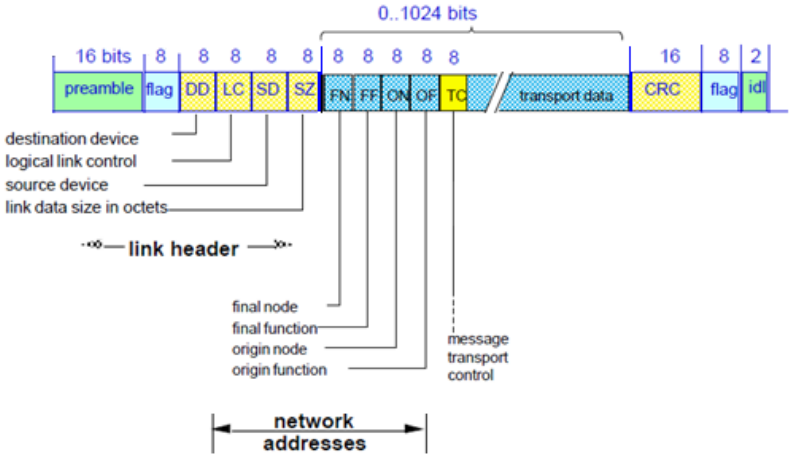
نمایشگر آدرس: مربوط به آدرس ایستگاه ثانویه است. به کمک یک بایت می‌توان ۱۲۸ ایستگاه را آدرس‌دهی کرد. یک بیت از آن نیز به عنوان بیت رزرو خواهد بود.

فیلد کنترل: یک سگمنت یک یا دو بیتی است که جهت کنترل جریان و کنترل خطا به کار می‌رود. تفسیر بیت‌ها در این فیلد برای انواع مختلف فریم‌ها متفاوت است.

فیلد اطلاعات: I-frame حاوی داده‌ی کاربر و U-frame شامل اطلاعات مربوط به مدیریت شبکه می‌باشد. بزرگی آن از شبکه‌ای به شبکه‌ی دیگر تفاوت دارد ولی داخل یک شبکه همواره مقدار ثابتی دارد.

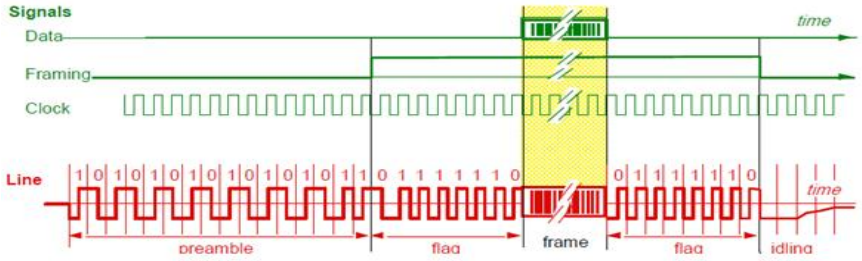
فیلد FCS: برای کشف خطا در HDLC استفاده می‌شود. هر FCS می‌تواند حاوی یک CRC دو بایتی یا چهار بایتی باشد.

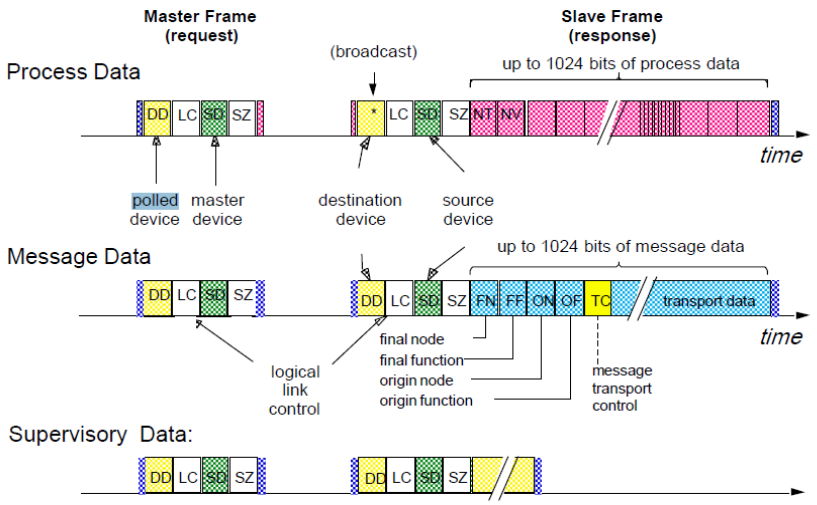
در زیر پروتکل ارسال پیغام به فرمت (ISO) ۳۳۰۹HDLC و منچستر کدینگ ویژه WTB قابل مشاهده است.



شکل ۱-۲-۵ نمایی از انواع فریم پروتکل WTB

بر اساس آنچه در معرفی فریم‌های مدل HDLC اشاره شد سه نوع بسته اطلاعات (پروسس، پیام و نظارت) در WTB به فرم زیر قابل تعریف است که در زیر ترسیم شده است.

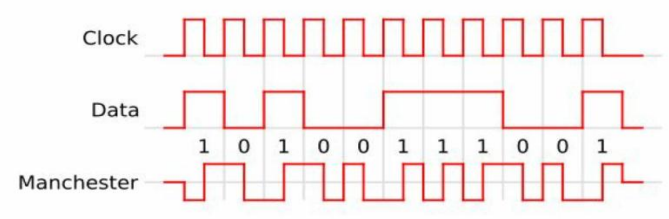




شکل ۱-۲-۶ انواع فریم پروتکل WTB

روش منچستر کدینگ

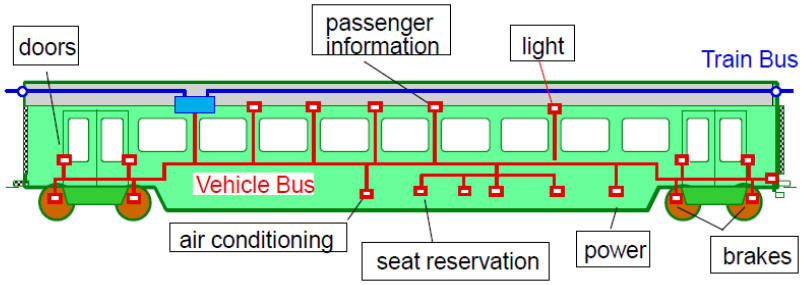
در این تکنیک محتوای دیتا و کلاک مورد استفاده قرار می‌گیرد بدین صورت که سیگنال کلاک و دیتا با هم XOR می‌شوند، در وسط هر دوره بیتی سیگنال منتجه، یک گذار وجود دارد که به عنوان مثال مانند شکل زیر است اگر از high به low برویم یا به عبارتی لبه پایین رونده داشته باشیم دیتا صفر منطقی بوده و اگر از low بهhigh برویم، یا به عبارتی لبه بالارونده داشته باشیم دیتا یک منطقی بوده است. این نوع کدینگ در شبکه اترنت قطارهای شهری طبق استاندارد ۳.۸۰۲IEEE نیز وجود دارد.



شکل ۱-۲-۷ نمایی از سیگنال کد شده به روش منچستر

کاربرد و نقش MVB

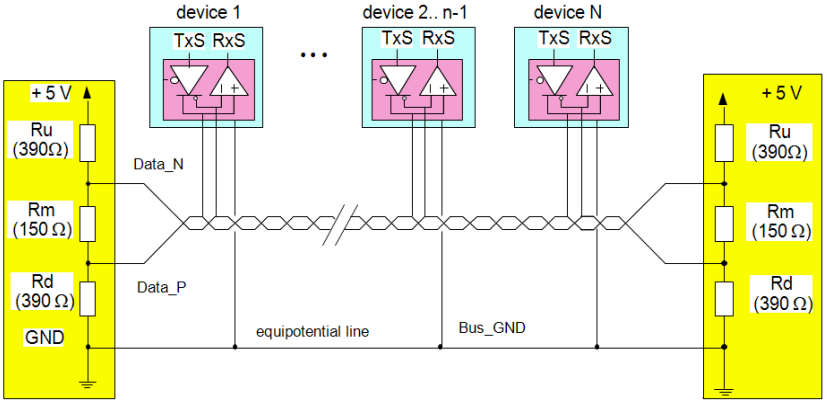
در شبکه TCN در لایه پائین تر برای اتصال تجهیزات داخلی یک واگن از یک Vehicle_Bus استفاده می‌شود. در استاندارد IEC۶۱۳۷۵-۱، MVB (Multifunction Vehicle Bus)به عنوان Vehicle bus استاندارد برای استفاده در این سطح معرفی شده است؛ اما از باس‌های دی‌گر از قبی‌ل CAN، CAN Open نیز می‌توان در این سطح استفاده کرد. بأس MVB یک ارتباط سریال با سرعت ۱/۵ Mbit/Sec و با تأخیر ۱ms می‌باشد که در لایه فیزیکی خود از زوج سیم به هم تابیده یا فیبر نوری یا سیم کوتاه استفاده می‌کند. در بأس MVB تعداد گره‌های قابل اتصال به شبکه ۲۵۶ گره می‌باشد و قابلیت اتصال ۴۰۹۵ سنسور و محرک را به خود دارد.



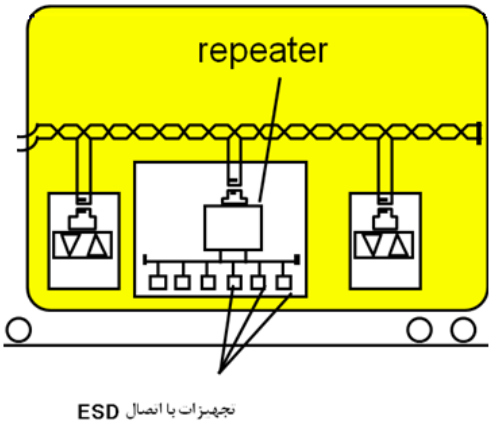
شکل ۱-۲-۸ ارتباطات بین Vehicle_Bus و Train_Bus و تجهیزات هر واگن

در لایه فیزیکی خود از سه روش زیر استفاده می‌کند:

۱- سیم در فواصل کم (Electrical Short Distance) ESD: در این روش از لایه سخت‌افزاری پروتکل Profibus که همان RS ۴۸۵ است، استفاده می‌شود از این روش در فواصل کمتر از ۲۰ متر استفاده می‌شود. در قطارهای شهری این روش جهت اتصال تجهیزات به تابلو اصلی MVB در هر واگن مورد استفاده قرار می‌گیرد.



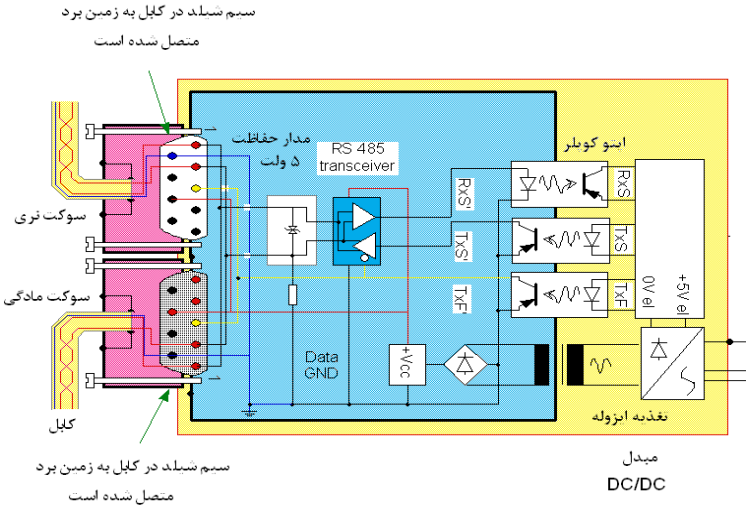
شکل ۱-۲-۹ ارتباطات بین تجهیزات در حالت ESD (Electrical Short Distance)



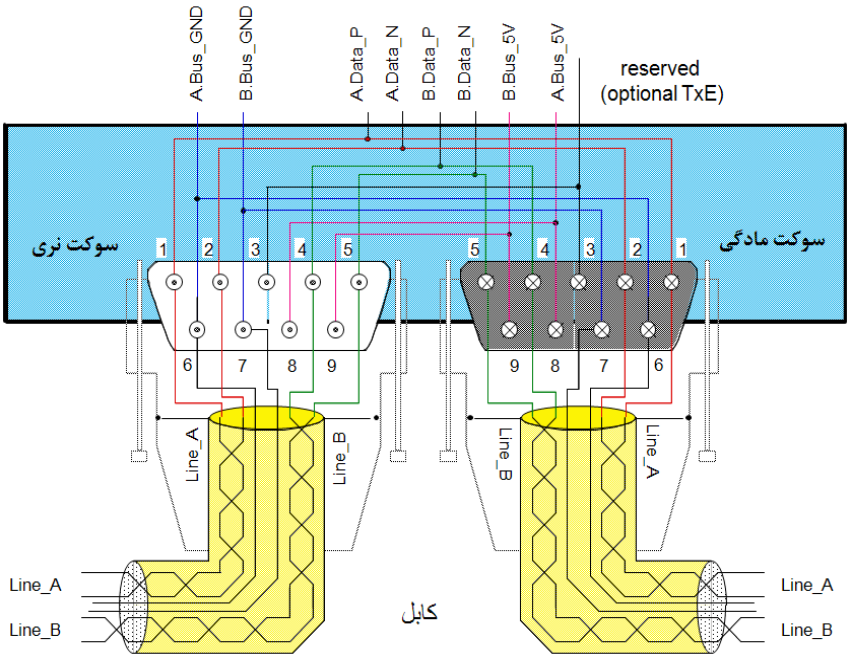
شکل ۱-۲-۱۰ ارتباطات بین تجهیزات در حالت ESD در یک واگن

ارتباطات در حالت ESD با ایزولاسیون الکتریکی

در این مدار از مدارات دیود بلاکینگ (سوپرسور)، اپتوکوپلر و تغذیه ایزوله DC/DC استفاده شده است.

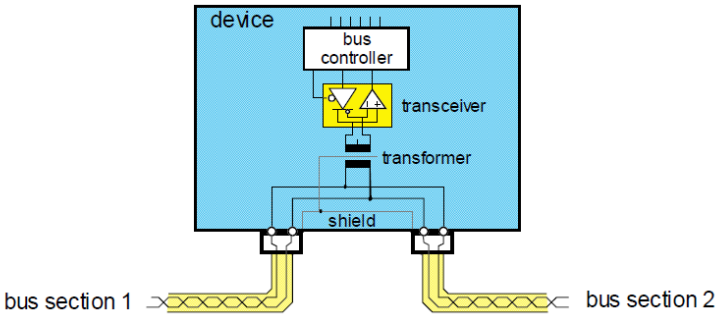


شکل ۱-۲-۱۱ ارتباطات در حالت ESD با ایزولاسیون الکتریکی

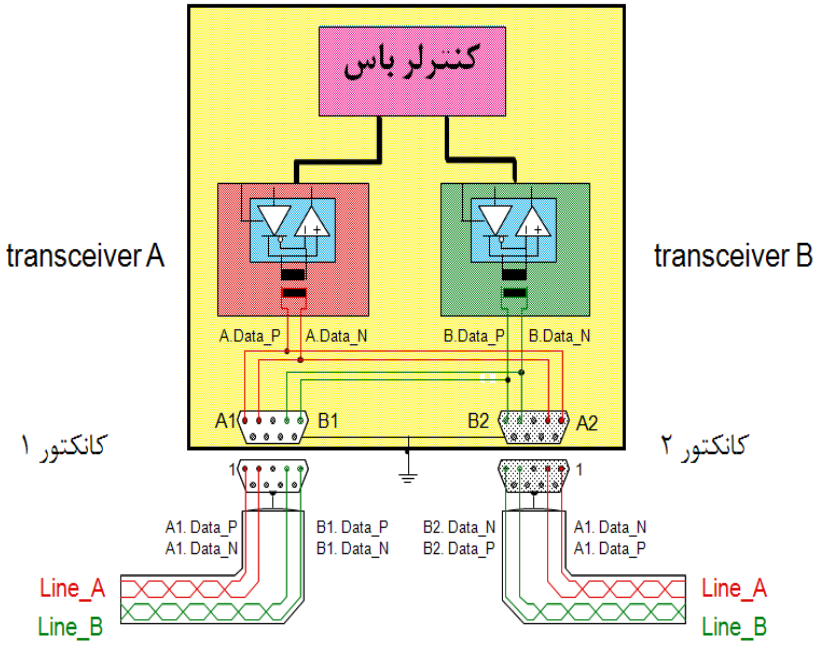


شکل ۱-۲-۱۲ ارتباطات ESD در حالت دو سیمِ

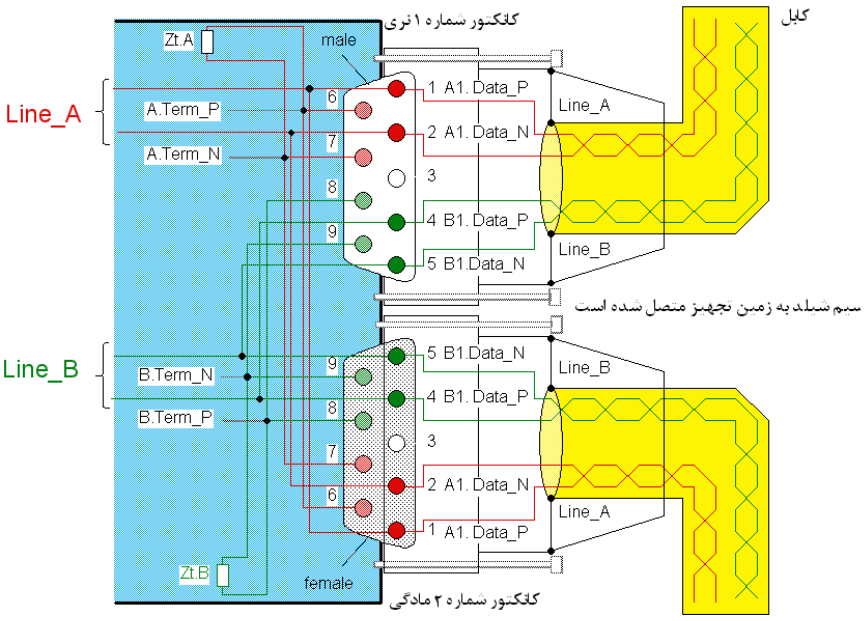
۲- زوج سیم به هم تابیده(EMD (Electrical Medium Distance : در این روش از یک کوپلینگ ترانسفورمری استفاده می‌شود و از این روش با کابل ۱۲۰ Ohm طبق ۲-۱۱۵۸IEC در فواصل کمتر از ۲۰۰ متر می‌توان ۳۲ تجهیز را به هم متصل نمود. این روش انتقال برای MVB در قطارهای شهری بسیار مرسوم است. در مترو تهران شرکت بمباردیر از این نوع اتصال استفاده کرده است.



شکل ۱-۲-۱۳ کوپلینگ ترانسفورمری استفاده شده در EMD

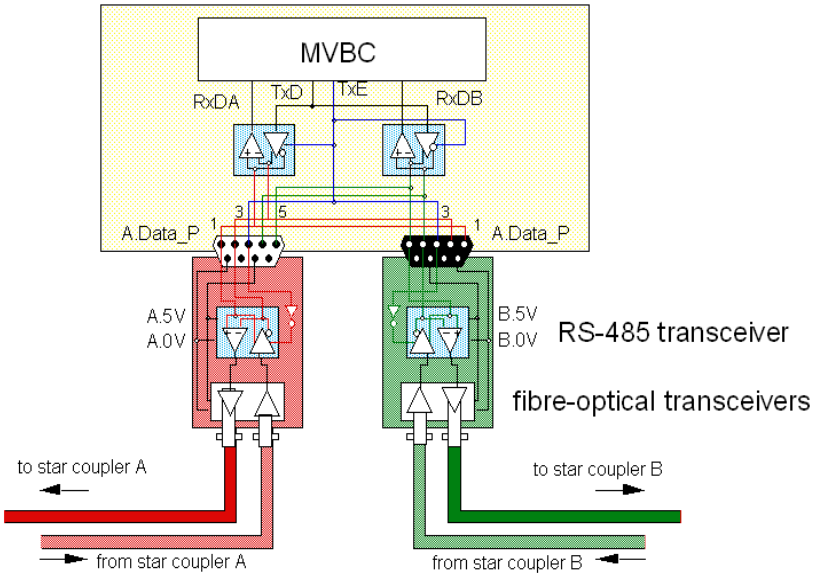


شکل ۱-۲-۱۴ اتصال EMD در حالت دو سیمِ



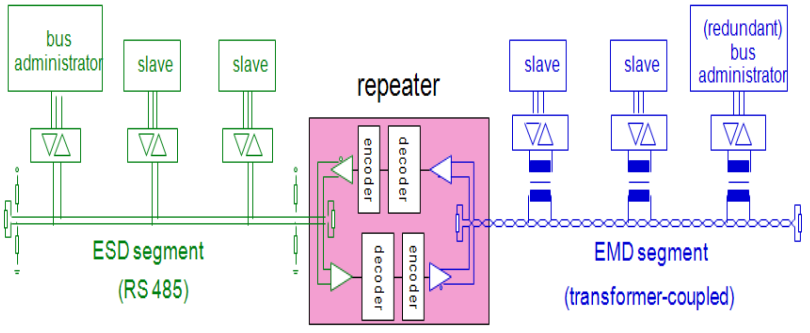
شکل ۱-۲-۱۵ اتصال کابل‌ها در حالت EMD دو سیمِ

۳- فیبر نوری (Optical Glass Fibre) OGF: در این روش و با استفاده از فیبر نوری ۲۴۰µm silica می‌توان دیتا را تا ۲۰۰۰ متر انتقال داد.

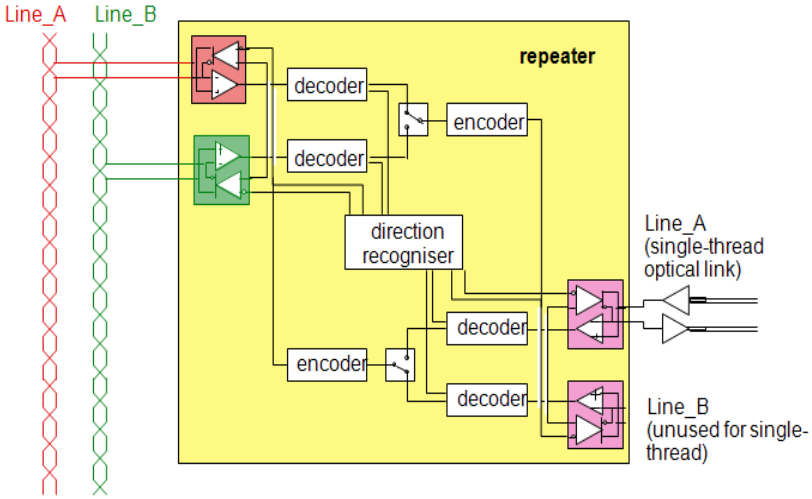


شکل ۱-۲-۱۶ مبدل ESDبه OGF

جهت اتصال EMD به ESD یا OGF باید از تجهیزی به نام Repeater استفاده کرد. این تجهیز انکودر دارد و سیگنال، جهت دیتا، وجود تصادم در دیتا را تشخیص داده سپس با دیکدر فریم دیتا را ساخته و باز ارسال می‌کند. در زیر نمایی از دو نوع Repeater آمده است.



شکل ۱-۲-۱۷ اتصال EMD به ESD با Repeater

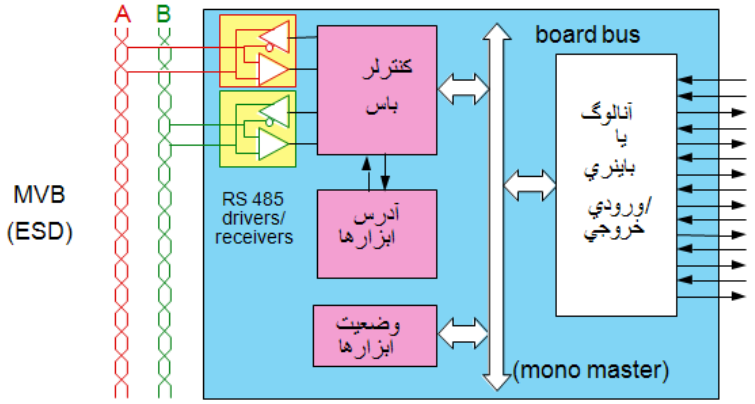


شکل ۱-۲-۱۸ اتصال EMD به OGF با Repeater

کلاس‌های تجهیزات دارای MVB

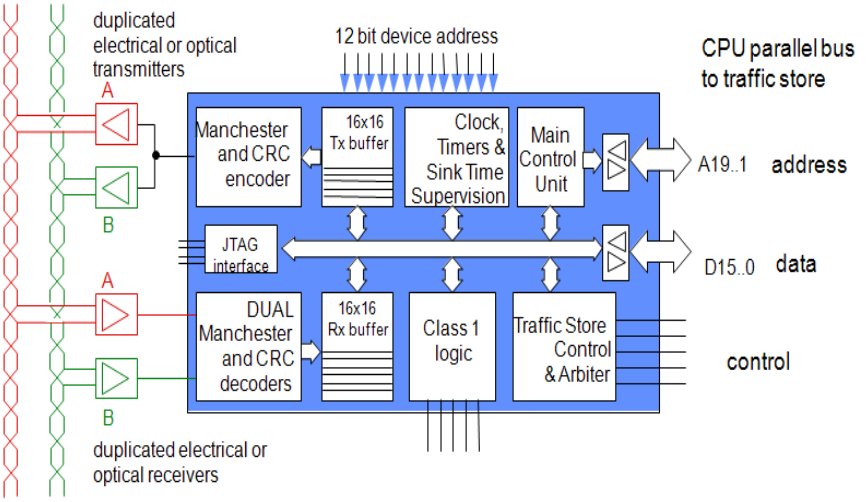
کلاس ۱:

برای اتصال سنسورها و اکچویتورها از این کلاس استفاده می‌شود، این کلاس تجهیزات به میکروکنترلر نیاز ندارند، این تجهیزات در ارسال محتوای پیام سهیم نیستند و کنترلر بأس هم ورودی/خروجی و هم بأس را کنترل می‌کند.



شکل ۱-۲-۱۹ MVB، کلاس ۱

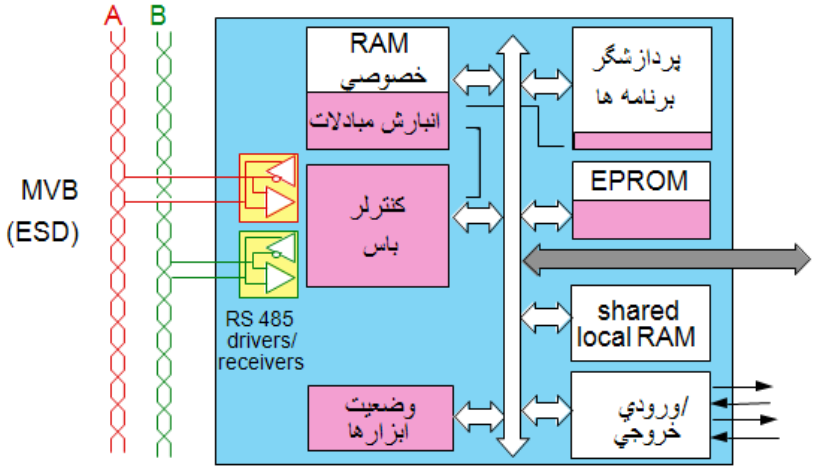
کنترلر بأس وظیفه تولید و آنالیز خودکار پیام‌ها را دارد. این تجهیز اسیلاتوری ۲۴ مگاهرتزی دارد و قابلیت ارتباط را با میکروکنترلرهای ۸ و ۱۶ بیتی را دارد. دارای ۲ به توان ۱۲ یعنی ۴۰۹۶ درگاه برای پردازش اطلاعات است (در کلاس ۱ حداکثر ۱۶ درگاه دارد)، دارای حافظه‌ای ۱۶ کیلوبایتی تا حداکثر ۱ مگا بایتی برای انبارش مبادلات و توابع مدیریت باس‌ها است، خطاهای ارتباطات در آن ثبت می‌شود، دارای پورت‌های جهت سنکرون کردن و هم زمانی است، به صورت یک پکیج ۱۰۰ پایه و با فوت پرینت QFP می‌باشد.



شکل ۱-۲-۲۰ کنترلر بأس MVBC- bus controller ASIC

کلاس ۲ و ۳:

کلاس تجهیزات ۲ و ۳ به میکروکنترلر نیاز دارند، این تجهیزات محتوای پیام تغییر می‌توانند بدهند. در کلاس ۲ تجهیزاتی هستند که درگاه ورودی/خروجی در آنها قابلیت پیکربندی دارد ولی قابل‌برنامه‌ریزی نمی‌باشند.



شکل ۱-۲-۲۱ MVB، کلاس ۲ و ۳

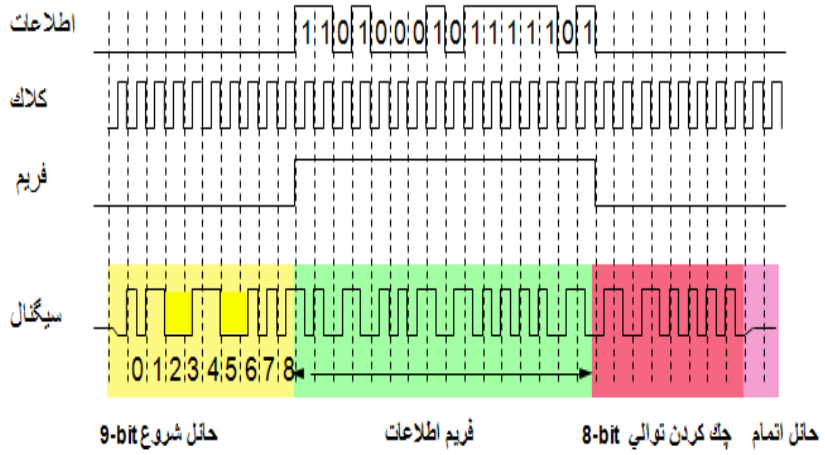
کلاس ۴ و ۵:

تجهیزات کلاس ۴ می‌توانند مدیر بأس شوند و امکانات تست وضعیت سیستم، نظارت، پیکربندی، خواندن وضعیت ابزارها و برنامه‌ریزی را فراهم آورند.

تجهیزات کلاس ۵ عملاً گذرگاه ارتباطات چند لایه مثل MVB و WTB می‌باشند.

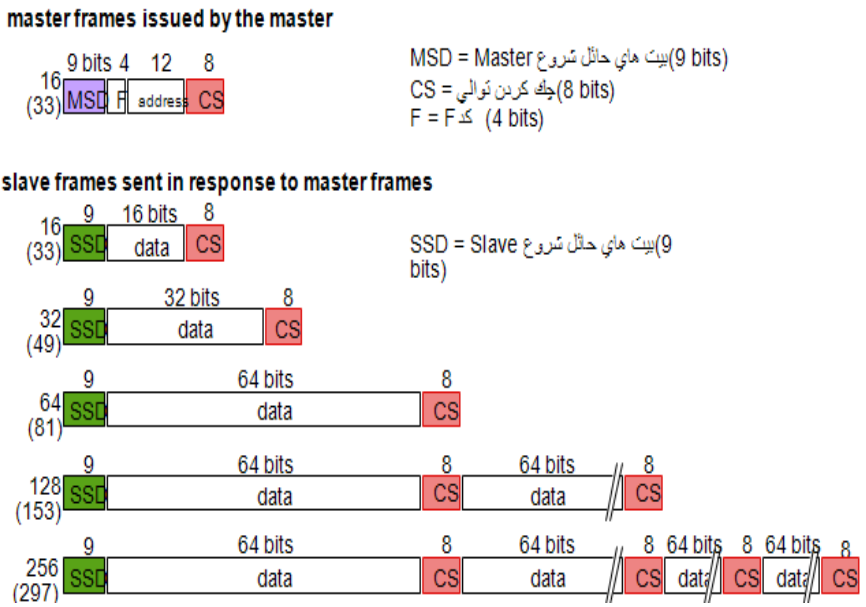
فریم پروتکل MVB

پروتکل ارسال و دریافت MVB در قالب منچستر کدینگ است. بیت‌های اطلاعات بعد از حائل شروع ۹ بیتی ارسال می‌شوند، این بیت‌ها جهت همزمانی ارسال می‌شود.



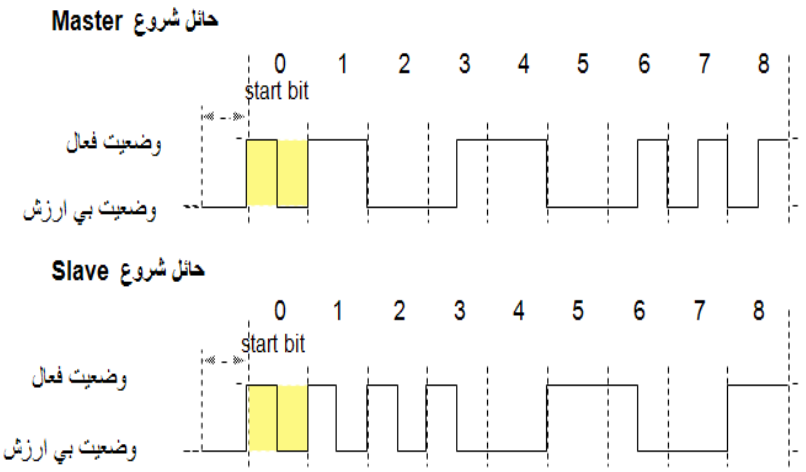
شکل ۱-۲-۲۲ رمز گذاری به روش منچستر

برای پروتکل ارسال و دریافت MVB دو فرم کلی master و slave وجود دارد. بیشترین زمان مجاز تأخیر بین این دو فریم حداکثر ۴۲/۷ میکرو ثانیه می‌باشد. این عامل، عامل محدود کننده طول هادی ارسال کننده می‌باشد.



شکل ۱-۲-۲۳ فرم کلی master و slave پروتکل MVB

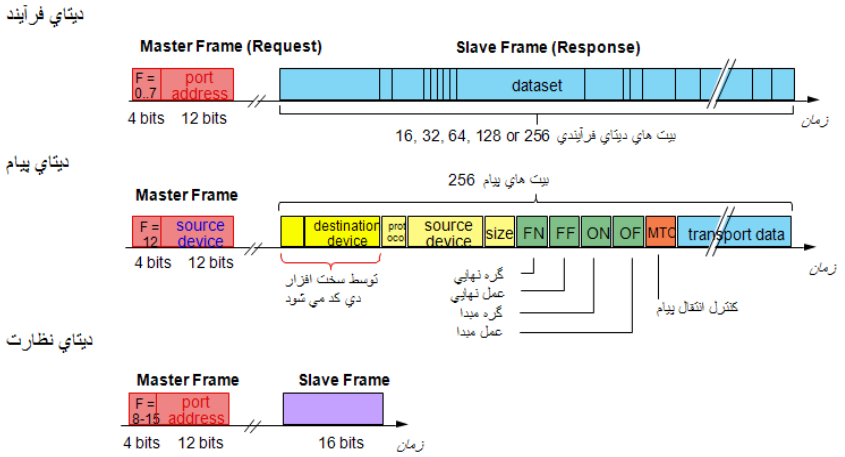
حائل شروع ۹ بیتی برای masterو slave شکل‌های متفاوتی دارد.



شکل ۱-۲-۲۴ حائل شروع master, slave

انواع فریم پروتکل MVB در حالت Slave و Master

سه نوع بسته اطلاعات (پروسس، پیام و نظارت) به فرم زیر در MVB ارسال و دریافت می‌شود. ارسال پیام در این سه روش جز بسته پیام دقیقاً به روشی است که در بالا اشاره شده است.

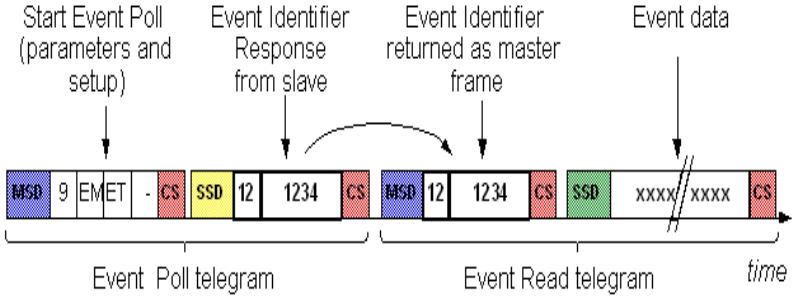


شکل ۱-۲-۲۵ انواع فریم پروتکل MVB در حالت Slave

بسته‌های ارسالی از سوی Master دارای ۴ بیت به نام F-code اند که در آن نوع عملیات درخواستی اعم از درخواست (پروسس، پیام، رخدادها و وضعیت تجهیزات) توسط آن به slave فهمانده می‌شود کد F جدولی مطابق با مقادیر زیر دارد.

Master Frame			Slave Frame			
F_code	address	request	source	size	response	destination
0				16		
1			single	32		all
2	logical	Process_Data	device	64	Process_Data	devices
3			subscribed	128	(application	subscribed
4			as	256	-dependent)	as
5		reserved	source	-		sink
6		reserved		-		
7		reserved		-		
8	all devices	Master_Transfer	Master	16	Master_Transfer	Master
9	device	General_Event	>= 1devices	16	Event_Identifier	Master
10	device	reserved	-	-		
11	device	reserved	-	-		
12	device	Message_Data	single device	256	Message_Data	selected device
13	group	Group_Event	>= 1devices	16	Event_Identifier	Master
14	device	Single_Event	single device	16	Event_Identifier	Master
15	device	Device_Status	single device	16	Device_Status	Master or monitor

به عنوان مثال Master فرمان General Event Poll را برای اطلاع از رخدادها به صورت زیر با F-Code مرتبط یعنی ۹ ارسال داشته است و Slave در جواب Event Id را ارسال کرده است. حال Master فرمان ارسال پیام را با F-Code مرتبط یعنی ۱۲ و با اطلاعات Event Id دریافتی از Slave را ارسال می‌دارد و Slave در جواب اطلاعات رخداد را ارسال می‌دارد.



شکل ۱-۲-۲۶ پروتکل MVB در حالت ارسال و دریافت رخدادها

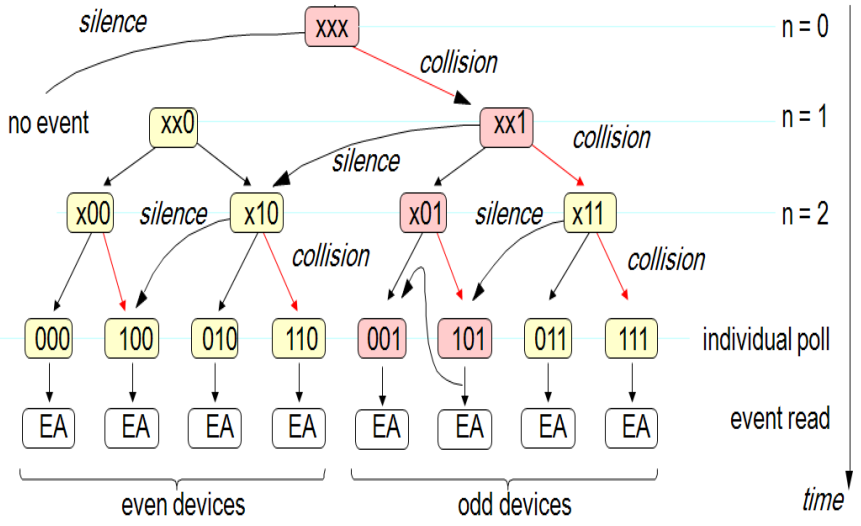
اگر در جواب گوی تصادم رخ داد این بار براساس آدرس‌های فیزیکی فرد بار دیگر Master سؤال می‌کند:

اگر فقط یک جواب آمد،Master مطابق بالا پیام را به Slave مورد نظر می‌فرستد.

اگر جواب نیامد Master از آدرس‌های فیزیکی زوج سؤال می‌کند.

اگر تصادم همچنان باقی است Master برروی بیت دوم آدرس فیزیکی متمرکز شده و از دسته دیگری از ابزارها سؤال می‌کند.

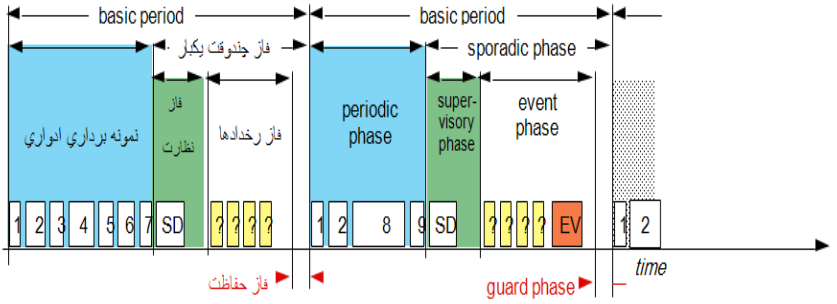
به عنوان مثال در یک گروه شامل ۸ تجهیز، عملیات رفع تصادم به صورت زیر است:



شکل ۱-۲-۲۷ عملیات رفع تصادم

پروتکل MVB در فرم master وظایف زیر را برعهده دارد:

۱. نمونه برداری ادواری از درگاه‌ها طبق لیست نمونه برداری (Poll list)
۲. توجه به پیام‌های نامنظم وقوع رخداد
۳. بررسی تجهیزات جهت نظارت
۴. انجام آزمون خود ارزیابی و انتقال Mastership



شکل ۱-۲-۲۸ وظایف MVB در فرم master

اطلاعات ادواری شامل: وضعیت متغیرها و وضعیت تجهیزات مثل موقعیت، سرعت و ارسال فرامین است بین ۱ تا ۲۰۰ میلی ثانیه پاسخ داده می‌شوند.

اطلاعات چندوقت یکبار شامل: رخدادهای تجهیزات همچون پیام‌های عیب یابی و ثبت رخدادها است که پاسخ انسانی باید بدان داده شود.

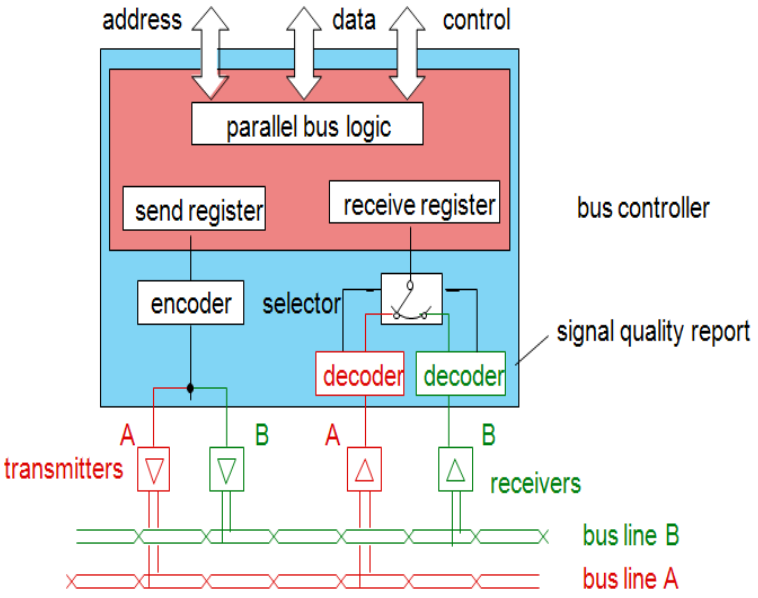
در زمان پیکربندی مدیر بأس MVB باید لیست نمونه برداری (Poll list) از روی اطلاعات زیر ساخته شود:

۱. لیست پورت آدرس‌ها
۲. مدت زمان پاسخ هر بأس
۳. لیست تجهیزات شناسایی شده برای بررسی تجهیزات

۴. لیست تجهیزات با قابلیت تبدیل شدن به مدیر بأس در عملیات انتقال Mastership

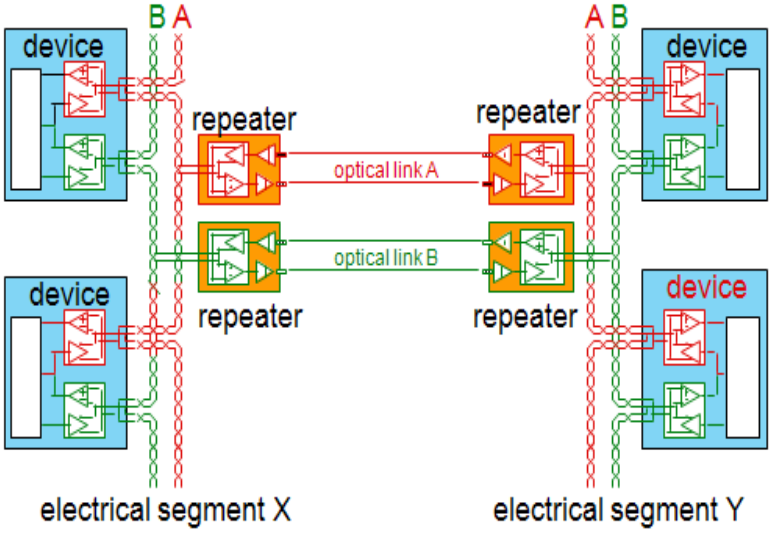
افزایش قابلیت اطمینان

به جهت افزایش قابلیت در دسترس بودن اطلاعات و نه به جهت یکپارچه سازی اطلاعات، از طریق دو کانال به نام‌های A و B انتقال می‌یابند. اطلاعات همواره از یک کانال قرائت و کانال دیگر مانیتور می‌شود. کانال اطلاعات قابل تعویض است و کیفیت اطلاعات کانال‌ها کنترل می‌شود و طی عملیات تعویض کانال یک فریم اطلاعات ممکن است از دست برود. طی عملیات ارسال از هر دو کانال پیام ارسال می‌گردد.



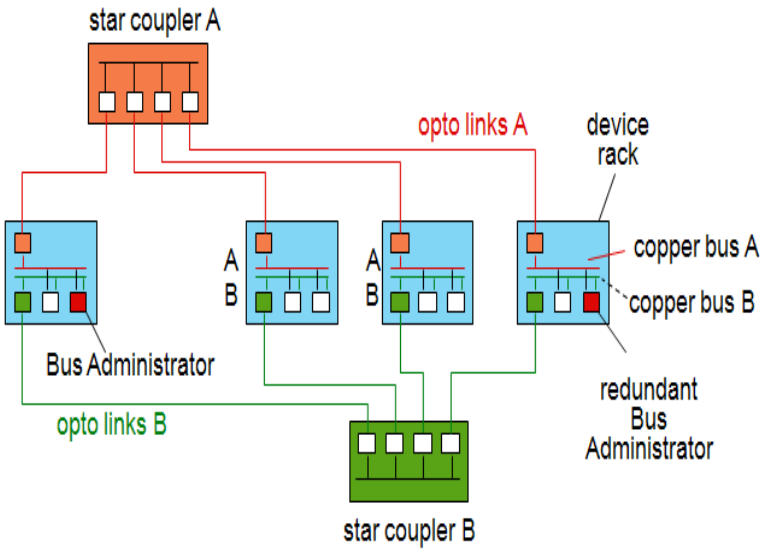
شکل ۱-۲-۲۹ انتقال از طریق دو کانال

اگر لایه فیزیکی تکرار کننده متصل کننده دو بخش به هم می‌باشد در این صورت نیز این لایه برای افزایش قابلیت اطمینان باید دو عدد در نظر گرفته شود.



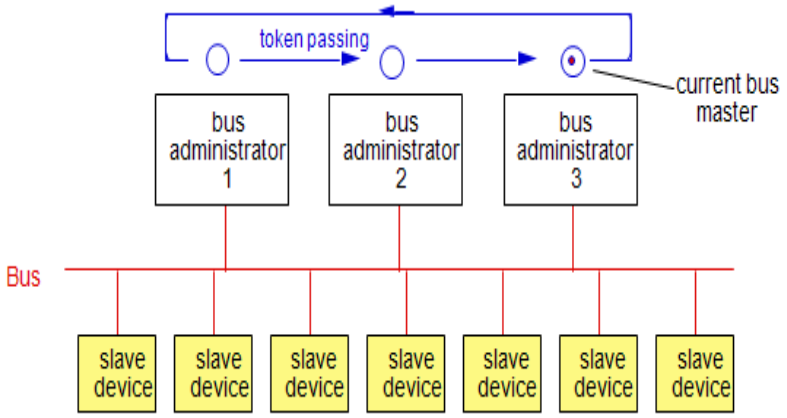
شکل ۱-۲-۳۰ لایه فیزیکی تکرارگر در انتقال از طریق دو کانال

اگر لایه فیزیکی فیبر نوری جهت اتصال دو بخش به هم استفاده می‌شود در این صورت نیز این لایه برای افزایش قابلیت اطمینان از آرایش زیر استفاده می‌شود.



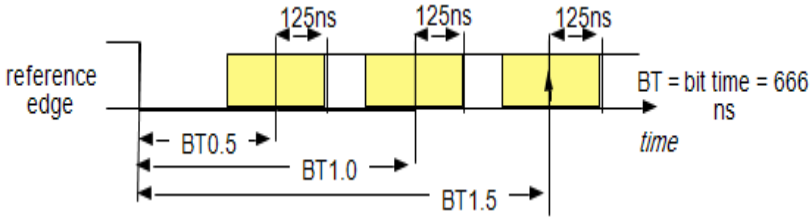
شکل ۱-۲-۳۱ لایه فیزیکی فیبر نوری در انتقال از طریق دو کانال

تمرکز مدیریت یک بأس در یک نقطه، نقطه ضعف سیستم است لذا یکی از راه‌های افزایش قابلیت اطمینان افزایش تعداد بأس‌های مدیریتی است. اگر مدیریت بأس تغییر و تحولی را مشاهده نکرد وارد سیکل حکمیت داخلی می‌شود اگر از حکمیت داخلی تجهیز را دارای نقص تشخیص دهد، مدیریت بأس به تجهیز پشتیبان دیگر انتقال می‌یابد.



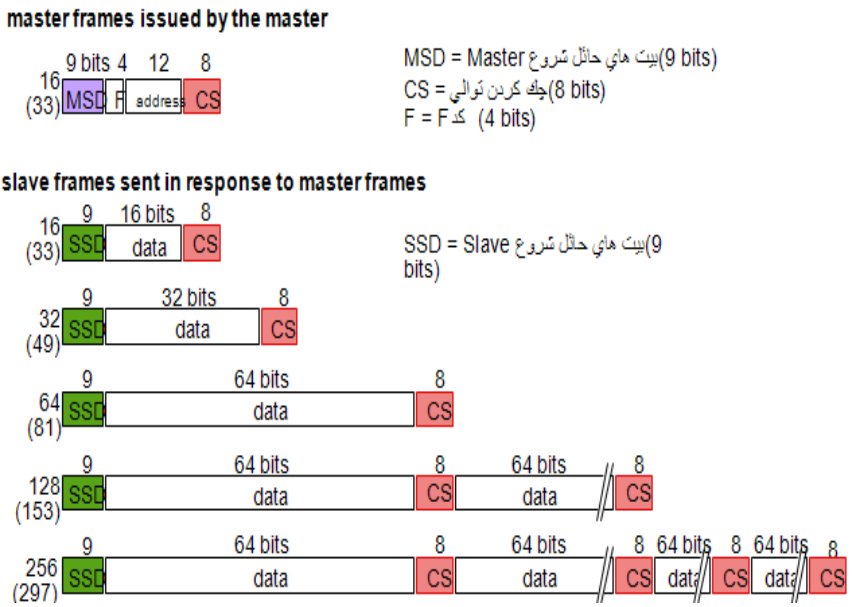
شکل ۱-۲-۳۲ افزایش تعداد بأس‌های مدیریتی

الگوریتمی به نام نظارت بر کیفیت سیگنال، فریم‌های مشکوک را به جهت کاهش اثر نویز حذف می‌نماید.



شکل ۱-۲-۳۳ الگوریتم نظارت بر کیفیت سیگنال

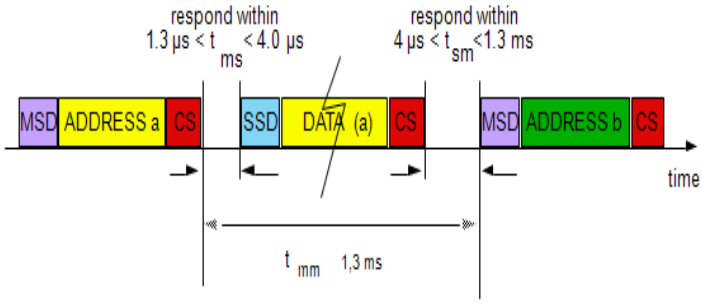
در فریم‌های Slave/Master، check sum های ۸ بیتی مطابق class ۵۷TC ۲FT برای هر گروه حداکثر ۶۴ بیتی و وجود دو فرم حائل متفاوت (SSD وMSD) در فریم‌های یاد شده و هم اندازه بودن سایز اطلاعات در مبدأ و مقصد دیگر راه‌های شناسایی اطلاعات معیوب می‌باشد.



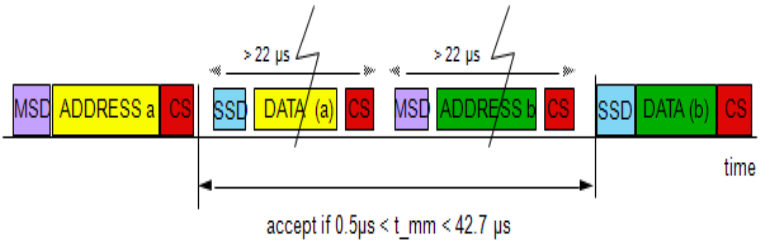
شکل ۱-۲-۳۴ check sum های ۸ بیتی و حائل متفاوت در Slave/Master

مدت زمان پاسخ سیستم نظارت در هنگام از دست رفتن یک یا دو بسته اطلاعات به صورت زیر است، رعایت این زمان‌بندی خاص باعث قبول شدن اطلاعات دریافتی است.

single frame loss:



double frame loss:



شکل ۱-۲-۳۵ از دست رفتن یک یا دو بسته اطلاعات

پروتکل MVB به‌طور خلاصه

به‌طور خلاصه MVB پروتکلی ارتباط سریال با سرعت ۱/۵ Mbit/Sec و با مشخصات زیر است:

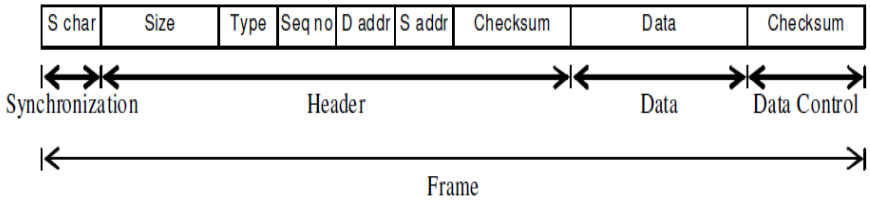
Topography:	bus (copper), active star (optical fibre)
Medium:	copper: twisted wire pair optical: fibres and active star coupler
Covered distance:	OGF: 2000 m, total 4096 devices EMD: 200 m copper with transformer-coupling ESD: 20 m copper (RS485)
Communication chip	dedicated IC available
Processor participation	none (class 1), class 2 uses minor processor capacity
Interface area on board	20 cm2 (class 1), 50 cm2 (class 2)
Additional logic	RAM, EPROM , drivers.
Medium redundancy:	fully duplicated for availability
Signalling:	Manchester II + delimiters
Gross data rate	1,5 Mb/s
Response Time	typical 10 µs (<43 µs)
Address space	4096 physical devices, 4096 logical ports per bus
Frame size (useful data)	16, 32, 64, 128, 256 bits
Integrity	CRC8 per 64 bits, HD = 8, protected against sync slip

پروتکل MVB در محصولات Bombardier

خط MVB نیز یک خط MVB پشتیبان دارد. این دو خط معروف به کانال‌های A و B می‌باشند. هر کانال نیز از ۲ سیم تشکیل شده است. دیتاهای

از نوع Process که بحرانی و بسیار مهم‌اند در این شبکه در کمتر از ۱۰۰ میلی ثانیه باید ارسال شوند.

ساختار فریم MVB از نوع Message شرکت بمباردیر:



شکل ۱-۲-۳۶ فریم پروتکل MVB شرکت بمباردیر

این ساختار شامل ۴ بخش اصلی است:

۱. بایت synchronization: جهت تشخیص شروع رشته دیتا MVB است این بیت‌ها، بیت‌های حائل شروع می‌باشند.

۲. بایت‌های Header: این بایت‌های شامل

۱. بیت‌های Size که سایز داده پیام را (که باید کمتر از ۱۶ کیلوبایت باشد) تعیین می‌کند.

۲. بیت‌های Type که نوع پیام ارسالی را تعیین می‌کند.

۳. بیت‌های شماره توالی که شماره ID فریم ارسالی را مشخص می‌کند و جهت تمیز دادن در ارسال مجدد استفاده می‌شود.

۴. بیت‌های آدرس مقصد

۵. بیت‌های آدرس فرستنده

۶. بیت‌های Checksum ویژه اطلاعات Header که اگر گیرنده به همان برسد پیغام بدون خطاست و بیت‌های خطا در پیام ارسال نخواهند شد.

۳. بایت‌های دیتا

۴. بیت‌های Checksum ویژه اطلاعات Data کاملاً مشابه Checksum ویژه Header

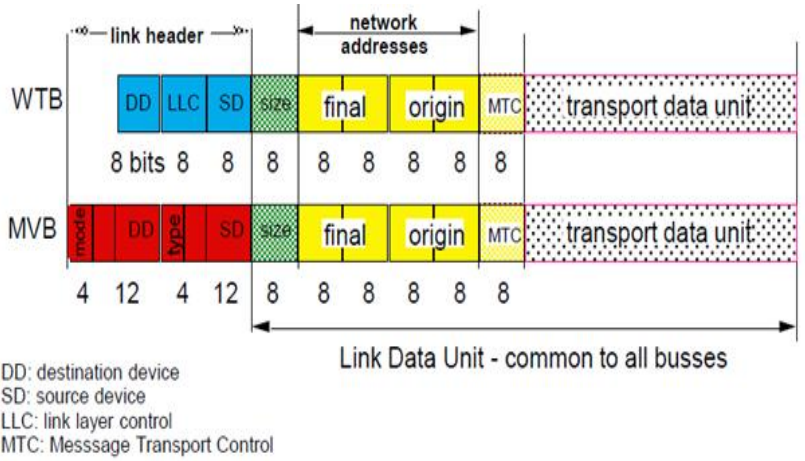
تفاوت‌های WTB و MVB:

تفاوت Train–bus و Vehicle-bus در لایه فیزیکی و لایه پیوند داده می‌باشد و در لایه‌های بالاتر تفاوتی با هم ندارند.

بأس MVB یک ارتباط سریال با سرعت ۱/۵ Mbit/Sec می‌باشد، ولی بأس WTB یک ارتباط سریال با سرعت ۱ Mbit/Sec می‌باشد.

در بأس MVB تعداد گره‌های قابل اتصال به شبکه ۲۵۶ گره می‌باشد و ۴۰۹۵ سنسور و اکچویتور را دارد و در مدل‌های مختلف از ۲۰ تا ۲۰۰۰ متر قابلیت ارسال داده را دارد ولی در بأس WTB تعداد گره‌های قابل اتصال به شبکه ۳۲ گره و حداکثر طول ۸۶۰ متر می‌باشد.

شکل ارسال بسته پیام در این دو استاندارد دارای تفاوت‌های زیر است:

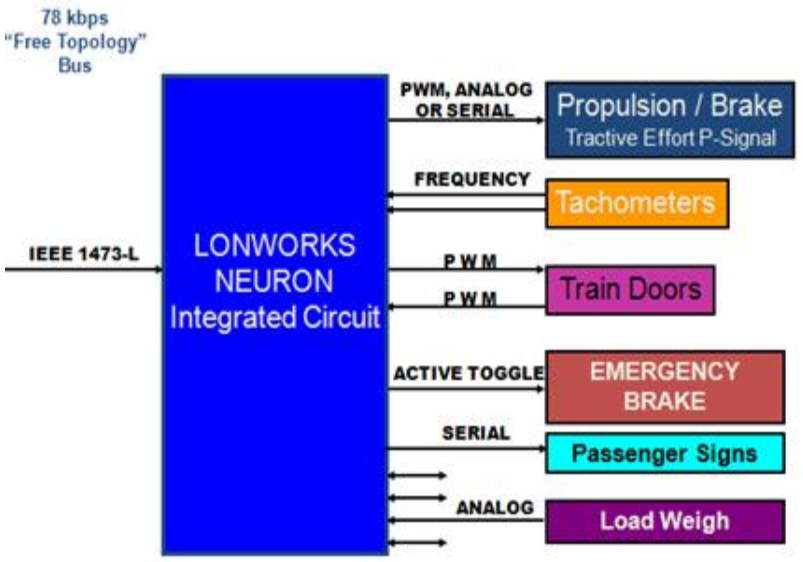


شکل ۱-۲-۳۷ مقایسه فریم پروتکل‌های MVB و WTB

بأس MVB دارای لایه سخت‌افزاری زوج سیم به هم تابیده است، ولی بأس WTB از لایه‌های سخت‌افزاری زوج سیم به هم تابیده یا فیبر نوری یا سیم کوتاه استفاده می‌کند.

استاندارد Lonworks

همان‌گونه که ذکر شد پروتکل Lonworks در سال ۱۹۹۹ تبدیل به استاندارد IEEE 1473-L گردید؛ اما پیش از آن نیز شرکت‌هایی از قبیل Bombardier, Kinki-Sharyo, Breda و ... محصولاتی را تحت پروتکل Lonworks طراحی کرده بودند؛ اما پس‌ازاین تاریخ استفاده از این استاندارد در طراحی محصولات ریلی در آمریکای شمالی و تمام نقاط دنیا گسترش یافت. این استاندارد نه تنها توسط سازندگان ریلی، بلکه توسط تأمین کنندگان قطعات ریلی نیز به سرعت مورد پذیرش قرار گرفت و در حال حاضر بسیاری از تأمین کنندگان قطعات، سیستم‌هایی نظیر: کنترل درب‌ها، کنترل اتوماتیک پیشرفته قطار AATC (Advance Automatic Train Control)، کنترل ترمز، کنترل Propultion و کنترل تهویه و ... را با این استاندارد تولید می‌کنند.



شکل ۱-۳-۱ کاربرد استاندارد IEEE۱۴۷۳L- و معماری شبکه Lonworks

فن‌آوری Lonworks در ابتدا توسط شرکت آمریکایی Echelon برای اولین بار عرضه شد.

معرفی اجزای فن‌آوری Lonworks

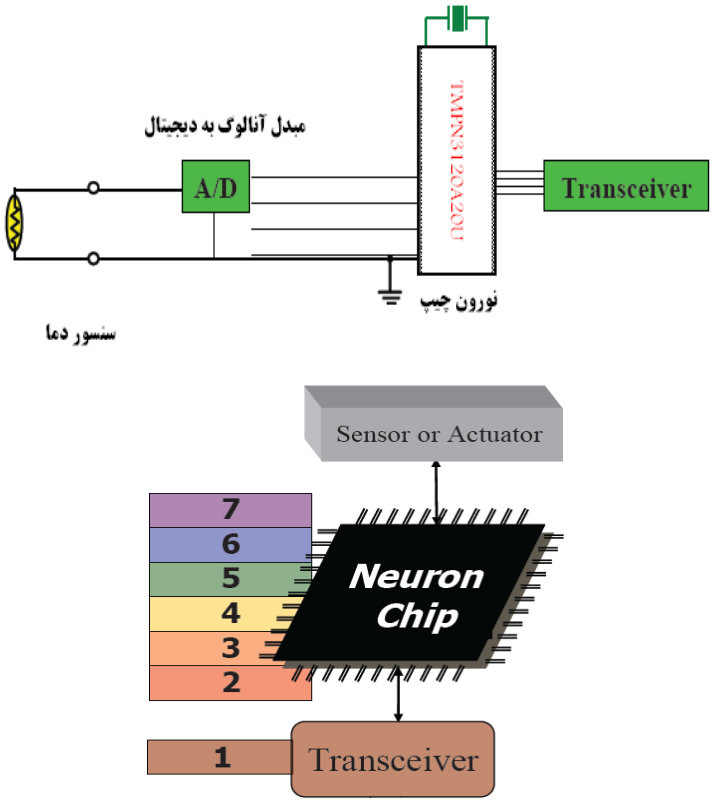
این فن‌آوری شامل اجزای زیر می‌باشد:

۱. نورن چیپ ها: نورون چیپ‌ها توسط توشیبا و موتورولا نیز تولید شده‌اند.

۲. پروتکل LON Talk: پروتکل LON Talk بر اساس مدل OSI عمل می‌کند.

۳. فرستنده و گیرنده‌های Lonworks

مزایت اصلی این پروتکل این است که تضمین می‌کند سازندگان بدون هیچ هماهنگی ابزارهای شبکه نظیر سنسورها، محرک‌ها یا کنترل کننده‌هایی که با هم کار می‌کنند را تولید کنند و تجهیزات بتوانند به دلیل اینکه آنها از پروتکل ارتباطی استاندارد عمومی استفاده می‌کنند، داده‌ها را با یکدیگر تبادل نمایند. به عنوان نمونه اتصال یک سنسور دما آنالوگ معمولی به شبکه Lonworks توسط نورن چیپ شرکت توشیبا به شماره U۲۰A۳۱۲۰TMPN در زیر مشاهده می‌شود.



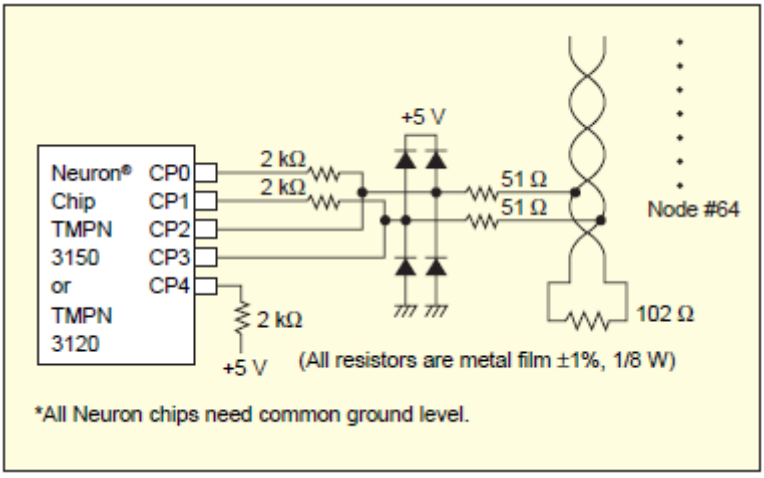
شکل ۱-۳-۲ اتصال سنسور دما آنالوگ به شبکه Lonworks

بعد از این اتصال برنامه‌ای به زبان Cبا استفاده از Neuron® C Programmer برای آن نوشته می‌شود که مبدل آنالوگ به دیجیتال بعد از هر بار RESET شدن مقدارoffset را فراخوانی کرده و کالیبراسیون را انجام دهد سپس نرخ نمونه‌گیری مبدل را تعیین کرده و سپس A/D روشن گردد و متغیر برگرداندن دما را در شبکه تعریف کند و بعد از اتمام تبدیل A/D اطلاعات را انتشار دهد.

```
#include <stdlib.h>
#include "a2d.h"

//   Declare node-level self-documentation

#pragma set_node_sd_string "&3.0@1Temp Snsr"

//   Declare sensor output network variable

network output sd_string("@1|1") SNVT_temp nvoValue;

//   Declare sensor configuration parameters

config network input sd_string("&0,5,0\x80,26") SNVT_temp nciOffset;
config network input sd_string("&0,1,0\x80,31") SNVT_muldiv nciGain;
config network input int nciSampleRate;

//   Reset task - initialize A/D converter

when( reset ) {
    a2d_enable(nciSampleRate);
    a2d_mux(0);
}

//   A/D conversion complete task - propagate network variable

when( a2d_done() ) {              // fixed point linear scaling
    nvoValue = muldiv(a2d_read(), nciGain.multiplier, nciGain.divisor) + nciOffset;
}
```

نمونه این کدها در سایت شرکت Echelon که سازنده Neuron® C Programmerمی‌باشد موجود است. پروتکل Lonworks این امکان را می‌دهد که مهندسان روی طراحی بهترین کاربرد ممکن متمرکز شوند و به پروتکل ارتباطی، سخت‌افزار و نرم‌افزار ارتباطی یا سیستم‌عامل توجهی نداشته باشند و بتوانند تعداد عظیمی از قطعات را انتخاب کنند. همچنین اضافه کردن، جابجایی و تعویض اجزا را برای کنترل شبکه در هر زمان آسان‌تر می‌کند و نودهای روی یک شبکه تقریباً نامحدود است.

در شبکه Lonworks در لایه فیزیکی امکان استفاده از ۴ روش وجود دارد که عبارت‌اند از: استفاده از زوج سیم به هم تابیده، استفاده از خطوط انتقال قدرت AC/DC، فیبر نوری و ارتباط بی‌سیم.

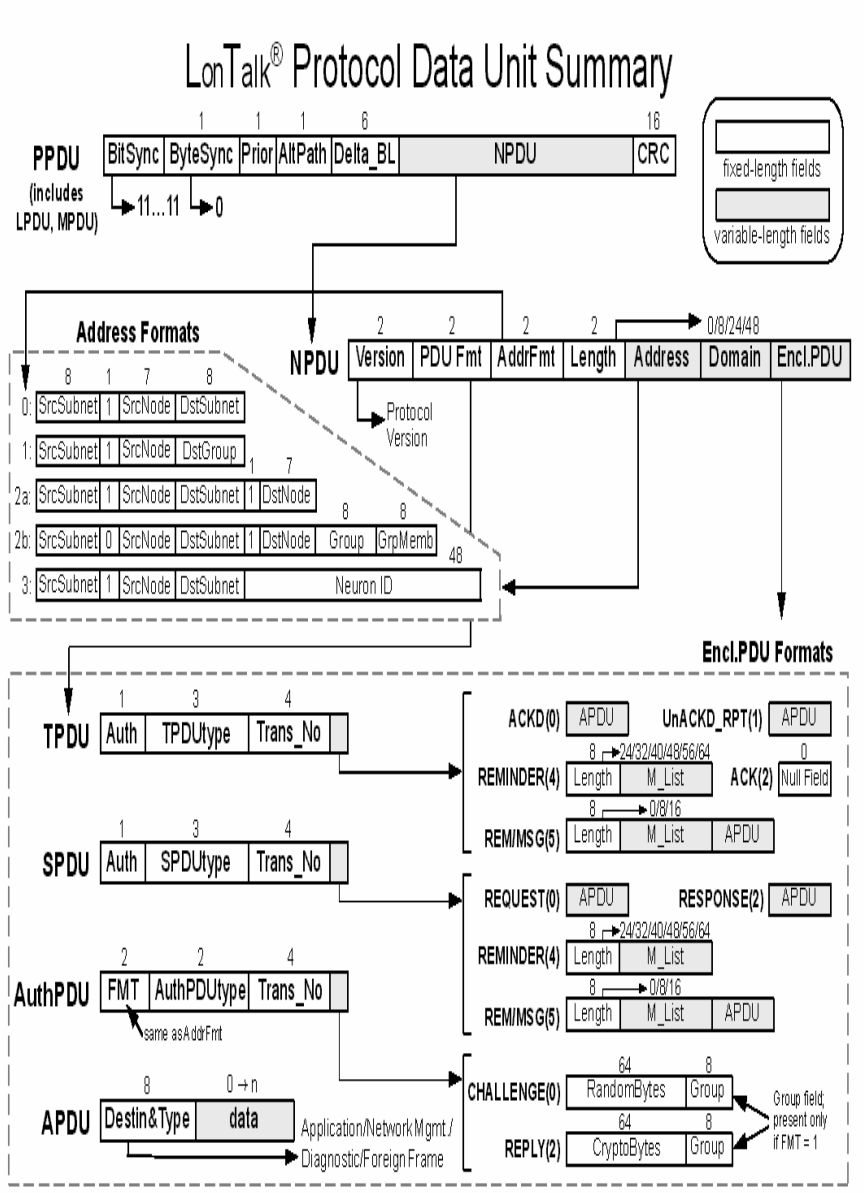


شکل ۱-۳-۳ اتصال مستقیم به زوج سیم به هم تابیده در مد تفاضلی

در شبکه Lonworks بنا به تشخیص طراح سیستم، هر کدام از رسانه‌های فوق و یا ترکیبی از آنها می‌تواند در سطح لایه فیزیکی مورد استفاده قرار گیرد و لزومی به استفاده از یک لایه فیزیکی یکسان در کل قطار نمی‌باشد.

شرکت Echelon محصولات متنوعی از قبیل کارت‌های شبکه، کارت‌های دیجیتال ورودی و خروجی، کارت‌های آنالوگ ورودی و خروجی و ... جهت استفاده در شبکه Lonworks ارائه می‌کند.

پروتکل Lonworks



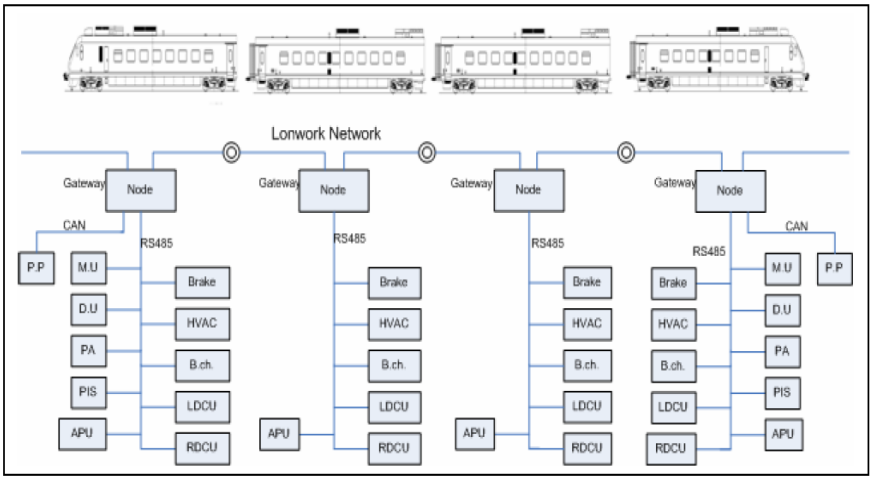
شکل ۱-۳-۴ پروتکل Lonworks

فرمت ارسال پیغام به شبکه Lonworks		
اسم بیت‌ها یا بایت‌ها	اندازه	هدف از ارسال بیت یا بایت
Preamble bit sync	bit + N۶	Receiver Synchronization (N is configurable)
Preamble byte sync	bit۱	Receiver synchronization
Log Data	bytes۲	Definition of the message type
Source address	bytes۲	Addressing
Destination	bytes۷-۱	Addressing
Domain	bytes۶-۰	Addressing
Message Type	bit۲	Definition that network variable is transmitted
Network Variable Selector	bit۱۴	Logical identifier
Data	bytes۳۱to ۱	Payload
CRC	bit۱۶	CRC
Line Code Violation	bit۵.۲	Violation of the Manchester code
فرمت ارسال پیغام صریح در شبکه Lonworks		
Preamble bit sync	bit + N۶	Receiver Synchronization (N is configurable)

Preamble byte sync	bit۱	Receiver synchronization
Log Data	bytes۲	Definition of the message type
Domain	bytes۶-۰	Addressing
Source address	bytes۲	Addressing
Destination	bytes۷-۱	Addressing
Message Type	bit۱	Definition Explicit Message
Message code	bits۷	Definition News Service
Data	۲۲۸ to ۱From bytes	Payload
CRC	bit۱۶	CRC
Line Code Violation	bit۵.۲	Violation of the Manchester code

همان طور مشخص است این پروتکل از پیچیدگی خاص خود برخوردار است لیکن در زمان کنونی و در سطح طراحی interfase با توجه به وجود Neuron® C Programmer و نورن چیپ‌های سازندگان مختلف برنامه‌نویس درگیر ایجاد پروتکل نخواهد شد و تنها با برنامه‌نویسی C به‌راحتی سنسورها و محرک‌ها را می‌تواند به شبکه متصل کند.

پیاده سازی Lonworks در قطار



شکل ۱-۳-۵ نمونه پیاده سازی یک پروتکل Lonworks در یک رام قطار (ریل بأس ارم)

به دلیل اهمیت و حجم اطلاعات بین سیستم کنترل یا power pack و سیستم کنترل و مانیتورینگ قطار (TCMS) از ۰.۲CAN با سرعت ۲۵۰ KB/S استفاده شده و سیستم ترمز با سیستم کنترل قطار توسط ۴۸۵RS و با نرخ ارسال اطلاعات kb/s۲.۱۹ و سیستم‌های تهویه، کنترل درب‌ها، سیستم اطلاع‌رسانی صوتی (PA)، سیستم اطلاع‌رسانی نوشتاری (PIS)، سیستم کنترل باطری شارژر با نرخ ارسال KB/S۶.۹ می‌باشند.

برای پیاده سازی شبکه Lonworks کارت‌هایی با تراشه AFG۱B۳۱۵۰TMPN شرکت توشیبا توسط واگن‌ساز کره‌ای شرکت HYUNDAI ROTEM ساخته شده است. در این حالت نرم‌افزار سیستم مانیتورینگ توسط نرم‌افزار سطح بالای رایج برنامه‌ریزی می‌گردد.

مقایسه استانداردهای TCN و Lonworks

هر دو استاندارد TCN و Lonworks به‌منظور استانداردسازی انتقال اطلاعات در واگن‌ها تدوین شده‌اند اما استاندارد TCN صرفاً جهت استفاده قطارها کاربرد دارد و پروتکلی باز نیست ولی استاندارد Lonworks یک استاندارد عمومی و باز می‌باشد در پروتکل Lonworks مدرکی تحت عنوان "Lon talk Protocol Specification" در اختیار کاربر وجود دارد و تراشه‌های آن در اختیار است و می‌توان این را در هر پروسسور پیاده سازی کرد. یکی از کاربردهای Lonworks کاربرد در صنایع ریلی می‌باشد و به همین دلیل و با توجه به بازار مصرف محدودتر پروتکل TCN و عدم وجود توجیه اقتصادی هنوز شرکت‌های سازنده تراشه‌ها به فکر طراحی تراشه خاص پروتکل TCN نیفتاده‌اند؛ اما در مقابل با توجه به عمومی بودن پروتکل Lonworks شرکت‌های توشیبا و موترولا و CYPRESS آمریکا تراشه‌های خاص را که تمام ویژگی‌های Lonworks را پشتیبانی کند ساخته و در اختیار عموم قرار داده‌اند. تاکنون بیش از ده‌ها میلیون از تراشه‌های Lonworks معروف به Neurons تولید شده و مورد استفاده قرار گرفته است.

در سال‌های اخیر شرکت EKE یک نمونه سخت‌افزار کامل تحت عنوان TCN Gateway به بازار عرضه کرده که شامل ماژول‌های CPU و WTB و

MVB می‌باشد و منطبق بر ۶۱۳۷۵IEC بوده و با زبان برنامه‌نویسی ISaGRAF منطبق بر زبان برنامه‌نویسی PLCها برنامه‌ریزی می‌گردد.

شکل ۱-۴-۱ TCN Gateway شرکت EKE

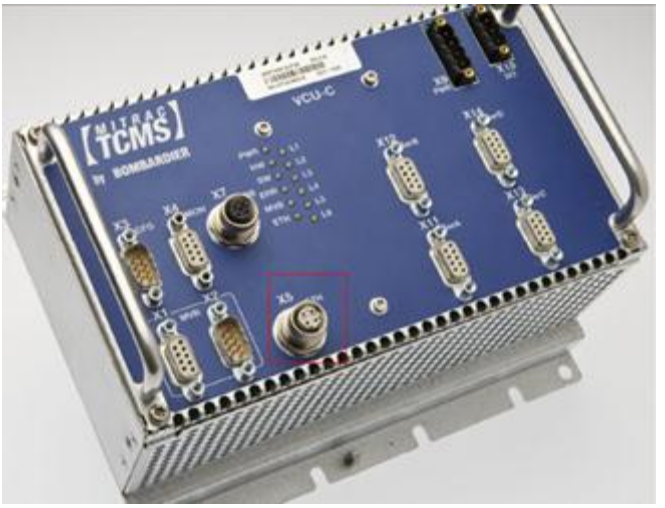
از آن سو اختصاصی بودن TCN که سبب امکان کوپل شدن انواع واگن‌ها و تشکیل رام قطار را می‌دهد و امکان اتصال لوکوموتیوهای مختلف با سیستم TCN در کشورهای اتحادیه اروپایی به یک رام قطار یک ویژگی مطلوب این پروتکل است.

با توجه پروتکل باز Lonworks می‌توان از این استاندارد جهت طراحی و ساخت TCMS بومی استفاده کرد ولی در MVB محدود به سخت‌افزار چند سازنده محدود در سطح جهان خواهیم شد که در وضع کنونی مناسب نیست. البته در استاندارد TCN پروتکل CAN نیز قابل‌استفاده است که در تأمین تراشه‌ها و تجهیزات آن محدودیت کمتری وجود دارد.

شبکه Ethernet

مقدمه

در اجزای سیستم کنترل و مانیتورینگ معمولاً چند لینک Ethernet, MVB, WTB وجود دارد، بسیاری از تجهیزات سایر سیستم‌های قطار شهری نیز پورت Ethernet را دارند این Ethernet معمولاً از نوع Base-T۱۰۰/۱۰ است.



شکل ۱-۵-۱ نمونه‌ای تجهیزات TCMS همراه پرت Ethernet

Summary of technical data	
Interface	MVB, 2 Ethernet, Serial x 2, USB

Summary of technical data	
Standard Interfaces:	
Ethernet	2 x 10/100 Base-T

تاریخچه Ethernet

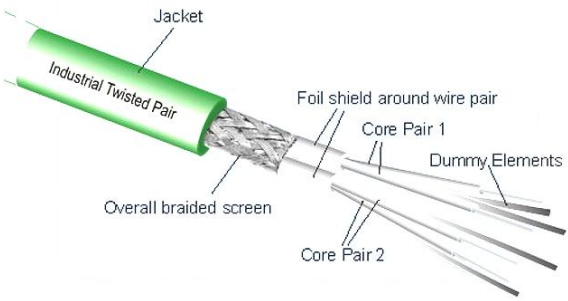
شبکه Ethernet در سال ۱۹۸۰ از سوی شرکت‌های DEC، Intel، Xerox ارائه گردید مشخصه اترنت همان وظایف لایه‌های پیوند داده‌های و فیزیکی مدل OSI در ارتباطات داده‌ای را انجام می‌دهد این طرح اساس مشخصه ۳.۸۰۲IEEE می‌باشد. در سال ۱۹۹۰، ۳.۸۰۲IEEE برای اجرا بر روی سیم زوج تابیده شده (Twisted Pair) منتشر شد که پس از انتشار آن به استاندارد Base T۱۰ مشهور شد.

مشخصات استاندارد Base T۱۰

از کابل UTP و یا زوج تابیده شده نوع شیلد دار STP و در کاربردهای صنعتی از ITP (Industrial Twisted Pair) استفاده می‌شود. دو زوج سیم امپدانس مشخصه ۱۰۰ اهم و سطح مقطع ۰/۴ تا ۰/۶ میلی متر برای هادی‌های خود دارند. با توجه به فرکانس کاری و با توجه به مشخصات کابل‌ها که در زیر آمده کابل ۵CAT کابلی مناسب در این استاندارد است.

Cate5 (UTP)

Cate5 (STP)



(ITP)Cate۵

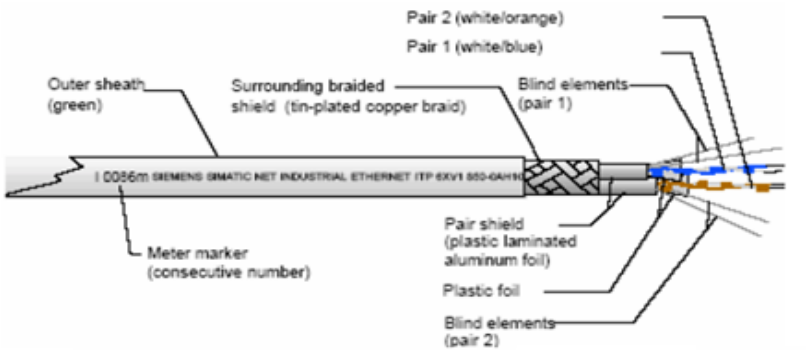
شکل ۱-۵-۲ نمونه‌ای کابل‌های Cate۵

Specification of Various Cable Categories				
Parameter	Category 5	Category 5E ('568-A-5)	Category 6	Category 7
Specified frequency range	1-100 MHz	1-100 MHz	1-250 MHz	1-600 MHz
Attenuation	24 dB	24 dB	21.7 dB (36 dB)	20.8 dB (54.1 dB)

رنگ‌های سیم‌های کابل‌های UTP و STP به صورت زیر است:

GRN , WHT/GRN	سبز – سفیدسبز
ORG , ORG/WHT	نارنجی – سفید نارنجی
BRN , BRN/WHT	قهوه ای –سفید قهوه ای
BLU , BLU/WHT	آبی –سفید آبی

در سیم ITP دو زوج سیم به رنگ‌های آبی/ سفید و نارنجی/ سفید وجود دارند. مشخصات الکتریکی این کابل به صورت زیر است.



شکل ۱-۵-۳ نمونه‌ای از سیم ITP

Cable categories EN 50173	CAT5		
DC loop resistance		maximum	124 ohm/km
DC insulation resistance		minimum	5Gigaohm/km
Attenuation/100 m	at 4 MHz 10 MHz 100 MHz	maximum	3.6 dB 5.7 dB 18.0 dB
Near end crosstalk loss (NEXT)/100m	at 1 --300 MHz	minimum	80 dB
Characteristic impe-dance	at 1 --100 MHz 100 --300 MHz		100 ohm+/- 15% 100 ohm+/- 45%
Transfer impedance	at 10 MHz	maximum	2m ohm/m
Structural return loss	at 1 --100 MHz 100 --300 MHz	minimum	23 dB 15 dB
Longitudinal conversion loss		minimum	43 dB
Capacitance unbalance pair to ground		maximum	3400 pF/km
Dielectric strength at 50 Hz -conductor/conductor -conductor/shield	 1min 1min	effective value	 700 V 700 V

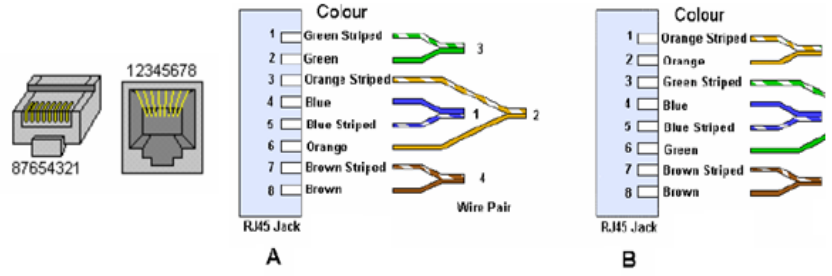
مشخصات استاندارد Base T۱۰۰

سرعت انتقال در این استاندارد Mbps۱۰۰ است. این استاندارد ۱۰ برابر سریع‌تر از Base T۱۰ است و به‌عنوان Fast Ethernet از آن یاد می‌شود. در این روش از دو کابل Twisted-Pair با کیفیت بسیار بالا همچون ITP استفاده می‌کند. بهتر است کابل در این پروتکل از کابل یک سطح بالاتر یعنی از نوع ۶CAT یاe۵ CAT انتخاب شود.

شکل ۱-۵-۴ نمای ظاهری Cate۵Cate ,۶

سیم بندی شبکه، Base T۱۰ Base T۱۰۰

برای اتصال کابل‌ها به ایستگاه‌ها و تجهیزات هاب، از اتصالات ۴۵Rj یا ۱۲M استفاده می‌شود در ۴۵Rj دو روش برای سیم بندی به نام‌های A,B وجود دارد.

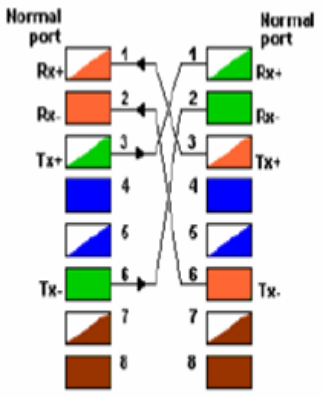


شکل ۱-۵-۵ استاندارد A,B در سیم بندی

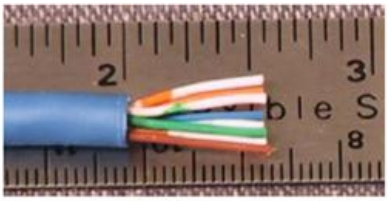
شماره پین	وظیفه	رنگ سیم طبق استاندارد A	رنگ سیم طبق استاندارد B
۱	ارسال کننده +	سفید سبز	سفید نارنجی
۲	ارسال کننده -	سبز	نارنجی
۳	دریافت کننده +	سفید نارنجی	سفید سبز
۶	دریافت کننده -	نارنجی	سبز
۴ و ۵ و ۷ و ۸	بدون استفاده	آبی، سفید آبی، سفید قهوه‌ای و قهوه‌ای	

در مواردی که قرار است دو وسیله مستقیماً به هم متصل شوند مثل اتصال دو هاب به یکدیگر، اتصال باید از نوع متقاطع (Cross) باشد. در این اتصال در یک طرف روش سیم بندی A و در طرف دیگر B است؛ به عبارت دیگر سیم‌های ارسال کننده به دریافت کننده متصل است.

1 WHT/ORG	1 WHT/GRN
2 ORG/WHT	2 GRN/WHT
3 WHT/GRN	3 WHT/ORG
4 BLU/WHT	4 BLU/WHT
5 WHT/BLU	5 WHT/BLU
6 GRN/WHT	6 ORG/WHT
7 WHT/BRN	7 BRN/WHT
8 BRN/WHT	8 WHT/BRN



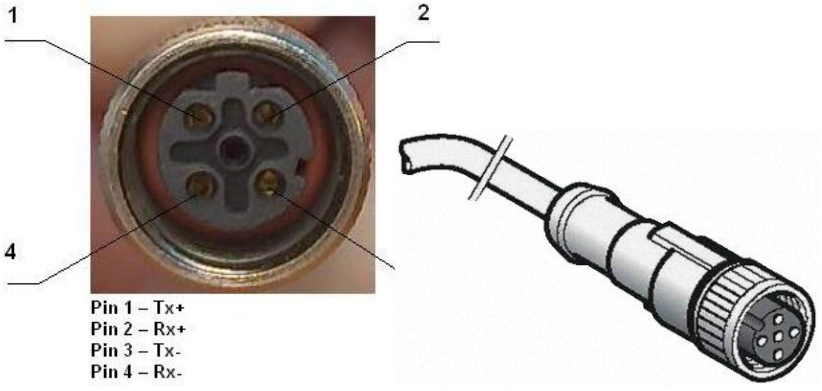
در حدود ۱ سانتی‌متر سیم‌های به هم تابیده را باز کرده و به صورت یکی از دو استاندارد قرار می‌دهیم و کانکتور ۴۵Rj را بدان پرس می‌کنیم. در این روش نیازی به لخت کردن سیم‌ها نیست.



شکل ۱-۵-۶ طریقه سوکت زدن کانکتور ۴۵RJ

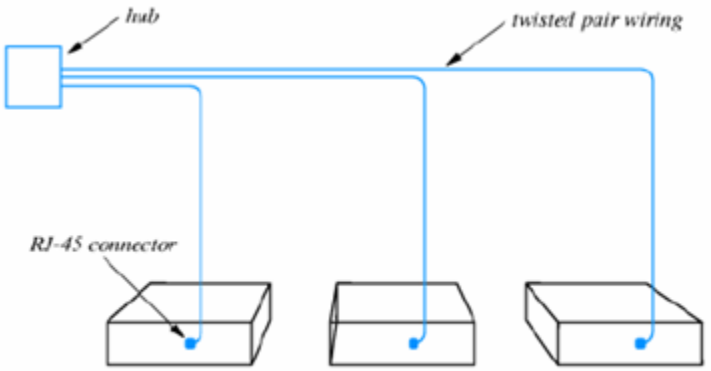
از کانکتور ۱۲M در محیط‌های صنعتی و برای شرایط IPبالا استفاده می‌شود این کانکتور به شکل زیر است. در این کانکتور نیز پین‌های ارسال کننده و دریافت کننده به صورت زیر وجود دارد. کابل مناسب برای آن نیز ITP است.

شکل ۱-۵-۷ کانکتور ۱۲M



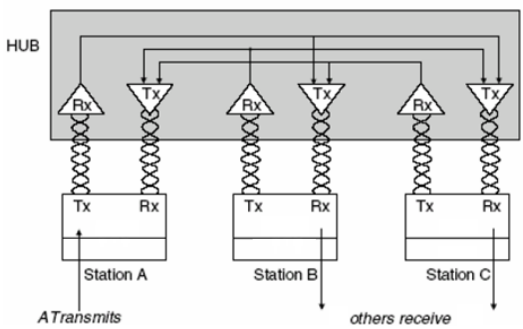
توپولوژی شبکه Base T۱۰ و Base T۱۰۰

شبکه Base T۱۰ و Base T۱۰۰ به ستاره بسته می‌شود مزیت این روش نسبت به روش اتصال خطی این است که با قطع یک کابل کل شبکه قطع نمی‌شود و فقط همان وسیله ارتباطش قطع می‌شود وسیله‌ای که کار ایجاد شبکه ستاره را بر عهده دارد HUB نام دارد.



شکل ۱-۵-۸ توپولوژی ستاره

تجهیز Hubسیگنال ورودی را تقویت کرده و بر روی بأس داخلی مشترکی قرار می‌دهد و این امکان را همراه با این سیگنال بر روی چند خروجی منتشر شود. خاصیت تقویت سیگنال نیاز به تکرار کننده (Repeater) را در شبکه از بین می‌برد.



شکل ۱-۵-۹ شماتیک مدار داخلی هاب

این تجهیز دارای ۴ تا ۲۴ خروجی می‌توانند باشند، در Base T۱۰ حداکثر فاصله تجهیزات تا هاب ۱۰۰ متر و حداقل آن ۲/۵ متر است، برای توسعه شبکه می‌توان از اتصال هابها به یکدیگر استفاده نمود. به صورت تئوری در صورت سری کردن Hubها حداکثر فاصله تجهیزات از هم تا ۴۰۰۰ متر قابل افزایش است. اتصال Hubها به یکدیگر نیز از طریق پورت‌هایی به نام uplink ممکن است. همان طور که ذکر شد اتصال دو هاب به یکدیگر، اتصال (Cross) و اتصال Hub به تجهیز اتصال یک به یک متناظر می‌باشد.

حداکثر طول شبکه نیازمند انجام محاسباتی به نام‌های PPV, span است.

محاسبه span :آن دو گره‌ی که در شبکه بیشترین فاصله را با هم دارند، باید تأخیری مجاز داشته باشند در غیر این صورت احتمال تصادم اطلاعات (collision) بالا می‌رود، این تأخیرها در رابطه زیر صدق کند.

Repeater Delays+ Cable Delay+ NIC Delays + Safety Factor (۵ bits minimum) <۲.۵۶ ms

جدول تأخیر بعضی المان‌ها در شبکه به صورت زیر است:

Component	Maximum Delay (µs)
Fast Ethernet NIC	0.25
Fast Ethernet Switch Port	0.25
Class I Repeater	0.7 max
Class II Repeater	0.46 max
UTP Cable (per 100 m/330 ft)	0.55
Multimode Fibre (per 100 m/330 ft)	0,50

محاسبهPPV : طبق ۳.۸۰۲IEEE برای جلوگیری از تصادم در شبکه، باید دو بسته متوالی دیتا یک فاصله زمانی مشخص رعایت شود، هر وسیله دارای یک Variability Value با واحد bit time است. به مجموع Variability Value وسایل که معرف فواصل زمانی است Path Variability Value(PPV) می‌گویند. PPV کل شبکه نباید بیشتر از ۴۰ bit timeشود به عبارت دیگر فاصله زمانی بین بسته‌ها مجموعاً کمتر از ۴۰ bit timeباید باشد.

تجهیز Hub به منابع تغذیه با قابلیت اطمینان بالا نیازمند هستند چون هر گونه اختلال در تغذیه سبب قطع شبکه خواهد شد.

پیاده سازی لایه سخت‌افزاری ۱۰Base T و ۱۰۰Base T:

در قطارهای شهری اگر سیستم کنترل و مانیتورینگ با یک کامپیوتر صنعتی یا PLC پیاده سازی شده باشد کارت شبکه لایه سخت‌افزاری شبکه ۱۰۰/۱۰Base T خواهد بود. همچنین امکان پیاده سازی لایه سخت‌افزاری توسط برخی تراشه‌ها همچون ۳۱۰۰AS, W۸۰۱۹,RTL۶۰j۲۸ENC + ۸۲۰۱RTL در کنار یک میکروکنترلر یا DSP و از طریق پرت SPI نیز وجود دارد.

شکل ۱-۵-۱۰ کارت Ethernet

ساختار فریم اترنت

ساختار فریم در لایه Data Link، تقریباً برای تمامی سرعت‌های اترنت (از ده تا ده هزار مگابیت در ثانیه) یکسان می‌باشد. این وضعیت در لایه فیزیکی وجود نداشته و هر یک از نسخه‌های اترنت دارای یک مجموعه قوانین جداگانه و مختص به خود می‌باشند.

مقدمه ۷	محاسبه FCS					FCS ۴
شروع ۱	مقصد ۶	مبدأ ۶	طول / نوع ۲	Pad | Data ۴۶ تا ۱۵۰۰		

۱. مقدمه (۵۶ بیت معادل هفت octet)

۲. شروع فریم (۸ بیت معادل یک octet)

۳. آدرس MAC مقصد (۴۸ بیت معادل شش octet)

۴. آدرس MAC مبدأ (۴۸ بیت معادل شش octet)

۵. طول / نوع (۱۶ بیت معادل دو octet)

درصورتی‌که مقدار موجود در این فیلد کمتر از ۰۶۰۰ (مبنای شانزده) باشد، مقدار طول و در غیر این صورت نوع پروتکل مشخص می‌گردد.

۶. داده (بین ۳۶۸ تا ۱۲۰۰۰ بیت، معادل چهل‌وشش تا یک هزار و پانصد octet)

درصورتی‌که مقدار موجود در این فیلد کمتر از چهل‌وهشت octet باشد، باید یک Pad به انتها اضافه گردد.

۷. FCS (سی‌ودو بیت معادل چهار octet)

فیلدهای فریم اترنت

برخی فیلدهای ضروری در فریم‌های استاندارد ۳. ۸۰۲ عبارت‌اند از:

۳.۸۰۲IEEE						
مقدمه	شروع	مقصد	مبدأ	طول / نوع	Pad | Data	FCS
۷	۱	۶	۶	۲	۴۶ تا ۱۵۰۰	۴
Ethernet						
مقدمه		مقصد	مبدأ	طول / نوع	Pad | Data	FCS
۸		۶	۶	۲	تا ۱۵۰۰	۴

۱. **مقدمه،** یک الگوی متناوب مشتمل بر مجموعه‌ای از صفر و یک است که از آن برای همزمانی در سرعت‌های ده مگابیت در ثانیه و یا سرعت‌های پائین تر استفاده می‌شود. با توجه به این که نسخه‌های سریع‌تر اترنت هم‌زمان

می‌باشند به اطلاعات فوق نیاز نبوده و صرفاً جهت سازگاری با نسخه‌های قبلی استفاده می‌گردد.

Preamble Filled
۱۰۱۰۱۰۱۰ ۱۰۱۰۱۰۱۰ ۱۰۱۰۱۰۱۰ ۱۰۱۰۱۰۱۰ ۱۰۱۰۱۰۱۰ ۱۰۱۰۱۰۱۰ ۱۰۱۰۱۰۱۰

۲. **شروع فریم یا SFD** (برگرفته از Start Frame Delimiter)، از هشت بیت تشکیل شده است و مسئولیت آن مشخص کردن انتهای اطلاعات مربوط به زمان‌بندی است. الگوی فوق به صورت ۱۰۱۰۱۰۱۱ می‌باشد.

۳. **آدرس مقصد،** شامل آدرس MAC مقصد است. آدرس مقصد می‌تواند به صورت تکی (Uncast)، گروهی (Multicast) و یا برای تمامی گره‌ها (broadcast) باشد.

۴. **آدرس مبدأ،** شامل آدرس MAC مبدأ است. آدرس مبدأ همواره به صورت تکی (Unicast) بوده و آدرس گره ارسال کننده اطلاعات را مشخص می‌نماید.

۵. **طول / نوع،** برای دو هدف متفاوت استفاده می‌گردد. درصورتی‌که مقدار این فیلد کمتر از ۱۵۳۶ (مبنای ده) و یا ۶۰۰x۰ (مبنای شانزده) باشد، طول را مشخص می‌نماید. از فیلد فوق به عنوان "طول" زمانی استفاده می‌گردد که مسئولیت مشخص کردن پروتکل استفاده شده بر عهده لایه LLC باشد. مقدار موجود در این فیلد به عنوان "طول"، تعداد بایت‌های داده را مشخص می‌نماید.

درصورتی‌که مقدار این فیلد به عنوان "نوع" در نظر گرفته شود، پروتکل لایه بالاتر که پس از تکمیل پردازش اترنت داده را دریافت می‌نماید، مشخص می‌گردد.

۶. **داده و Pad**، هر طولی را می‌تواند داشته باشد مشروط به این که از حداکثر اندازه فریم تجاوز ننماید. حداکثر اطلاعاتی را که می‌توان در هر مرتبه ارسال نمود، یک هزار و پانصد octet می‌باشد. درصورتی‌که داده موجود در فیلد "داده" به حداقل مقدار لازم (چهل‌وشش octet) نرسیده باشد، باید از Pad استفاده گردد.

۷. **FCS**، از چهار octet تشکیل و شامل مقدار CRC است که توسط دستگاه فرستنده محاسبه و توسط دریافت کننده به منظور تشخیص بروز خطا در زمان ارسال اطلاعات، مجدداً محاسبه می‌گردد. با توجه به این که خرابی صرفاً یک بیت از ابتدای فیلد "آدرس مقصد" تا انتهای فیلد "FCS" باعث محاسبه Checksum متفاوتی خواهد شد، تشخیص این موضوع که اشکال مربوط به فیلد FCS و یا سایر فیلدهای شرکت کننده در محاسبه CRC است را غیر ممکن می‌نماید.

پروتکل ارتباطی CAN

مقدمه

همان‌طور که اشاره شد در استاندارد MVB،۱-۶۱۳۷۵IEC به عنوان Vehicle bus استاندارد برای استفاده در معرفی شده است؛ اما از باس‌ها CAN(Control Area Network) نیز می‌توان در این سطح استفاده کرد. امروزه استفاده از این بأس در قطارهای شهری بسیار رایج است. پروتکل CAN توسط شرکت بوش ابداع و با همکاری اینتل بازسازی و تکمیل شد. در سال ۱۹۸۷ شرکت اینتل اولین تراشه‌ی CAN موسوم به تراشه‌ی ۸۲۵۲۶ را ساخت که بعدها در تراشه‌ی ۸۲۵۲۷ تکمیل شد. در همان زمان کارخانه‌ی نیمه هادی فیلیپس نیز اقدام به ساخت تراشه‌ی ۲۰۰C۸۲ برای کنترلر CAN نمود. پس از آن شرکت‌های موتورولا و NEC نیز اقدام به ساخت تراشه‌های CAN نمودند. در سال ۱۹۹۲ گروه CAN in Automation(CiA) تشکیل شد و در سال بعد استاندارد ۱۱۸۹۸ISO جهت تعریف CAN برای استفاده‌های صنعتی انتشار یافت. پروتکل ارتباطی شبکه کنترل کننده CAN ابتدا برای استفاده در ماشین‌های حمل و نقل طراحی شد. به دنبال استفاده موفقیت‌آمیز و همچنین مزایای زیر استفاده از این روش در صنایع اتوماسیون مانند وسایل کنترل کننده، سنسورها و تحریک کننده‌ها، استفاده از روش CAN رواج پیدا کرد.

- در جهت طراحی و پیاده سازی مقرون به صرفه هستند.
- قابل اطمینان در محیط‌های پر نویزو اغتشاش می‌باشند.
- به سادگی قابل پیاده سازی و پیکر بندی هستند.
- یک محیط مرکزی جهت تشخیص و خطاها در هنگام طراحی و یا زمان بهره‌برداری از آنها فراهم می‌شود؛ به عبارت دیگر در زمان طراحی یا به‌کارگیری از این شبکه می‌توان در یک مرکز مشخص خطاهای احتمالی کارکرد آنها را پیگیری کرد.

به دنبال استفاده روزافزون از این روش، در جهت توسعه و تکمیل آن پروتکل‌های دیگری مانند CAL based سیستم باز CAN و... از بطن روش اصلی، طراحی و پیشنهاد شده است.

معرفی شبکه‌ی CAN

CAN مخفف شبکه محلی کنترلر است. اساساً این شبکه برای محیط‌های پر نویز صنعتی طراحی شده است. CAN-BUS رابط دو سیمِ تفاضلی است که روی یک جفت سیم به‌هم‌پیچیده شده محافظ دار ((STP یا بدون محافظ ((UTP یا کابل تخت ((Ribbon cable به همراه سیم زمین اجرا می‌شود و به این سیم‌ها CAN_L و CAN_H گفته می‌شود. تعداد وسایل قابل اتصال ۱۱۰ وسیله است. توپولوژی به‌صورت خطی (بأس) است که دو طرف آن ترمیناتور نیاز دارد. برای هر گره از یک کانکتور نوع Pin D Male۹ استفاده می‌شود. نحوه‌ی رمز گذاری بیتی برای گذرگاه دیفرانسیلی دو سیمِ به صورت بدون بازگشت به صفر با یک بیت Stuffing یا (Non Return to Zero (NRZ)) است. استفاده از رمز گذاری بدون بازگشت به صفر ارسال پیغام‌های ترکیبی را با

حداقل تعداد انتقال و اطمینان بالا برای اغتشاش‌های خارجی، تضمین می‌کند.

Pin #	Signal Names	Signal Description
۱	Reserved	Upgrade Path
۲	CAN_L	Dominant Low
۳	CAN_GND	Ground
۴	Reserved	Upgrade Path
۵	CAN_SHLD	Shield, Optional
۶	GND	Ground, Optional
۷	CAN_H	Dominant High
۸	Reserved	Upgrade Path
۹	CAN_V+	Power, Optional

پین‌های کانکتور خروجی CAN Bus

با به کارگیری این روش، کنترل کننده‌ها، سنسورها و محرک‌ها با یکدیگر به وسیله تنها دو سیم ارتباط برقرار می‌کنند. پروتکل CAN برای رسانیدن پیغام‌های کوتاه با طول حداکثر ۸ بایت طراحی شده است.

این پروتکل هیچ گونه وقفه‌ای برای انتقال پیغام ندارد اما اولویت فرستادن پیغام را برای عدم برخورد خواهد داشت و معمولاً پیغام اورژانسی را در اولویت قرار می‌دهد. سیستم‌های CAN بسیار سریع هستند و قابلیت انتقال حداکثر ۷۶۰۰ پیغام ۸ بایتی و ۱۸۰۰۰ سیگنال راه‌انداز در ثانیه را دارا خواهند بود.

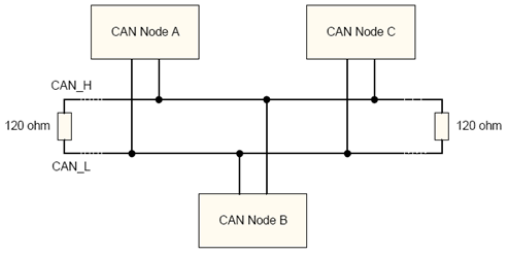
بالاترین نرخ ارسال داده در این پروتکل ۱Mbps و کمترین آن ۱۰Kbps می‌باشد. تمام ماژول‌های استفاده شونده در این پروتکل باید حداقل نرخ ارسال ۲۰Kbps را پشتیبانی کنند. طول کابل به نوع ارسال استفاده شده بستگی دارد. به‌طورمعمول همه‌ی قطعات استفاده شده در سیستم انتقال باید دارای نرخ انتقال یکسان باشند. حداکثر طول خط انتقال یک کیلومتر و حداقل آن ۴۰ متر در نرخ ۱Mbps است. بیشترین زمان انتقال در فریم ۸ بایتی با ۱۱ بیت شناسایی، ۱۳۴ مرتبه‌ی بیتی است (یعنی ۱۳۴ میکرو ثانیه در حداکثر نرخ انتقال ۱Mbps).

طول کابل	سرعت
m۴۰	Mbps۱
m۲۵۰	Kbps۲۵۰
m۵۰۰	Kbps۱۲۵
Km۱	Kbps۱۰

ماکزیمم نرخ بیت بر حسب طول بأس

مقاومت‌های ترمیناتور در انتهای هر خط قرار داده می‌شوند. مقاومت‌های ترمیناتور در حدود ۱۲۰ اهم است که در دوطرف بأس بین CAN_L و

CAN_H قرار می‌گیرد. نقش ترمیناتور تطبیق امپدانس و جذب سیگنال‌هاست. بدون وجود ترمیناتور وقتی سیگنال به دو سر سیم باز می‌رسد اکو شده و با دامنه‌ی معکوس روی سیم برگشت پیدا می‌کند. این سیگنال برگشتی شبیه نویز بوده و سیگنال‌های داده را خراب می‌کند. لذا نحوه سیم کشی در شبکه CAN به صورت زیر خواهد بود.



شکل ۱-۶-۱ نحوه سیم کشی در شبکه CAN

حالات سیگنال الکتریکی در بأس CAN

دو نوع حالت در توصیف بأس وجود دارد: حالت نهفته ((Recessiveو حالت غالب و برجسته (Dominant).

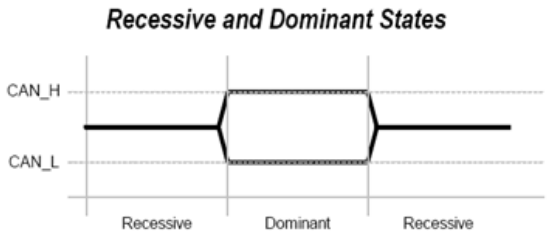
حالت نهفته هنگامی رخ می‌دهد که دو سیم CAN_L و CAN_H دارای پتانسیل یکسانی باشند.

حالت برجسته زمانی رخ می‌دهد که دو سیم با یکدیگر اختلاف پتانسیل داشته باشند.

بأس CAN وقتی غیر فعال است به صورت نهفته باقی می‌ماند.

تشخیص حالات بأس نهفته و برجسته از صفر و یک کار بسیار مهم و دشواری است. صفر و یک برای سیستم اعداد باینری مناسب است ولی آنها

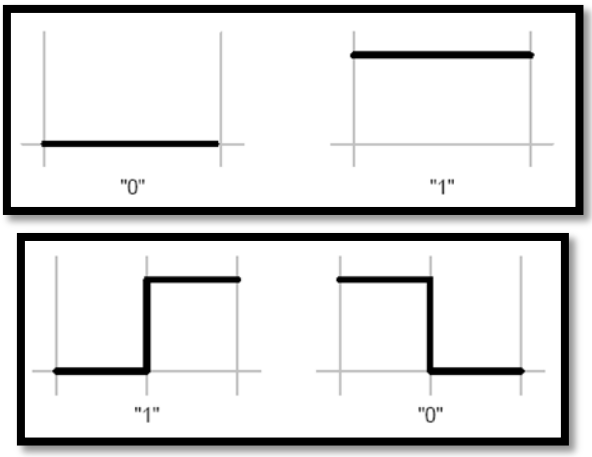
نمی‌توانند حالات بأس را بازگو کنند. این دو مفهوم نهفته و برجسته بأس یک مفهوم مهم و خاص در هنگام بحث داوری بأس و میدان کنترل CAN خواهد بود.



شکل ۱-۶-۲ حالات برجسته و نهفته بأس

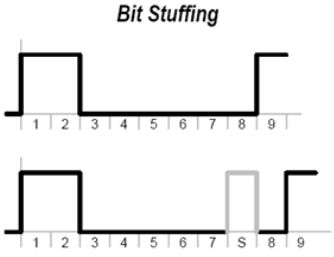
در پروتکل CAN برخلاف MVB دیگر از روش کدگذاری منچستر استفاده نشده و در آن از روش کدگذاری بدون بازگشت به صفر استفاده شده است. در این روش نیاز به نمایش هر بیت نیست و این سیگنال بازه‌ی زمانی ۰ یا ۱ ورودی را به قوت خود نگه می‌دارد.

اگر یک فریم شامل رشته‌ای از ۰ یا ۱ باشد این سیگنال بازه‌های زمانی زیادی را به صورت ثابت نگه می‌دارد. عیب این روش این است که برای تشخیص شروع و انتهای هر بیت راه آسانی وجود ندارد و برای رفع کردن این مشکل نیاز به استفاده ازکلاک هم فرکانس با فریم داریم تا بتواند آن رشته را رمز گشایی کند.



شکل ۱-۶-۳کدگذاری بدون بازگشت به صفر در مقایسه با کدگذاری بیتی منچستر

چون روش بدون بازگشت به صفر سیگنال‌ها را برای مدت زیادی نگه می‌دارد ممکن است اسیلاتورها از سنکرون بودن خارج شوند. پروتکل CAN یک سیگنال دیگر را به عنوان کلاک تولید می‌کند و در هر ۵ بیت یک بیت اضافی را وارد فریم می‌کند این بیت را به عنوان stuff bit می‌شناسیم. دریافت کننده از این بیت برای سنکرون سازی کلاک استفاده می‌کند. هر شاخه stuff bit مربوط به خود را می‌شناسد و اگر دریافت کننده ۵ بیت صفر یا یک را ببیند به آن اعتنا نمی‌کند و سراغ بیت بعدی می‌رود.



شکل ۱-۶-۴ قرار دادن Stuff bit بعد از پنج بیت برای سنکرون سازی

شاخه‌های CANاز دو روش برای سنکرون کردن کلاک هایشان استفاده می‌کنند: سنکرون سازی سخت و دوباره سنکرونی.

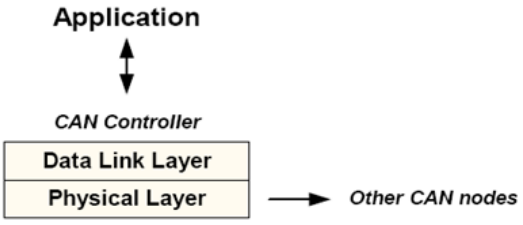
سنکرون سازی سخت تنها در زمان انتقال پیغام و معمولاً در ابتدای فریم پیغام جدید رخ می‌دهد. قبل از اینکه فریم شروع شود حالت بأس به صورت نهفته و غیر فعال است.

اولین بیت آغازگر فریم به صورت برجسته انتقال پیدا می‌کند. هر شاخه ساعت خود را با استفاده از انتقال ساخته شده توسط فریم سنکرون می‌کند. کلاک‌ها معمولاً قادر به سنکرون نگه داشتن خود در تمام طول فریم نیستند بنابراین باید دوباره سنکرون شوند.

دوباره سنکرون سازی هر بار که حالت بأس از نهفته به برجسته تغییر پیدا می‌کند رخ می‌دهد. اگر یک رشته از ۰ یا ۱ وجود داشته باشد آنگاه شاخه‌های CAN از انتقال ساخته شده توسط stuff bit برای دوباره سنکرون سازی کلاک هایشان استفاده می‌کنند.

شبکه CAN فقط از دولایه‌ی مدل مرجع استفاده می‌کند

شبکه CANچون فقط برای انتقال فرمان در شبکه (مثل پیغام‌های ساده برای شرایط راه‌انداز یا نمایش مقادیر دما و فشار) استفاده می‌شود، نیازی به ابزاری برای امنیت شبکه ندارد، به همین خاطر در این شبکه از ۲ لایه فیزیکی و لایه پیوند داده‌ها استفاده می‌شود.



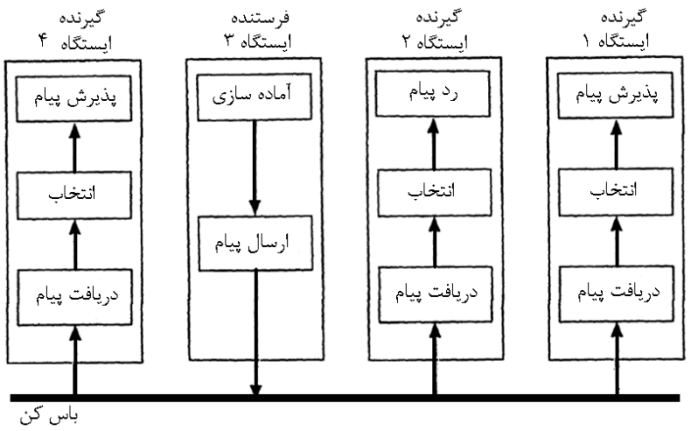
شکل ۱-۶-۵ شبکه CAN و استفاده از تنها دولایه‌ی مدل مرجع

لایه فیزیکی اطلاعات انتقال داده شده از لایه پیوند داده‌ها را تفسیر کرده و به سیگنال‌های الکترونیکی تبدیل می‌کند. لایه پیوند داده‌ها اقدام به ساختن فریم اطلاعات برای کنترل و نگه داشتن اطلاعات می‌کند. قسمتی از این اطلاعات کنترلی برای عدم ایجاد تداخل در شبکه هنگامی‌که چند پیغام در یک زمان ثابت در یک قسمت از شبکه فرستاده خواهند شد استفاده می‌شود. تابعی که منجر به این کار می‌شود کنترل دسترسی متوسط بأس نام دارد. این پروتکل نقشی مفید در عدم تداخل در شبکه خواهد داشت و فریم‌های اطلاعات را بر حسب اولویت به بأس می‌رساند.

مراحل تبادل داده‌ها

ارسال یک پیغام از هر ایستگاه به بأس و دریافت آن توسط ایستگاه‌های دیگر دارای چند مرحله است:

- Make ready: آماده سازی و ارسال دیتا و شناسه‌ها به چیپ CAN
- Send Message: بازسازی و ارسال پیغام توسط چیپ CAN به محض دریافت تخصیص بأس
- Receive Message: تمام ایستگاه‌های دیگر به عنوان گیرنده پیغام خواهند بود.
- Select: انتخاب، هر ایستگاهی که به درستی پیغام را دریافت کرده است بررسی می‌کند که آیا دیتای دریافتی مربوط به آن ایستگاه است؟
- Accept: در صورت دارای اهمیت بودن دیتا برای ایستگاه پردازش می‌شود.



شکل ۱-۶-۶ ارسال اطلاعات و پذیرش یا رد توسط ایستگاه‌ها

فرمت فریم‌ها در شبکه CAN

فریم یک بسته اطلاعاتی شامل یک پیغام کامل از انتقال‌دهنده می‌باشد. CAN دارای ۴ نوع فریم می‌باشد:

- **فریم داده**: یک پیغام استاندارد که برای انتقال اطلاعات درشبکه از آن استفاده می‌شود.
- **فریم راه دور**: یک پیغام با درخواست یک دریافت کننده، اطلاعات از شاخه دیگر با فاصله فرستاده خواهد داشت.
- **فریم خطا**: در مواقع از بین رفتن فریم یک پیغام فرستاده می‌شود و این خطا در فریم به فرستنده اطلاع می‌دهد تا دوباره پیغام فرستاده شود.
- **فریم بار زیاد**: این فریم شبیه به فریم خطاست. در هنگام زیاد شدن بار شبکه و به وجود آمدن خطا این فریم توسط گیرنده به فرستنده فرستاده خواهد شد تا پیغام بعدی با تأخیر فرستاده شود.

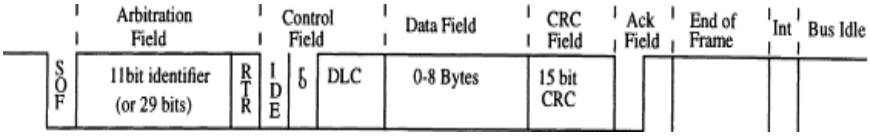
به دلیل اهمیت فریم داده در اینجا تنها به توضیح این فریم می‌پردازیم. همان‌طور که گفته شد این فریم برای انتقال اطلاعات به کار می‌رود:

فریم داده دارای دو فرمت اصلی برای ارسال پیغام است:

۱. فرمت استاندارد (A)۰.۲ که طول شناسه‌ی آن (کد شناسایی) ۱۱ بیت است، بنابراین طول فیلد داوری ۱۲ بیت است.

۲. فرمت طولانی یا توسعه‌یافته)B)۰.۲ که طول شناسه‌ی آن ۲۹ بیت است، بنابراین طول فیلد داوری ۳۲ بیت است.

معنی قسمت داوری از یک طرف تعیین اولویت پیام و از طرف دیگر آدرس منطقی اطلاعات است. این آدرس در استاندارد A۰.۲CAN تعداد ۲۰۳۲ آدرس منطقی مجاز وجود دارد. این موضوع بدان معنی است که تعداد ۲۰۳۲ موضوع مخابراتی مختلف (پیام) در این استاندارد CAN تعریف شده است. این تعداد پیام در استاندارد CAN توسعه یافته معادل ($۲^{۲۹}$) عدد می‌باشد.



شکل ۱-۶-۷ فریم A۰.۲CAN

بیکار	فاصله	انتهای فریم	پاسخ	خطا	Data	کنترل	RPT	هویت	شروع
۱	۷	۷	۲	۴	۰ تا ۶۴	۶	۱	۱۱	۱

توضیح این بیت‌ها در پیام‌های CAN به شرح زیر می‌باشد:

در این پیام، ابتدا بیت‌های Arbitration یا همان فیلد داوری ارسال می‌شوند. این بیت‌ها شامل بخش شناسه ID(کد شناسایی) و بیت RTR هستند. درصورتی‌که بیت RTR صفر منطقی باشد، معرف فریم داده‌ها و درصورتی‌که یک منطقی باشد، معرف درخواست فریم داده است.

- بیت شروع (۱ بیت): این بیت ابتدای هر پیام را مشخص می‌کند. پس از مدت رسانی که خط ارتباطی مورد استفاده قرار نگرفته و درحالت اصطلاحاً بیکار ((Idleمی‌باشد، لبه پایین رونده بیت شروع جهت سنکرون کردن (همزمان کردن) نقطه‌های مختلف شبکه مورد استفاده قرار می‌گیرد.

- بیت‌های تعیین هویت، ۱۱ بیت: آدرس منطقی و اولویت پیام را تعیین می‌کند. هر چه مقدار این پیام کمتر باشد. اولویت آن بیشتر است (عدد صفر دارای بالای اولویت است).

- بیت درخواست انتقال اطلاعات از راه دور (RTR)، ۱ بیت: این بیت مورد استفاده گیرنده برای در خواست اطلاعات از یک فرستنده و دیگر در راه دور مورد استفاده قرار می‌گیرد. اگر این بیت معادل ۱ (گرفتن) باشد، به معنی آن است که فریم اطلاعات صرف‌نظر از آنچه که کدهای دیگر مشخص می‌کنند، حاوی هیچ گونه اطلاعاتی نیست، در این حالت بقیه گره‌های شبکه چک می‌کنند که آیا اطلاعاتی جهت ارسال به نقطه‌ای که تقاضای اطلاعات دارد را دارند یا خیر، این تقاضای در سال و پاسخ احتمالی به آن و فریم متفاوت اطلاعاتی بر روی گذرگاه داده‌ها می‌باشند. این مسئله بدان معنی است که پاسخ این در خواست می‌تواند به دلیل وجود پیام‌های دیگر با اولویت ارتباطی بالاتر به تأخیر بیفتد.

- بیت‌های کنترل، ۶ بیت: اولین بیت این فریم بیت تعیین هویت است. اگر اولین بیت صفر منطقی باشد بدان معنی است که بیت‌های تعیین هویت دیگر فرستاده نخواهند شد و فریم اطلاعاتی یک فریم استاندارد CAN می‌باشد. بیت ro برای استفاده‌های بعدی رزرو شده و چهار بیت بعدی که تعیین کننده طول اطلاعات برای فریم اطلاعات همراه با این پیام را مشخص می‌کنند.

- بیت‌های اطلاعات،۰ تا ۶۴ بیت معادل ۵ تا ۸ بایت: محتویات اطلاعات پیام را منتقل می‌کنند.

- کد تشخیص خطا یا کد چرخشی، ۱۶ بیت: که مشخص کننده خطا در بیت‌های قبلی ارسال شده در پیام را نشان می‌دهد. این بیت‌ها تنها برای تشخیص خطا مورد استفاده قرار می‌گیرند و نمی‌توان از آن برای تصحیح خطا استفاده کرد. فاصله هم یک این کدها معادل ۶ است؛ یعنی می‌توان با استفاده از این کدها معادل ۶ بیت خطا را تشخیص داد که در کل پیام پراکنده شده یا خطاهای تجمعی (buist)تا ۱۵ بیت را مشخص می‌کنند.

- بیت تصدیق ACK، ۲ بیت: هر نقطه از شبکه که پیام درستی را بر روی سیم‌های انتقالی دریافت نمود در قسمت تأیید علامتی را جهت تأیید دریافت درستی پیام ارسال می‌دارد. این علامت به وسیله فرستنده پیام خوانده می‌شود و اگر این تأیید به توسط فرستنده خوانده نشد نشان‌دهنده بروز خطا در ارتباط می‌باشد. توجه شود که اگر فرستنده این تأیید را در یافت کرد نمی‌تواند مطمئن باشد که پیام موردنظرش توسط گیرنده مربوط دریافت شده

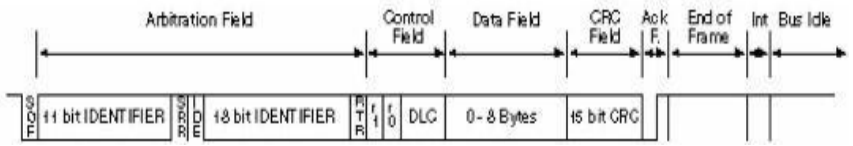
است. بلکه این تأییدیه تنها نشان می‌دهد که پیام به‌طور درستی بر روی گذرگاه CAN منتقل شده است.

- بیت‌های انتهای فریم Eof، ۷ بیت: در این قسمت از روش کدینگ شکستی (violated code)استفاده می‌شود. بدین صورت که در شرایط عادی بعد از ۵ بیت معادل یکدیگر بیت بعدی به‌طور مخالف به جریان اطلاعات اضافه می‌شود. این بیت‌ها هنگامی‌که Eof فعال است به همین ترتیب ادامه می‌یابد این بیت‌ها انتهای فریم CAN را مشخص می‌کنند.

- بیت‌های فاصله داخل فریم (IFS)، ۷ بیت: این بیت‌ها زمانی را برای کنترل کننده CAN جهت انتقال یک فریم اطلاعات سالم به ناحیه‌ای که اطلاعات در آنجا ذخیره می‌شوند تا توسط دستگاه مربوطه مورد استفاده قرار گیرند را به وجود می‌آورد.

- بیت بیکار (IOLE): در این هنگام گذرگاه مورد استفاده هیچ دستگاهی نبوده و هر نقطه شبکه می‌تواند ارتباط مورد نظر خود را شروع کند.

در حالت توسعه یافته ارسال اطلاعات نیز شامل همان هفت فیلد اصلی (آغاز، داوری، کنترل، داده، کد چرخشی، تصدیق و پایان) است، با این تفاوت که کد شناسایی در فیلد داوری دارای طول‌های متفاوتی هستند. در فرمت طولانی کد شناسایی دارای ۲۹ بیت است که ۱۱ بیت پایه و ۱۸ بیت افزوده شده می‌باشد، بنابراین طول فیلد داوری در این فرمت به ۳۲ بیت می‌رسد.



شکل ۱-۶-۸ فریم CAN توسعه یافته

- بیت SSR، ۱ بیت: این بیت جایگزین بیت RTR شده و معنی جدید دیگری ندارد.

- بیت IDE، ۱ بیت: این بیت در سیستم CAN توسعه یافته همواره یک است که مشخص می‌کند بیت‌های تعیین هویت بیشتری نیز به دنبال آن خواهد آمد.

- فریم کنترل، ۶ بیت: در بیت اول این فریم (r۱ و r۰) برای استفاده‌های بعدی رزرو شده، بقیه این بیت‌ها به همراه قسمت‌های دیگر پیام معنی مشابه پیام‌های CAN استاندارد دارد.

طول مجاز اطلاعات در این روش ۰ تا ۸ بایت است بنابراین اطلاعات بیش‌تر از ۸ بایت بنا به مشخصات تعریف شده در سیستم CAN مجاز نیست. هم‌چنین امکان بکارگیری پیام‌های استاندارد و توسعه یافته در یک شبکه به‌طور همزمان وجود دارد. این امکان با به کارگیری کنترل کننده‌های CAN با مشخصات B۰.۲CAN امکان‌پذیر است. در این صورت که این کنترلرها با ارزیابی بیت IOE پیام‌های نوع توسعه یافته از استاندارد را تشخیص می‌دهند. این نوع کنترل کننده‌ها از دریافت پیام‌های توسعه یافته خودداری می‌نمایند. کنترل کننده که بدون این امکان باشد نسبت به پرچم‌های خطا از خود عکس‌العمل نشان می‌دهد.

تخصیص بأس

در توپولوژی بأس در هر لحظه فقط یکی از ایستگاه‌ها می‌تواند به بأس دسترسی داشته باشد. برای اختصاص بأس به هر کدام از ایستگاه‌ها به منظور اینکه آن ایستگاه اطلاعاتی را به شبکه ارسال کند، دو روش وجود دارد:

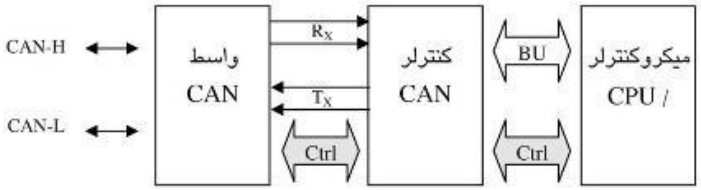
۱. تخصیص با برنامه زمانی ثابت: در فواصل زمانی از پیش تعیین شده بأس به ترتیب به هر کدام از ایستگاه‌ها اختصاص داده می‌شود.
۲. تخصیص بر حسب نیاز: هر ایستگاه که نیاز به ارسال اطلاعات دارد در خواستی صدور می‌کند و در صورت در اولویت قرار گرفتن درخواست، بأس به آن ایستگاه اختصاص می‌یابد.

درصورتی‌که چند ایستگاه با هم بخواهند پیغامی را ارسال کنند با توجه به اینکه ضرورت تبادل پیغام‌ها در شبکه متفاوت است، کمیت‌ها با تغییرات سریع (مانند جریان موتور) نسبت به کمیت‌های آهسته (مانند دمای موتور) دارای اولویت بیشتر ارسال می‌باشند. به این صورت که هر پیغام از نظر اهمیت و اولویت دارای یک شناسه منحصربه‌فرد است که در زمان رقابت ایستگاه‌ها شناسه با عدد باینری کوچک‌تر دارای اولویت بیشتر است.

روش برقراری ارتباط بین تجهیزات و سیستم CAN

روش اول: استفاده از تراشه‌های CAN

از مدارها و تراشه‌های مخصوص این‌گونه ارتباط که به صورت مجزا ساخته شده‌اند و ارتباط از طریق اتصال این قطعات به میکروکنترلر و برنامه‌نویسی تراشه‌ها صورت می‌پذیرد.



شکل ۱-۶-۹ روش Stand-alone

امروزه شرکت‌های زیادی اقدام به تولید چیپ‌های CAN کرده‌اند. برخی از آنها بدین شرح است:

Bosch {IP Modules; CAN Core, C_CAN, D_Can, TTCAN}

Cast {CAN Core, Bus Controller ICs}

freescale {۳۳۳۸۹/۳۳۳۸۸ low speed fault tolerant CANBus transceiver}

Infineon {۸۲C۹۰۰ Stand-alone TwinCan Controller-TLE۶۲۵۰ CAN Transceiver IC Manufacturer}

Inicore Inc. {CAN IP Core IC Manufacturer}

Intel, Intel App Notes {CanBus Interface ۸۲۵۲۷ IC}

Linear Technology {CAN Transceiver IC Manufacturer}

Maxim Integrated Products {DS۸۰C۳۹۰ Dual CAN High-Speed Microprocessor, bus controller IC}

Melexis {CAN Bus Transceiver IC Manufacturer}

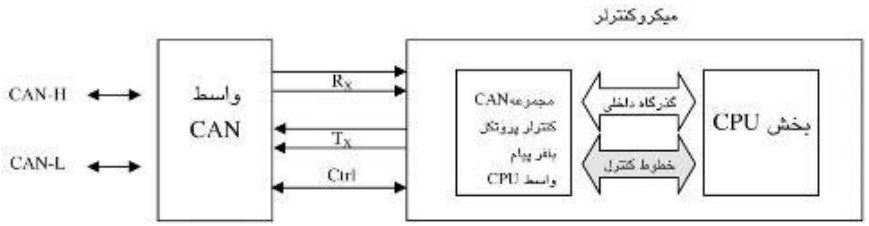
Microchip {MCP۲۵۱۰ Stand-alone CanBus Controller IC}

National Semiconductor {uP with CANBus Interface}

NXP {۸/۱۶-bit CAN Bus ۲.۰ Controllers/Transceiver}

OKI {CAN Controller}

روش دوم: استفاده از میکروکنترلرهایی است که قابلیت‌های فوق، درون آنها تعبیه شده است.



شکل ۱-۶-۱۰ روش On-Chip CAN

شرکت‌های Motorola و Atmel پیشنهاد دادند که میکرو پردازنده‌ها با CAN ترکیب شوند. امروزه شرکت‌های زیادی اقدام به تولید چیپ‌های CAN کرده‌اند. برخی از آنها بدین شرح است:

Analog Devices, Inc. {Mixed-Signal-DSPs (ADSP-۲۱۹۹۲) with ۱۶۰MIPS and On-Chip CAN V۲.۰b}

Atmel Corp. {۸-bit RISC transceivers and microcontrollers. CAN bus standard (۲.۰A & ۲.۰B) with ۸۰C۵۱ core and AVR core}

Xilinx {CAN IP Core, Spartan, Virtex}

TI {TMS۳۲۰F۲۴۱ with CANbus Interface-۳.۳v Line Transceiver ICs}

از جمله میکروکنترلرهایی که از این پروتکل صنعتی پشتیبانی می‌کنند می‌توان به آی سی‌های زیر اشاره کرد:

- میکروکنترلرهای ۸۰۵۱:

۰۳cc۵۱c۸۹, AT ۰۲cc۵۱c۸۹, AT۰۱cc۵۱c۸۹AT

- میکروکنترلرهای AVR:

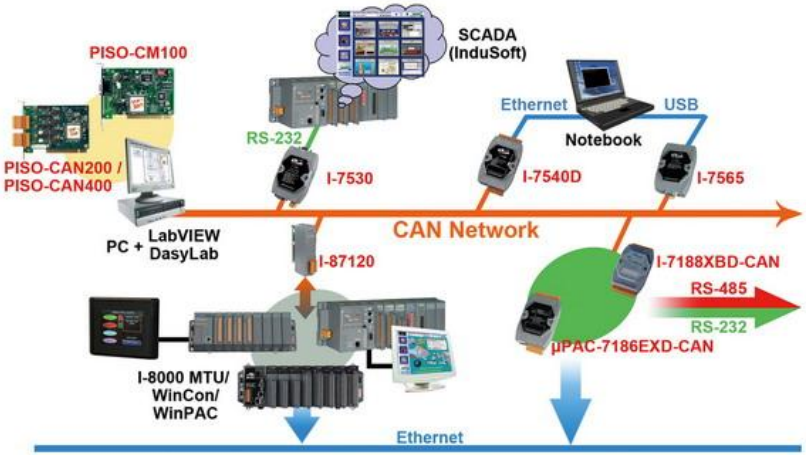
۳۲CAN۹۰, AT۶۴CAN۹۰, AT۱۲۸CAN۹۰AT

- میکروکنترلرهای PIC:

۴۵۸F۱۸PIC

روش سوم: کارت‌ها و مبدل‌ها CAN برای رایانه‌های صنعتی

ارتباط شبکه CAN با دیگر پروتکل‌های ارتباطی مانند ۲۳۲RS , USB۴۸۵RS, و شبکه‌ی Ethernet امروزه به صورت گسترده در شبکه‌های قطارهای شهری و شبکه‌های صنعتی مشاهده می‌شود.



شکل ۱-۶-۱۱ ارتباط شبکه CAN با دیگر پروتکل‌های ارتباطی به وسیله مبدل‌ها

پروتکل CANopen

استانداردهای مختلف در دنیا در سیستمهای کنترل و مونیتورینگ مورد توجه قرار گرفته است و سازندگان معتبر ریلی دنیا و تأمین کنندگان تجهیزات از این استانداردها پیروی می‌کنند.

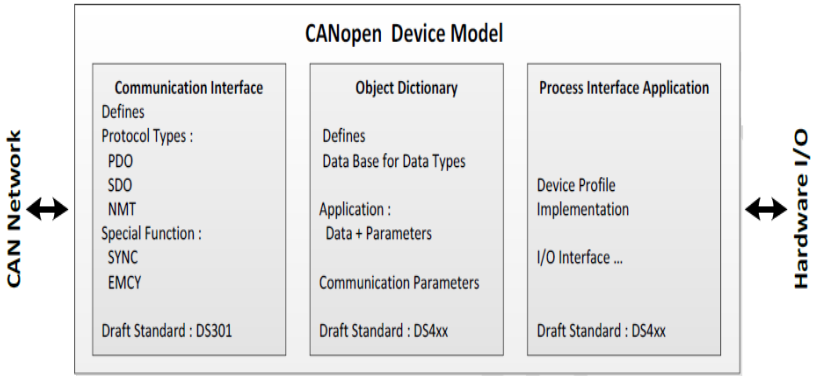
معروف‌ترین این استانداردها، استاندارد TCN یا Train Communication Network می‌باشد که به ۶۱۳۷۵IEC معروف است. در پایین‌ترین سطح ارتباطی پروتکلی موسوم به MVB برقرار می‌شود و پروتکل‌های دیگری همانند Can Open در این سطح قابل استفاده است. برخی تولید کنندگان پورت‌های CAN open را نیز در محصولات خود دیده‌اند: Leutze و MEN و amit و selectron ...

به عنوان نمونه قطار شهری تهران براساس پروتکل MVB و قطار شهری مشهد براساس Can Open و هر دو توسط شرکت CNR پیاده سازی شده است.

استاندارد TCN براساس MVBصرفاً جهت استفاده قطارها کاربرد دارد و پروتکل متن بازی نیست ولی استاندارد Can Open استانداردهای عمومی و باز می‌باشد و با توجه به بازار مصرف محدودتر پروتکل TCN شرکت‌های سازنده تراشه‌ها به فکر طراحی تراشه خاص پروتکل TCN نیفتاده‌اند؛ اما در مقابل با

توجه به عمومی بودن پروتکل Can Open شرکت‌های زیادی تراشه‌های خاص را ساخته و در اختیار عموم قرار داده‌اند. در TCN با تراشه MVBمحدود به سخت‌افزار چند سازنده محدود در سطح جهان خواهیم شد که در وضع کنونی مناسب نیست. نظر به مطالب بیان شده Can Open بهترین گزینه برای توسعه برای کشور است و بسیاری از مهندسین در ایران با این شبکه در کارهای صنعتی آشنا هستند و تولیدکنندگان زیادی در جهان تراشه‌های مربوط به این پروتکل را می‌سازند.

پروتکل CANopen از سه بخش لایه ارتباطی، دیکشنری اشیاء و واسط سخت‌افزاری تشکیل شده است و در صنعت توسعه پیدا کرده است.



استاندارد CANopen، ID۱۱ بیتی فریم CAN را به یک کد تابع ۴ بیتی و یک ID گره CANopen ۷ بیتی تقسیم می‌کند که باعث می‌شود تعداد دستگاه‌های یک شبکه به ۱۲۷ عدد محدود شود. یک توسعه از استاندارد CAN bus (CAN B) ID۰.۲ ی فریم‌های گسترش یافته را از ۲۹ بیت مجاز می‌شمارد ولی در عمل شبکه‌های CANopen به‌قدری گسترده هستند که نیاز

به محدوده ID گسترش یافته به‌ندرت مشاهده می‌شود. در CANopen، ID ۱۱ بیتی از فریمِ CAN، به عنوان شناسه شیء ارتباطی یا COB-ID شناخته می‌شود.

در CAN bus، به فریمی که کوچک‌ترین ID را دارد اجازه می‌دهد اول‌ازهمه و بدون تأخیر منتقل شود. استفاده از تعداد کم کد برای توابع زمان بحرانی کمترین تأخیر ممکن را تضمین می‌کند، چون ۴ بیت اول ID فریم در فریم‌های CANopen برای کد تابع رزرو شده‌اند.

محتویات یک فریم CANopen استاندارد:

	Function code	**Node ID**	**RTR**	**Data length**	**Data**
Length	bits۴	bits۷	bit۱	bits۴	bytes۸-۰

استاندارد، COB-ID های خاصی را برای مدیریت شبکه و انتقالات SDO رزرو می‌کند. برخی کدهای تابع و COB-IDها باید بر اساس عملکرد استاندارد بعد از مقداردهی اولیه دستگاه، ترسیم شوند، ولی بعداً می‌توانند برای کاربردهای دیگر پیکربندی شوند.

بسته‌های متشکل از یک ID ۱۱ بیتی، بیت درخواست انتقال از راه دور (RTR) و ۰ تا ۸ بایت داده را انتقال دهد

COB-ID

هر پیام در این پروتکل دارای یک شناسه منحصربه‌فرد است که نشان‌دهنده نود و نوع پیام است.

CANopen message types

Message Type	Description	COB-ID
NMT	Network Management (broadcast)	0h
NMT Error Control	Network management error control	701h – 77Fh
BOOT-UP	Boot-Up message	701h – 77Fh
SYNC	Synchronization message (broadcast)	80h
EMERGENCY	Emergency messages	81h - FFh
TIME STAMP	Time stamp (broadcast)	100h
PDO	Process Data Objects	181h - 57Fh
SDO	Service Data Objects	581h – 67Fh

COB-ID	Object	Mandatory/optional
0x000	NMT	Mandatory
0x080	SYNC	Optional
0x081 – 0x0FF	EMERGENCY	Optional
0x100	TIME STAMP	Optional
0x181 – 0x1FF	PDO1 (transmit)	Optional
0x201 – 0x27F	PDO1 (receive)	Optional
0x281 – 0x2FF	PDO2 (transmit)	Optional
0x301 – 0x37F	PDO2 (receive)	Optional
0x381 – 0x3FF	PDO3 (transmit)	Optional
0x401 – 0x47F	PDO3 (receive)	Optional
0x481 – 0x4FF	PDO4 (transmit)	Optional
0x501 – 0x57F	PDO4 (receive)	Optional
0x581 – 0x5FF	SDO (transmit / server)	Mandatory
0x601 – 0x67F	SDO (receive / client)	Mandatory
0x701 – 0x77F	NMT Error Control	Mandatory

مدل‌های ارتباطی

انواع مدل‌های ارتباطی مختلف در پیام‌رسانی بین گره‌های CANopen استفاده می‌شوند.

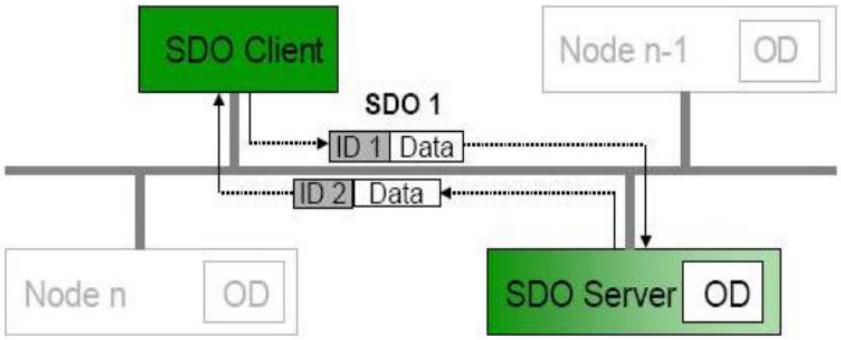
۱- ارتباط **slave/master**، یک گره CANopen به عنوان **master**طراحی شده است که اطلاعات را به **slave** می‌فرستد یا از آن‌ها درخواست می‌کند. پروتکل NMT یک مثال از مدل ارتباطی مَستر/اسلیو است.

ارتباط **server /Clint** در پروتکل SDO پیاده سازی شده که در آن کلاینت SDO داده (اندیس و زیر اندیس دیکشنری شیء) را به یک سرور SDO می‌فرستد که با یک یا تعداد بیشتری بسته SDO شامل داده درخواستی (محتویات دیکشنری شیء در اندیس داده شده)، پاسخ می‌دهد.

مدل **consumer /producer** در پروتکل Heartbeat و PDO و پروتکل‌های امنیتی گره‌ها استفاده می‌شود. در حالت push از مدل تولید کننده/مصرف کننده، تولید کننده داده را بدون درخواست خاصی به مصرف کننده می‌فرستد، در حالی که در مدل pull، مصرف کننده مجبور به درخواست داده از تولید کننده است.

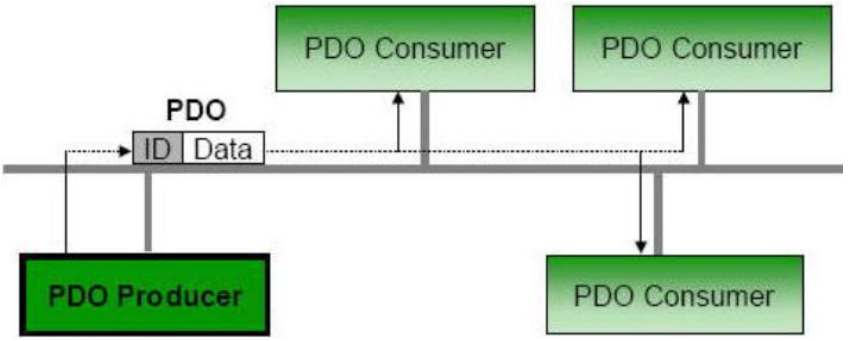
پیامSDO :

این پیام برای ارسال تنظیمات و یا خواندن و نوشتن از دیکشنری اشیا کاربرد دارد در زیر چگونگی ارسال و دریافت را نشان داده‌ایم: رجیسترها در پیام SDO توسط Index شانزده بیتی و Subindex هشت بیتی مشخص می‌شوند.



پیام PDO:

برای تبادل اطلاعات در لحظه استفاده می‌شود و ۸ بایت را می‌تواند جابجا شود و پیام فقط توسط یک عضو شبکه صادر ولی چندین عضو می‌توانند آن را دریافت دارند. (تولید کننده / مشتری)



این پیام به دو دسته RPDO و TPDO تقسیم می‌شود که RPDOها برای دریافت اطلاعات شبکه و TPDOها برای ارسال اطلاعات به شبکه‌اند.

پروتکل SDO (Service Data Object)

پروتکل SDO برای تنظیم و خواندن دیکشنری شیء از یک دستگاه راه دور استفاده می‌شود. دستگاهی که دیکشنری شیء اش در دسترس است سرور SDO است و دستگاهی که به دستگاه راه دور دسترسی پیدا می‌کند کلاینت SDO است.

ارتباط همواره با کلاینت SDO آغاز می‌شود. در اصطلاحات CANopen، ارتباط از دید سرور SDO بیان می‌شود، برای همین خواندن از دیکشنری شیء منجر به آپلود می‌شود و نوشتن روی دیکشنری یک دانلود SDO محسوب می‌شود.

COB-ID های ۶۰۱x۰-F۶۷x۰ برای ارتباط کلاینت‌ها با سرور اند و COB-ID های ۵۸۱x۰-FF۵x۰ برای ارتباط سرور با کلاینت‌ها است.

Client: *COB-ID = ۰x۶۰۰ + [Node ID of server to reach]*

*+ [Node ID of client to reach]۵۸۰x۰*Server: *COB-ID =*

فرمت اول: برای شروع دانلود، کلاینت SDO داده زیر را در یک پیام CAN با "دریافتCOB-ID " از کانال SDO، می‌فرستد. **(initiate upload)**

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs	reserved(=0)	n	e	s	index	subindex	data

CS: تعیین کننده دستورات SDO است، صفر: برای دانلود بخش SDO یک: برای شروع دانلود،

دو: برای درخواست آپلود، سه: برای آپلود بخش SDO چهار: برای لغو یک انتقال SDO

n تعداد بایت‌های در بخش داده پیام است که حاوی داده نیستند، فقط زمانی که هر دو بیت e و s یک باشند معتبر است.

e: یک: یک انتقال تسریع شده را نشان می‌دهد، یعنی تمام داده‌های مبادله شده در پیام ۴ بایتی موجود هستند.

صفر: پیام به صورت یک انتقال تقسیم شده خواهد بود که داده در یک پیام نگنجیده و از چندین پیام استفاده شده است.

S : یک: اگر بیت e هم یک باشد مقدار n معتبر خواهد شد و اندازه داده مشخص می‌شود.

یک: اگر بیت e صفر باشد مقدار Dataاندازه داده را مشخص می‌کند.

Index اندیس دیکشنری شیء OD داده قابل دسترسی است.

Subindex زیر اندیس متغیر دیکشنری شیء OD است.

Data : اگر e یک باشد: داده آپلود شونده

اگر e صفر باشد و s یک باشد: اندازه داده آپلود شونده را نشان می‌دهد.

فرمت دوم: دیتا به فرمت زیر در sdo ارسال می‌شود

3 bits	1 bit	3 bits	1 bit	7 bytes
cs	t	n	l	data

CS:

تعیین کننده دستورات SDO است، صفر: برای دانلود بخش SDO در کلاینت یک: برای در خواست نوشتن در سرور یا دانلود، دو: برای درخواست آپلود، سه: اعلام آمادگی برای آپلود بخش SDO چهار: برای لغو یک انتقال SDO

n تعداد بایت‌های در بخش داده پیام است که حاوی داده نیستند، فقط زمانی که هر دو بیت e و s یک باشند معتبر است.

t: این بیت باید در هر فریم ارسال toggle شود.

l: این بیت نشان دهنده آخرین فریم است و در آخرین فریم یک می‌شود.

Data: اطلاعات ارسالی

خواندن مقدار منقطع CANopen_SDOC_Exp_Read

مثال: کدهای لازم برای خواندن از دیکشنری شیء به ایندکس h۲۰۰۰ و زیر ایندکس ۰۰ را بنویسید فرض کنید می‌توانید از نوع انتقال تسریع شده استفاده کنید:

مرحله اول: درخواست آپلود که client می‌فرستد به فرم زیر است و در آن مقدار دیتا چون دستور قرائت است، صفر است.

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs = 2	reserved(=0)	n = 0	e = 0	s = 0	Index = 0x2000	Subindex = 0x00	Data = 0x00

مرحله دوم: سرور باید به آن جواب بدهد که نشان دهد این ایندکس وجود دارد و مقدار دیتای موجود در زیر ایندکس ۰۰x۰ برابر XAA۰ است را در مقدار DATA به صورت یک بایتی ارسال می‌کند. چون سه بایت دیتا آزاد می‌ماند باید n مقدار ۳ را بگیرد و e و s هر دو یک شوند.

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs = 2	reserved(=0)	n = 3	e = 1	s = 1	Index = 0x2000	Subindex = 0x00	Data = 0xAA

Type	ID	DLC	Data
Data	601	8	40 00 20 00 00 00 00 00
Data	581	8	4F 00 20 00 AA 00 00 00

```
CANopen_SDOC_Exp_Read(CAN_SLAVE۱_NODE, ۰x۲۰۰۰, ۰x۰۰, (uint۸_t*)buffer, NULL);

while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));
```

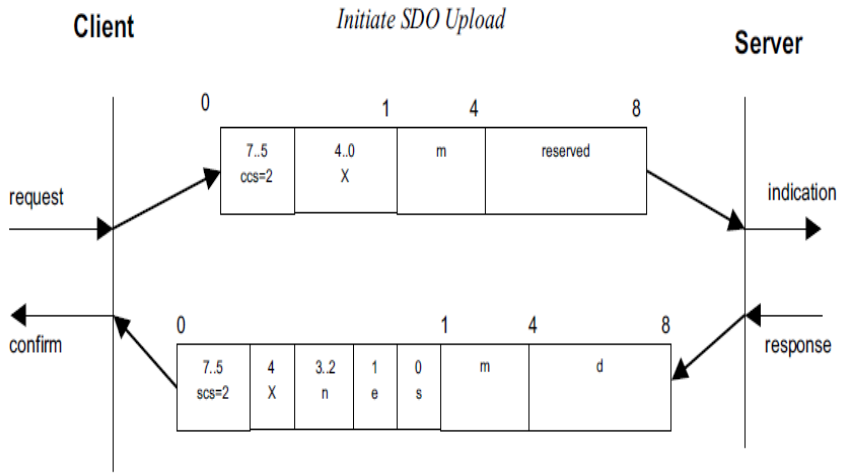


Figure 20: Initiate SDO Upload Protocol

Refrence: CiA-۳۰۱ page ۴۶

نوشتن منقطع CANopen_SDOC_Exp_Write

مثال کدهای لازم برای نوشتن در دیکشنری شیء به ایندکس h۲۰۰۰ و زیر ایندکس ۰۰ را بنویسید فرض کنید می‌توانید از نوع انتقال تسریع شده استفاده کنید:

مرحله اول: درخواست دانلود که client می‌فرستد به فرم زیر است و در آن مقدار دیتا و آدرس آن وجود دارد. چون سه بایت دیتا آزاد می‌ماند باید n مقدار ۳ را بگیرد و e و s هر دو یک شوند.

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs = 1	reserved(=0)	n = 3	e = 1	s = 1	Index = 0x2000	Subindex = 0x00	Data = 0x55

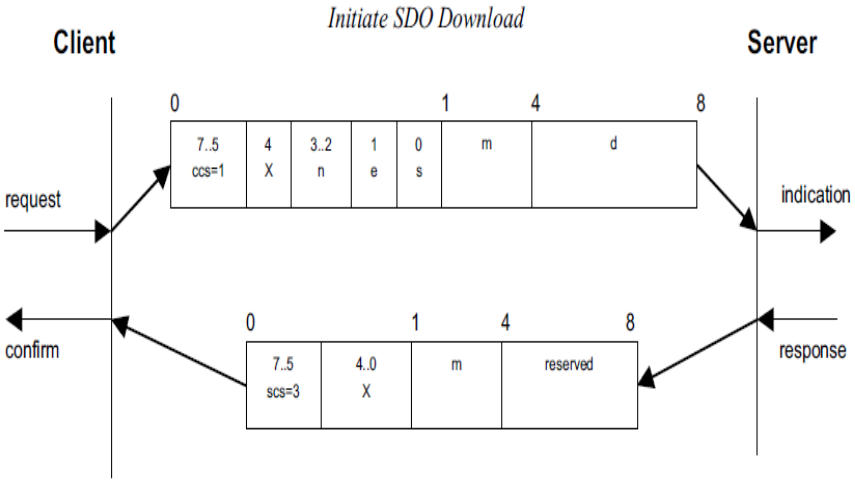
مرحله دوم: سرور باید به آن جواب بدهد و با =cs۳ تأیید کند که نشان دهد این ایندکس وجود دارد.

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs = 3	reserved(=0)	n = 0	e = 0	s = 0	Index = 0x2000	Subindex = 0x00	Data = 0x00

Type	ID	DLC	Data
Data	601	8	2F 00 20 00 55 00 00 00
Data	581	8	60 00 20 00 00 00 00 00

CANopen_SDOC_Exp_Write(CAN_SLAVE۲_NODE, ۰x۲۰۰۰, ۰x۰۰, (uint۸_t*)buffer, ۱);

while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));



Initiate SDO Download Protocol

Refrence: CiA-۳۰۱ page ۴۴

خواندن رشته بلند از دیکشنری CANopen_SDOC_Seg_Read

مثال: کدهای لازم برای خواندن از دیکشنری شیء به ایندکس h۲۲۰۰ و زیر ایندکس ۰۰ را بنویسید فرض کنید به علت حجم اطلاعات نمی‌توانید از نوع انتقال تسریع شده استفاده کنید و باید از انتقال تقسیم شده استفاده کنید:

مرحله اول: درخواست آپلود که client می‌فرستد به فرم زیر است و در آن مقدار دیتا چون دستور قرائت است، صفر است.

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs = 2	reserved(=0)	n = 0	e = 0	s = 0	Index = 0x2200	Subindex = 0x00	Data = 0x00

مرحله دوم: سرور باید به آن جواب بدهد که نشان دهد این ایندکس وجود دارد و دیتایی در این زمان انتقال نمی‌یابد پس دیتا و n و e وs همه صفرند.

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs = 2	reserved(=0)	n = 0	e = 0	s = 0	Index = 0x2200	Subindex = 0x00	Data = 0x00

مرحله سوم: کلاینت آمادگی خود را برای دریافت اعلام می‌دارد. اولین تاگل با صفر شروع می‌شود. بیت‌های n و I و t و دیتا باید صفر باشند.

3 bits	1 bit	3 bits	1 bit	7 bytes
cs = 3	t = 0	n = 0	I = 0	data = 0x00

مرحله چهارم: سرور اولین بسته ۷ بایتی از آدرس مورد نظر می‌فرستد

3 bits	1 bit	3 bits	1 bit	7 bytes
cs = 0	t = 0	n = 0	l = 0	data = ??

مرحله پنجم: دوباره مرحله ۳ و ۴ تکرار می‌شود یعنی کلاینت ack می‌کند و سرور می‌فرستد تا فریم دیگر ۷ بیت نشود.

مرحله ششم: در اینجا L برای نشان دادن آخرین بیت یک می‌شود و n تعداد بایت‌های خالی را نشان می‌دهد.

3 bits	1 bit	3 bits	1 bit	7 bytes
cs = 0	t = ??	n = ??	l = 1	data = ??

Type	ID	DLC	Data
Data	601	8	40 00 22 00 00 00 00 00
Data	581	8	40 00 22 00 00 00 00 00
Data	601	8	60 00 00 00 00 00 00 00
Data	581	8	00 42 6F 6F 74 2D 75 70
Data	601	8	70 00 00 00 00 00 00 00
Data	581	8	10 20 76 61 6C 75 65 20
Data	601	8	60 00 00 00 00 00 00 00
Data	581	8	00 6F 66 20 53 44 4F 20
Data	601	8	70 00 00 00 00 00 00 00
Data	581	8	10 32 32 30 30 68 00 00
Data	601	8	60 00 00 00 00 00 00 00
Data	581	8	00 00 00 00 00 00 00 00
Data	601	8	70 00 00 00 00 00 00 00
Data	581	8	10 00 00 00 00 00 00 00
Data	601	8	60 00 00 00 00 00 00 00
Data	581	8	00 00 00 00 00 00 00 00
Data	601	8	70 00 00 00 00 00 00 00
Data	581	8	10 00 00 00 00 00 00 00
Data	601	8	60 00 00 00 00 00 00 00
Data	581	8	09 00 00 00 00 00 00 00

CANopen_SDOC_Seg_Read (node_id, index, subindex, buff, buffSize);

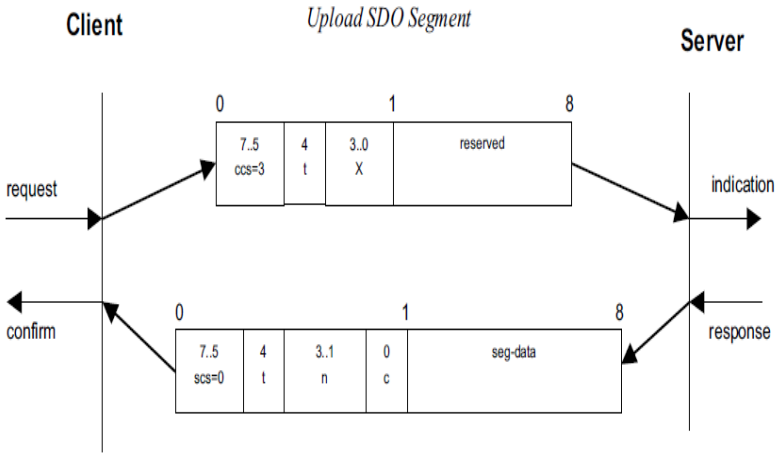


Figure 21: Upload SDO Segment Protocol

Refrence: CiA-۳۰۱ page ۴۷

نوشتن رشته بلند در دیکشنری CANopen_SDOC_Seg_Write

مثال: کدهای لازم برای نوشتن در دیکشنری شیء به ایندکس h۲۲۰۰ و زیر ایندکس ۰۰ را بنویسید فرض کنید به علت حجم اطلاعات نمی‌توانید از نوع انتقال تسریع شده استفاده کنید و باید از انتقال تقسیم شده استفاده کنید:

مرحله اول: درخواست دانلود که client می‌فرستد به فرم زیر است و در آن مقدار طول دیتای که قصد نوشتن داریم را در بخش data قرار گرفته می‌دهیم.

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs = 1	reserved(=0)	n = 0	e = 0	s = 1	Index = 0x2200	Subindex = 0x00	Data = length

مرحله دوم: سرور باید به آن جواب بدهد که نشان دهد این ایندکس وجود دارد

3 bits	1 bit	2 bits	1 bit	1 bit	2 bytes	1 byte	4 bytes
cs = 3	reserved(=0)	n = 0	e = 0	s = 0	Index = 0x2200	Subindex = 0x00	Data = 0x00

مرحله سوم: کلاینت شروع به ارسال به ارسال دیتا در قالب **بسته‌های ۷ بایتی** می‌کند

3 bits	1 bit	3 bits	1 bit	7 bytes
cs = 0	t = 0	n = 0	I = 0	data = ??

مرحله چهارم: سرور دریافت اولین بسته را تأیید ack می‌کند مقدار دیتا در این مرحله اهمیتی ندارد و بیت‌های n و I و t باید صفر باشند.

3 bits	1 bit	3 bits	1 bit	7 bytes
cs = 1	t = 0	n = 0	I = 0	data = ??

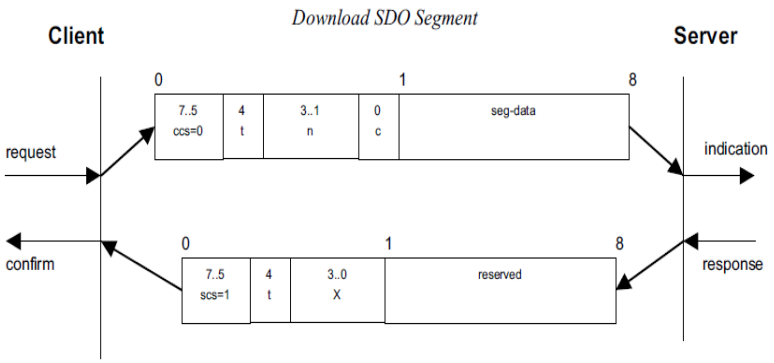
مرحله پنجم: دوباره مرحله ۳ و ۴ تکرار می‌شود یعنی کلاینت می‌فرستد و سرورack می‌کند تا فریم دیگر ۷ بیت نشود.

مرحله ششم: در اینجا L برای نشان دادن آخرین بیت یک می‌شود و n تعداد بایت‌های خالی را نشان می‌دهد.

3 bits	1 bit	3 bits	1 bit	7 bytes
cs = 0	t = ??	n = ??	I = 1	data = ??

Type	ID	DLC	Data
Data	601	8	21 00 22 00 2C 00 00 00
Data	581	8	60 00 22 00 00 00 00 00
Data	601	8	00 54 68 69 73 20 69 73
Data	581	8	20 00 22 00 00 00 00 00
Data	601	8	10 20 61 20 6D 65 73 73
Data	581	8	30 00 22 00 00 00 00 00
Data	601	8	00 61 67 65 20 65 6E 74
Data	581	8	20 00 22 00 00 00 00 00
Data	601	8	10 65 72 65 64 20 66 72
Data	581	8	30 00 22 00 00 00 00 00
Data	601	8	00 6F 6D 20 74 68 65 20
Data	581	8	20 00 22 00 00 00 00 00
Data	601	8	10 74 65 72 6D 69 6E 61
Data	581	8	30 00 22 00 00 00 00 00
Data	601	8	0B 6C 00 00 00 00 00 00
Data	581	8	20 00 22 00 00 00 00 00

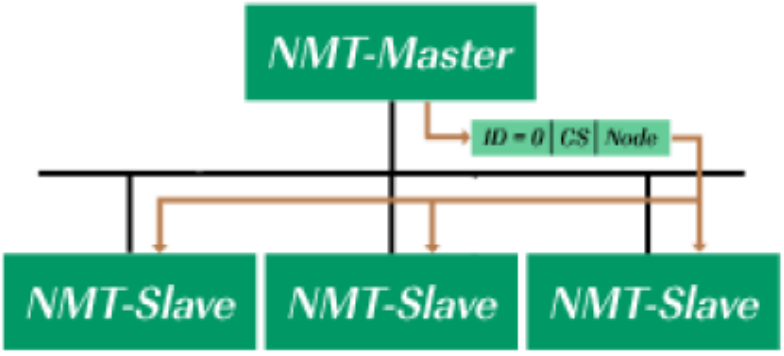
CANopen_SDOC_Seg_Write(node_id, index, subindex,buff, length);



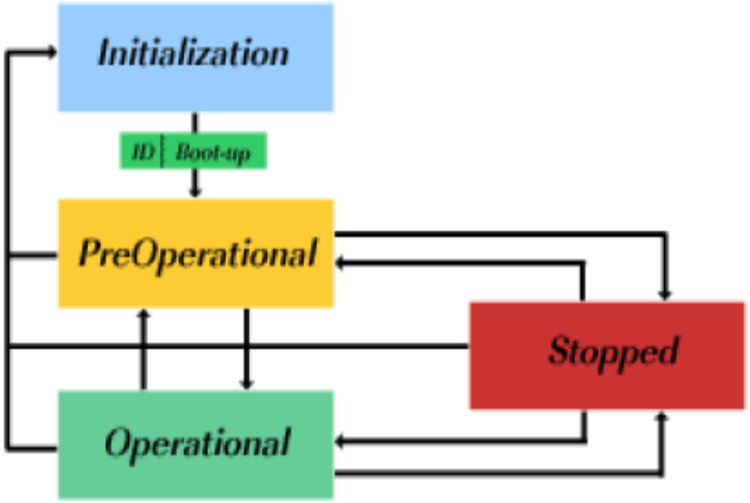
Download SDO Segment Protocol

پروتکل‌های مدیریت شبکه(NMT)

برای صدور دستورات تغییر در حالت دستگاه و یا نشان دادن حالت دستگاه استفاده می‌شود.



تمام اسلیوها دارای حالت دستگاه چهارگانه زیرند (بالا آمدن دستگاه‌ها و قبل بهره‌برداری، بهره‌برداری و توقف)



۱- حالت بالا آمدن:

بعد از روشن شدن تمام نودهای اسلیو باید پیام bootup را صادر و وارد مرحله قبل بهره‌برداری یا PreOperational شوند. پیام bootup به صورت زیر است و با دیتای یک بایتی برابر صفر است:

COB-ID ۷۰۰x۰ [node-id of NMT slave] +

۲- حالت قبل بهره‌برداری یا PreOperational:

انجام ارتباط SDO در این مرحله مجاز است ولی ارتباط PDO مجاز نیست.

۳- حالت بهره‌برداری: انجام ارتباط SDO و PDO در این مرحله مجاز است

۴- حالت توقف: انجام ارتباط SDO و PDO در این مرحله مجاز نیست.

دستورNMT مستر:

در ابتدا مستر وضعیت تمام اسلیوها را با ارسال دستور تعیین می‌دارد مقدار دیتا ارسالی در این حالت دو بایت است که به بایت اول Command Specified (CS) می‌گویند و بایت دوم نیزnode-ID است. این دیتا به همراه COB-ID ارسال می‌شود. آدرس نود صفر نیز معنا دارد به این معنی که هر گره در شبکه این پیام را پردازش خواهد کرد.

فهرست دستورات نیز به صورت زیر است:

۱: Start remote node.

۲: Stop remote node.

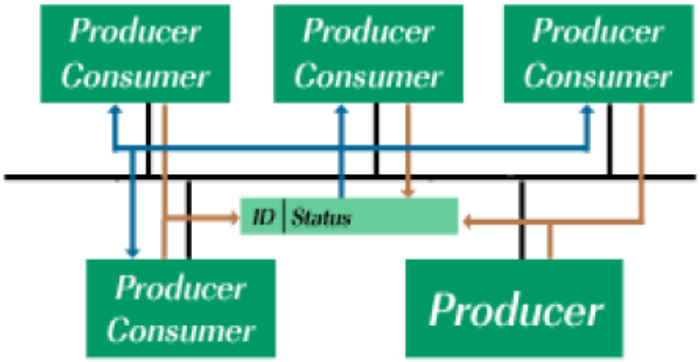
۱۲۸: Enter Pre-Operational.

۱۲۹:Reset node.

۱۳۰:Reset Communication.

پروتکل Heartbeat

برای مانیتور کردن گره‌ها در شبکه و تأیید اینکه هنوز زنده (alive) هستند، به کار می‌رود. تولید کننده هارت بیت پیام هارت بیت را به‌طور چرخه‌ای با توالی تعریف شده در هدف زمانی تولید کننده منتقل می‌کند. یک یا چند مصرف کننده هارت بیت نشان را دریافت خواهند کرد. رابطه میان تولید کننده و مصرف کننده از طریق ورودی‌های دیکشنری هدف قابل پیکره‌بندی است. مصرف کننده هارت بیت دریافت هارت بیت را در زمان مصرف کننده محافظت می‌کند اگر هارت بیت در این زمان دریافت نشود رویداد هارت بیت ایجاد خواهد شد



یک تولید کننده heartbeat که معمولاً یک دستگاه اِسلیو است یک پیام را به صورت دوره‌ای با کد تابع باینری ۱۱۱۰ و ID گره = COB-ID) ۷۰۰x۰ +

(node IDمی‌فرستد. بخش داده فریم شامل یک بایت نشان دهنده وضعیت گره است. مصرف کننده heartbeat این پیام‌ها را می‌خواند. اگر پیام‌ها در یک محدوده زمانی خاص (در دیکشنری شیء دستگاه تعریف شده است) موفق به رسیدن نشوند، مصرف کننده نیز می‌تواند اقدام کند، برای مثال دستگاه را ریست کند یا یک خطا را نشان دهد. فرمت فریم به صورت زیر است:

COB-ID	۰Data Byte
+ node ID۷۰۰x۰	State

مقدار وضعیت به صورت زیر است:

۰: Boot-up

۴: Stopped.

۵: Operational.

۱۲۷: Pre-Operational.

پروتکل heartbeat توسط ۲ ایندکس یکی برای تولید کننده به ایندکس h۱۰۱۷ و یکی برای مصرف کننده به ایندکس h۱۰۱۶ در دیکشنری اشیاء کنترل می‌شود.

ایندکس h۱۰۱۶ یک آرایه است که در زیر ایندکس ۰۰x۰ آن یک بایت حاوی بیشترین مقدار نودهایی که می‌تواند مشاهده و نظارت شود می‌باشد. دیگر زیر ایندکس‌ها ۴ بایت به صورت زیر می‌باشد. شماره نودهای ذکر شده

نودهایی‌اند که باید نظارت شوند؛ و زمان یاد شده نیز مربوط فاصله زمانی بین دو مصرف کننده هارت بیت بر حسب میلی ثانیه است.

bits	31...24	32...16	15...0
Value	0	Node-ID	Heartbeat time
Encoding	-	Unsigned8	Unsigned16

ایندکس h۱۰۱۷ یک آرایه با یک زیر ایندکس به آدرس ۰۰x۰ است که حاوی دو بایت است که عددی از نوع Unsigned۱۶ و فاصله زمانی بین دو تولید کننده هارت بیت بر حسب میلی ثانیه است.

مثالی از هارت بیت

۱- تنظیمات اولیه اولین نود اسلیو: تنظیم زمان هارت بیت برای تولید کننده به صورت ۴۰۰۰ میلی ثانیه و برای مصرف کننده به صورت ۴۵۰۰ میلی ثانیه.

بدین صورت که در خط اول پیامی از نوع SDO کلاینت به ایندکس h۱۰۱۷ حاوی مقدار دو بایت ۰FA۰x۰ برابر ۴۰۰۰ میلی ثانیه تولید می‌شود.

در خط دوم سرور وجود ایندکس را با SDO تأیید می‌کند.

در خط سوم پیامی از نوع SDO کلاینت به ایندکس h۱۰۱۶ و در زیر ایندکس حاوی دو بایت ۱۱۹۴x۰ برابر ۴۵۰۰ میلی ثانیه تولید می‌شود. شماره نود برای مصرف کننده هم D۷x۰ تنظیم شده است.

در خط آخر سرور وجود ایندکس و زیر ایندکس را با SDO تأیید می‌کند.

Time	Type	ID	DLC	Data
2.2381	Data	601	8	2B 17 10 00 A0 0F 00 00
2.2391	Data	581	8	60 17 10 00 00 00 00 00
2.2400	Data	601	8	23 16 10 01 94 11 7D 00
2.2410	Data	581	8	60 16 10 01 00 00 00 00

۲- هم مستر و هم اسلیوها هارت بیت تولید می‌کنند وضعیت آنها در حالت قبل بهره‌برداری است یا ۱۲۷ یا (f۷x۰) را تولید می‌کنند.COB-ID به صورت ۷۰۰x۰ + node ID است چون شماره نود برای مصرف کننده D۷x۰ و برای تولید کننده ۰۱x۰ تنظیم شده است d۷۷x۰ و ۷۰۱x۰ تولید می‌شود مقدار دیتا هم همان ۱۲۷ یا F۷x۰ است.

Time	Type	ID	DLC	Data
2.4830	Data	77D	1	7F
6.2380	Data	701	1	7F
6.4829	Data	77D	1	7F
10.2381	Data	701	1	7F
10.4828	Data	77D	1	7F
14.2380	Data	701	1	7F
14.4828	Data	77D	1	7F
18.2381	Data	701	1	7F
18.4829	Data	77D	1	7F
22.2381	Data	701	1	7F
22.4829	Data	77D	1	7F

۳- از دست رفتن هارت بیت اسلیو را شبیه سازی می‌کنیم. بعد از ۴۵۰۰ میلی ثانیه مستر می‌تواند کشف کند هارت بیت اسلیو از دست رفته است و NMT فرمان مدیریت شبکه مستر ۸۲x۰ یا ۱۳۰ یا همان فرمان ریست ارتباط است را برای نودهای اسلیو می‌فرستد. مقدار دیتا ارسالی در این حالت دو بایت است که به بایت اول ۸۲x۰ و بایت دوم نیز ۰۱x۰ یا node-ID است COB-ID . در NMT به صورت ۰ است.

Time	Type	ID	DLC	Data
26.7380	Data	000	2	82 01

۴- در پاسخ به فرمان NMT ۱۳۰ یا ریست ارتباط، اسلیو پارامترهای ارتباطی خود را ریست می‌کند و بعد از انجام تنظیمات به حالت پیش از بهره‌برداری وارد می‌شود و در حالت هارت بیت پیام boot-up را صادر می‌کند.COB-ID به صورت ۷۰۰x۰ + node ID است چون شماره نود برای تولید کننده ۰۱x۰ تنظیم شده است ۷۰۱x۰ تولید می‌شود مقدار دیتا هم همان ۰ یا boot-up است.

Time	Type	ID	DLC	Data
26.7384	Data	701	1	00

۵- نود مستر پیام بوت را می‌بیند و به یاد می‌آورد که هارت بیت گم شده‌ای رخ داده است پس در این حالت مستر، اسلیو را برای هارت بیت تولید کننده و مصرف کننده باز تنظیم می‌کند. این مرحله دقیقاً مثل گام اول است.

Time	Type	ID	DLC	Data
26.7500	Data	601	8	2B 17 10 00 A0 0F 00 00
26.7510	Data	581	8	60 17 10 00 00 00 00 00
26.7534	Data	601	8	23 16 10 01 94 11 7D 00
26.7544	Data	581	8	60 16 10 01 00 00 00 00

۶- هم مستر و هم اسلیو هارت بیت را دوباره تولید می‌کند. این مرحله هم دقیقاً مثل گام دوم است.

Time	Type	ID	DLC	Data
30.4929	Data	77D	1	7F
30.7502	Data	701	1	7F
34.4929	Data	77D	1	7F
34.7502	Data	701	1	7F

بررسی توابع اصلی CANopen برای تراشه CXX۱۱LPX

پوینتر به عنوان درایور در ROM

کد شامل یک پوینتر به عنوان درایور در ROM در تراشه CXX۱۱LPX است این تراشه کلاً ۱۶ کیلوبیت رام دارد که ۸ کیلوبیت آخر آن از آدرس ۸FF۱FFF۱x۰ به این درایور اختصاص یافته است این پوینتر نیاز به ۲۲ بایت یا ۱۷۶ بیت حافظه دارد.

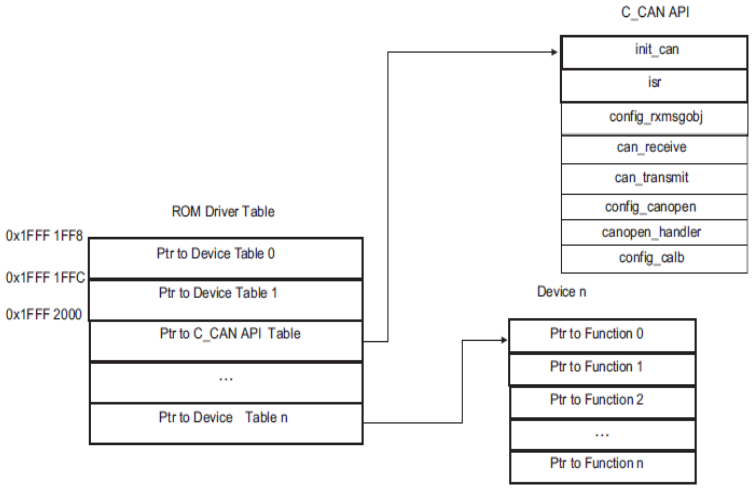
```
ROM **rom = (ROM **)۰x۱FFF۱FF۸;/* ROM entrypoint for on-chip CAN drivers */
/* Global variables */
volatile uint۳۲_t CANopen_Timeout;/* CANopen timeout timer for SDO client */
volatile uint۸_t* CANopen_SDOC_Exp_ValidBytes;/* Number of valid bytes for SDO client expedited read */
volatile uint۳۲_t CANopen_SDOC_InBuff;/* Number of bytes in SDO client buffer for segmented read/write */
volatile uint۸_t* CANopen_SDOC_Buff;     /* Buffer for SDO client segmented read/write */
volatile uint۳۲_t CANopen_SDOC_Seg_BuffSize;/* Size of SDO client buffer in number of bytes */
```

volatile uint۱۶_t CANopen_SDOC_Seg_ID; /* Target node ID for SDO client segmented read/write */

volatile uint۸_t CANopen_NMT_MyState; /* Present state of NMT state machine */

volatile uint۸_t CANopen_SDOC_State;/* Present state of SDO client state for controlling SDO client transfers */

volatile uint۳۲_t CANopen_Heartbeat_Producer_Counter;



توابع موجود در canopen_driver

در canopen_driver.h لیستی از توابع وجود دارد که وظایف خاصی دارند و به صورت زیر است:

۱. CANopenInit: انجام تنظیمات اولیه درایور Canopen که در ادامه این تابع توضیح بیشتری داده شده.

۲. _CANopen۱ms_tick: هر یک میلی ثانیه باید فرآخوانی شود و کارهای زیر ا را انجام می‌دهد:

- بحث timeout در SDO را بررسی و وضعیت CANopen_SDOC_Fail را تولید می‌کند

- تولید هارت بیت تولید کننده را بررسی می‌کند و به محض رسیدن به مقدار حداکثر تعیین شده تابع CANopen_Heartbeat_Send را که وظیفه تولید و ارسال پیام مدیریت شبکه در مورد رخداد هارت بیت است را صدا میزند.

- لیت مشاهده را بررسی و در عدم در یافت هارت بیت و به محض رسیدن به مقدار حداکثر تعیین شده تابع CANopen_Heartbeat_Consumer _Failed را برای تولید پیام مدیریت شبکه را صدا می‌زند.

- سپس برای باز تنظیم اسلیو مقدار زمان هارت بیت تولید کننده به آدرس ایندکس ۱۰۱۷x۰ و زیر اندکس ۰ از طریق SDO_EXP یا تسریع شده ارسال می‌شود.

- سپس برای باز تنظیم اسلیو در صورت لزوم مقدار هارت بیت مصرف کننده به آدرس ایندکس ۱۰۱۶x۰ و زیر اندکس ۰۱x۰ که برای از طریق SDO_EXP یا تسریع شده ارسال می‌شود.

۳. CANopen_Init_SDO: این تابع در تنظیمات اولیه درایور Canopen و در ریست شدن نود توسط فرمان مدیریت شبکه فرآخوانی می‌شود. این تابع به صورت زیر است که مقادیر هارت بیت‌ها را دوباره تنظیم و صفر می‌کند:

```
for(i=۰; i<WatchListLength; i++)
{
```

```
WatchList[i].value = ۰;

WatchList[i].counter = ۰;

}

CANopen_Heartbeat_Producer_Value =
HEARTBEAT_PRODUCER_TIME;

CANopen_Heartbeat_Producer_Counter = ۰;
```

۴. CANopen_NMT_Reset_Node_Received: این تابع در هنگام دریافت فرمان مدیریت شبکه دال بر "ریست کردن نود" فرآخوانی می‌شود.

۵. CANopen_NMT_Reset_Comm_Received: این تابع در هنگام دریافت فرمان مدیریت شبکه دال بر "ریست کردن ارتباط" فرآخوانی می‌شود.

۶. CANopen_Heartbeat_Consumer_Failed: این تابع وقتی که یک نود در لیست مشاهده هارت بیت را در زمان مشخص ارسال ندارد فرآخوانی می‌شود.

۷. CANopen_NMT_Consumer_Bootup_Received: وقتی یک نود در لیست مشاهده پیام Bootup را ارسال دارد این تابع فرآخوانی خواهد شد.

توابع درایور

لیست توابع APIها به کاررفته به صورت زیر است:

```
typedef      struct _CAND {

 void (*init_can)(uint۳۲_t * can_cfg, uint۸_t isr_ena);

 void (*isr)(void);

 void (*config_rxmsgobj)(CAN_MSG_OBJ * CANopen_Msg_Obj);
```

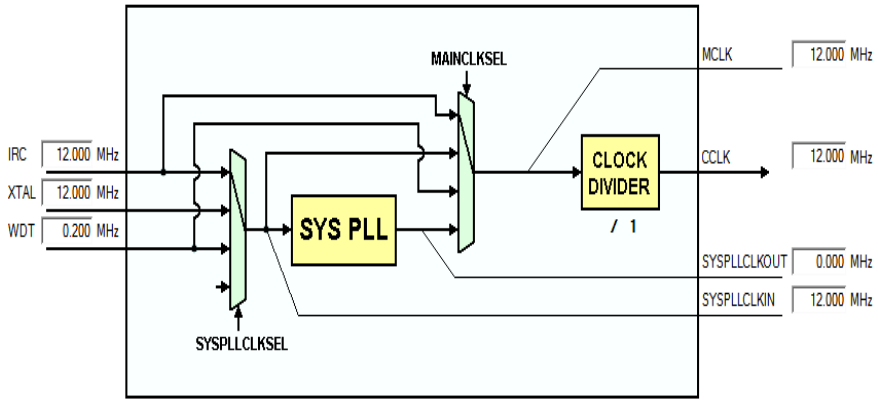
```
uint۸_t (*can_receive)(CAN_MSG_OBJ * CANopen_Msg_Obj);
void (*can_transmit)(CAN_MSG_OBJ * CANopen_Msg_Obj);
void (*config_canopen)(CAN_CANOPENCFG * canopen_cfg);
void (*canopen_handler)(void);
void (*config_calb)(CAN_CALLBACKS * callback_cfg);
}CAND;
```

تابع تنظیمات اولیه CAN (init_can)

شامل تنظیمات پری اسکیلر کلاک و رجیستر BTR یا Bus Timing است.

```
void CANopenInit(void)
{
CAN_MSG_OBJ CANopen_Msg_Obj;
uint۳۲_t ClkInitTable[۲] = {/* Initialize CAN Controller structure*/
 ۰x۰۰۰۰۰۰۰۰UL,/* CANCLKDIV */
 ۰x۰۰۰۰۱C۵۷UL/* CAN_BTR */
};
BitRate = Fcclk/(APBDIV * (BRP+۱) * ((Tseg۱+۱)+(Tseg۲+۱)+۱))
```

تنظیمات کلاک سیستم در پروژه حاضر:

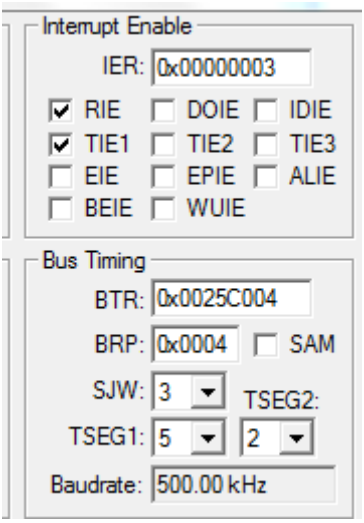



```
// Initialize CAN Controller @۱۲MHz

uint۳۲_t ClkInitTable[۲] =

{

 ۰x۰۰۰۰۰۰۰۰UL, // CANCLKDIV: /۱ = ۱۲MHz

 ۰x۱۲۰۱UL// CAN_BTR: ۱۰۰۰khz



};

//Initialize CAN Controller ۴۸MHz-۱۰۰۰kbps

uint۳۲_t ClkInitTable[۲] = {

 ۰x۰۰۰۰۰۰۰۰UL, //CANCLCLKDIV -> div: ۰
```

```
    ۰x۰۰۰۰۲BC۲UL, //CAN_BTR -> BRP: ۳, Quanta: ۱۶, Seg۱: ۱۲,
Seg۲: ۳, SJW: ۳, Sample ۸۱%
    };

    // Initialize CAN Controller @۴۸MHz ۵۰۰kHz
    uint۳۲_t ClkInitTable[۲] = {
     ۰x۰۰۰۰۰۰۰۲UL, // CANCLKDIV
     ۰x۱۰۷UL// CAN_BTR
    };
```

در نهایت با ۱ و ۰ نهایی تعیین می‌شود که آیا وقفه در کنترلر CAN رخ دهد یا به روش سرکشی عمل شود.

```
    /* Initialize the CAN controller */
    (*rom)->pCAND->init_can(&ClkInitTable[۰], ۱);
```



تابع تنظیمات در دریافت پیام‌ها config_rxmsgobj

در این مرحله تنظیمات دیکشنری برای دریافت انجام می‌شود.

```
typedef struct _CAN_MSG_OBJ {
 uint۳۲_t mode_id;
 uint۳۲_t mask;
 uint۸_t data[۸];
 uint۸_t dlc;
 uint۸_t msgobj;
}CAN_MSG_OBJ;
uint۸_t (*can_receive)(CAN_MSG_OBJ * CANopen_Msg_Obj);
```

/* Configure message object ۳ to receive all ۱۱-bit messages ۰x۷۰۰-۰x۷۷F: NMT EROOR */

CANopen_Msg_Obj.msgobj = ۳;

CANopen_Msg_Obj.mode_id = ۰x۷۰۰;

CANopen_Msg_Obj.mask = ۰x۷۸۰;

(*rom)->pCAND->config_rxmsgobj(&CANopen_Msg_Obj);

/* Configure message object ۵ to receive all ۱۱-bit messages ۰x۰۰۰: NMT Object */

CANopen_Msg_Obj.msgobj = ۵;

CANopen_Msg_Obj.mode_id = ۰x۰۰۰;

CANopen_Msg_Obj.mask = ۰x۷FF;

(*rom)->pCAND->config_rxmsgobj(&CANopen_Msg_Obj);

/* Configure message object ۷ to receive all ۱۱-bit messages ۰x۵۸۰-۵FF: SDO Server Object */

CANopen_Msg_Obj.msgobj = ۷;

CANopen_Msg_Obj.mode_id = ۰x۵۸۰;

CANopen_Msg_Obj.mask = ۰x۷۸۰;

(*rom)->pCAND->config_rxmsgobj(&CANopen_Msg_Obj);

تابع دریافت CAN

```
(*rom)->pCAND->can_receive(&CANopen_Msg_Obj);
```

تابع ارسال CAN

```
CANopen_Msg_Obj.msgobj = ۸;

CANopen_Msg_Obj.mode_id = ۰x۶۰۰ + CANopen_SDOC_Seg_ID;

CANopen_Msg_Obj.data[۰] = ۰x۶۰ | ((CANopen_Msg_Obj.data[۰]
& (۱<<۴)) ^ (۱<<۴));/* toggle */

CANopen_Msg_Obj.data[۱] = ۰x۰۰;

CANopen_Msg_Obj.data[۲] = ۰x۰۰;

CANopen_Msg_Obj.data[۳] = ۰x۰۰;

CANopen_Msg_Obj.data[۴] = ۰x۰۰;

CANopen_Msg_Obj.data[۵] = ۰x۰۰;

CANopen_Msg_Obj.data[۶] = ۰x۰۰;

CANopen_Msg_Obj.data[۷] = ۰x۰۰;

(*rom)->pCAND->can_transmit(&CANopen_Msg_Obj);
```

تنظیمات CANopenCFG

شامل یک ساختار جهت تنظیمات دیکشنری شیء و پروتکل SDO است و شامل موارد زیر است:

۱. شماره نود از ۱ تا ۱۲۷ که برای مستر و اسلیوها تنظیم می‌شود

۲. شماره شیء پیام دریافتی و ارسالی در SDO

۳. پرچم تصمیم‌گیری رخداد پاسخ‌گویی به SDO از طریق وقفه یا تابع

۴. دو پوینتر که اندازه دو جدول زیر را تعیین خواهد کرد

۵. جدول اول شامل ورودی‌های ثابت و فقط خواندی در دیکشنری اشیا است که در SDO تسریع شده فقط کاربرد دارند و طول اطلاعات ثبت شده در آن ۴ بایت یا کمتر (۴/۲/۱ بایت) است.

۶. جدول دوم شامل متغییر های دیکشنری شیء و مجوزهای نوشتن و خواندن است

```
#define CAN_MASTER_NODE      ۰x۷D   /* ۱۲۵ */

#define CAN_SLAVE۱_NODE ۰x۰۱/* ۱ */

#define CAN_SLAVE۲_NODE ۰x۰۲/* ۲ */

#define CAN_NODE_ID      CAN_MASTER_NODE


CAN_CANOPENCFG myCANopen =

{

CAN_NODE_ID,     /* node_id */

۱,/* msgobj_rx */

۲,/* msgobj_tx */
```

۱,/* isr_handled true=۱, false=۰ */

(uint۳۲_t)NULL, /* od_const_num, set to right value in CANopen initialization */

(CAN_ODCONSTENTRY *)myConstOD, /* od_const_table */

(uint۳۲_t)NULL,/* od_num, set to right value in CANopen initialization */

(CAN_ODENTRY *)myOD,/* od_table */

};

typedef struct _CAN_CANOPENCFG {

uint۸_t node_id;

uint۸_t msgobj_rx;

uint۸_t msgobj_tx;

uint۸_t isr_handled;

uint۳۲_t od_const_num;

CAN_ODCONSTENTRY *od_const_table;

uint۳۲_t od_num;

CAN_ODENTRY *od_table;

}CAN_CANOPENCFG;

مثال از فراخوانی و انجام تنظیمات:

myCANopen.od_const_num = NumberOfmyConstODEntries;

myCANopen.od_num = NumberOfmyODEntries;

```
(*rom)->pCAND->config_canopen((CAN_CANOPENCFG
*)&myCANopen);
```

انواع مجوزهای نوشتن و خواندن در بایت‌ها:

```
#define OD_NONE ۰x۰۰/* Object Dictionary entry doesn't exist */

#define OD_EXP_RO ۰x۱۰/* Object Dictionary entry expedited,
read-only */

#define OD_EXP_WO ۰x۲۰/* Object Dictionary entry expedited,
write-only */

#define OD_EXP_RW ۰x۳۰/* Object Dictionary entry expedited,
read-write */

#define OD_SEG_RO ۰x۴۰/* Object Dictionary entry segmented,
read-only */

#define OD_SEG_WO ۰x۵۰/* Object Dictionary entry segmented,
write-only */

#define OD_SEG_RW ۰x۶۰/* Object Dictionary entry segmented,
read-write */
```

جدول دوم شامل متغییر های دیکشنری شیء ورودی و مجوزهای نوشتن و خواندن است در SDO تسریع شده طول اطلاعات ثبت شده باید (۴/۲/۱ بایت) باشد و درSDO منقطع اندازه مهم نیست.

```
Typedef struct _CAN_ODENTRY {

uint۱۶_t index;
```

```
uint۸_t subindex;
uint۸_t entrytype_len;
uint۸_t *val;
}CAN_ODENTRY;
```

این تنظیم وجود داردCAN_Node_Def.Cمثالی از جدول دوم: که در

```
/* Application variables used in variable OD */
uint۸_t error_register;
uint۸_t LEDArray;
volatile SDOS_Buffer_t SDOS_۲۲۰۰ =
{
(uint۸_t*)SDOS_۲۲۰۰_Data,
sizeof(SDOS_۲۲۰۰_Data),
};
volatile uint۸_t SDOS_۲۲۰۰_Data[۲۵۵] = "Boot-up value of SDO ۲۲۰۰h";
uint۳۲_t CANopen_Heartbeat_Producer_Value;
WatchNode_t WatchList[۱];
uint۸_t WatchListLength = sizeof(WatchList)/sizeof(WatchList[۰]);
```

```
CAN_ODENTRY myOD [] =
{
/* index, subindex,access_type | length,        value_pointer */
{ ۰x۱۰۰۱, ۰x۰۰, OD_EXP_RO | ۱, (uint۸_t *)&Error_Register },
{ ۰x۱۰۱۶, ۰x۰۰, OD_EXP_RO | ۱, (uint۸_t *)&WatchListLength},
{ ۰x۱۰۱۶, ۰x۰۱, OD_EXP_RW | ۴, (uint۸_t *)&WatchList[۰].value},
{ ۰x۱۰۱۶, ۰x۰۲, OD_EXP_RW | ۴, (uint۸_t *)&WatchList[۱].value},
{ ۰x۱۰۱۷, ۰x۰۰, OD_EXP_RW | ۲, (uint۸_t *)&CANopen_Heartbeat_Producer_Value},
};//master and Slave
//{ ۰x۲۰۰۰, ۰x۰۰, OD_EXP_RW | ۱, (uint۸_t *)&LEDArray},//slave
//{ ۰x۲۲۰۰, ۰x۰۰,OD_SEG_RW, (uint۸_t *)&SDOS_۲۲۰۰},//slave
```

پوینتر اندازه این جدول:

```
uint۳۲_t NumberOfmyODEntries = sizeof(myOD)/sizeof(myOD[۰]);
```

جدول اول شامل ورودی‌های ثابت و فقط خواندی در دیکشنری اشیا است که در SDO تسریع شده فقط کاربرد دارند و طول اطلاعات ثبت شده در آن ۴ بایت یا کمتر (۴/۲/۱ بایت) است.

```
Typedef struct _CAN_ODCONSTENTRY {
uint۱۶_t index;
uint۸_t subindex;
uint۸_t len;
uint۳۲_t val;
}CAN_ODCONSTENTRY;
```

مثالی از جدول ۱: که در CAN_Node_Def.C این تنظیم وجود دارد.

```
CAN_ODCONSTENTRY myConstOD [] =
{
/* index, sub index, length, value */
{ ۰x۱۰۰۰, ۰x۰۰, ۴, ۰x۰۰۰۰۰۰۰۰UL },
{ ۰x۱۰۱۸, ۰x۰۰, ۱, ۰x۰۰۰۰۰۰۰۱UL },/* only vendor ID is specified */
{ ۰x۱۰۱۸, ۰x۰۱, ۴, ۰x۰۰۰۰۰۲DCUL },/* NXP vendor ID for CANopen */
};
```

پوینتر اندازه این جدول:

```
uint۳۲_t NumberOfmyConstODEntries = sizeof(myConstOD) / sizeof
(myConstOD[۰]);
```

تابع فرآخوانی تمام توابع APIها (CALLBACKS)

این تابع خواندن و نوشتن تسریع شده و منقطع را در canopen تعریف و آماده به کار می‌نماید.

```
CAN_CALLBACKS callbacks = {
CAN_RX,/* callback for any message received CAN frame which ID matches */
CAN_TX,/* callback for every transmitted CAN frame */
CAN_Error,/* callback for CAN errors */
CANopen_SDOS_Exp_Read,        /* callback for expedited read access (SDO server) */
CANopen_SDOS_Exp_Write,       /* callback for expedited write access (SDO server) */
CANopen_SDOS_Seg_Read,        /* callback for segmented read access (SDO server) */
CANopen_SDOS_Seg_Write,       /* callback for segmented write access (SDO server) */
NULL,/* callback for fall-back SDO handler (not used) */
};
/* Configure the CAN callback functions */
(*rom)->pCAND->config_calb(&callbacks);
```

اطلاعات موجود در CAN_Node_Def.h

۱. شماره نودهای مستر و اسلیو

۲. مقدار زمان TIMEOUT بر اساس میلی ثانیه برای کلاینت‌ها در SDO

۳. دیکشنری اشیاء و متغییرها متصل شده به دیکشنری اشیا

۴. لیست نودهای قابل مشاهده

```
#define CAN_MASTER_NODE      ۰x۷D  /* ۱۲۵ */

#define CAN_SLAVE۱_NODE ۰x۰۱/* ۱ */

#define CAN_SLAVE۲_NODE ۰x۰۲/* ۲ */

#define CAN_NODE_ID     CAN_MASTER_NODE

#define CANOPEN_TIMEOUT_VAL          ۱۰۰     /* in ms */

extern uint۸_t error_register;/* CANopen error register */

extern uint۸_t LEDArray;/* LEDs of MCB۱۱C۱۴ board */

extern uint۳۲_t CANopen_Heartbeat_Producer_Value;/* heartbeat producer value */

extern volatile SDOS_Buffer_t SDOS_۲۲۰۰;/* buffer structure associated with segmented entry ۲۲۰۰h */

extern volatile uint۸_t SDOS_۲۲۰۰_Data[۲۵۵];/* buffer associated with segmented entry ۲۲۰۰h */

extern WatchNode_t WatchList[];     /* for watching nodes */
```

```
extern uint8_t WatchListLength;      /* number of nodes in watchlist must be known */
```

تابع main.c مستر

۱- زمان هارت بیت برای تولید کننده و مصرف کننده.

```
#define HEARTBEAT_PRODUCER_TIME ۴۰۰۰/* in ms */

#define HEARTBEAT_CONSUMER_TIME ۴۵۰۰/* in ms */
```

۲- فرآخوانی تابع انجام تنظیمات سیستمی و تنظیمات CANopenInit که قبلاً توضیح داده شده است.

```
SystemInit();
CANopenInit();
```

۳- فعال سازی تایمر صفر

```
/* Timer۱۶B۰ */

LPC_SYSCON->SYSAHBCLKCTRL |= (۱<<۷);/* Enable clock for ۱۶-bit counter/timer ۰ */

LPC_TMR۱۶B۰->MCR = ۳;/* Interrupt and Reset on MR۰ */
```

LPC_TMR۱۶B۰->MR۰ = SystemCoreClock/۱۰۰۰;/* Interrupt and Reset on MR۰ */

NVIC_EnableIRQ(TIMER_۱۶_۰_IRQn);/* Enable the Timer۰ Interrupt */

LPC_TMR۱۶B۰->TCR = ۱;/* Enable Timer۰ */

۴- تولید لیست مشاهده نودها در تنظیمات مستر

WatchList[۰].NodeID = CAN_SLAVE۱_NODE;

WatchList[۰].ConsumerTime = HEARTBEAT_CONSUMER_TIME;

WatchList[۰].ProducerTime = HEARTBEAT_PRODUCER_TIME;

WatchList[۱].NodeID = CAN_SLAVE۲_NODE;

WatchList[۱].ConsumerTime = HEARTBEAT_CONSUMER_TIME;

WatchList[۱].ProducerTime = HEARTBEAT_PRODUCER_TIME;

۵- انجام تنظیمات اسلیو اول

برای انجام تنظیمات هارت بیت را غیر فعال می‌کنیم.

//Configure slave node ۱

WatchList[۰].value = ۰x۰۰; /* Disable heartbeat checking while configuring */

مقدار زمان هارت بیت تولید کننده به آدرس ایندکس ۱۰۱۷x۰ و زیر اندکس ۰ از طریق SDO_EXP یا تسریع شده برای اسلیو اول ارسال می‌شود.

```
Buffer[۰] = (WatchList[۰].ProducerTime >> ۸) & ۰xFF;

buffer[۱] = (WatchList[۰].ProducerTime >> ۰) & ۰xFF;

CANopen_SDOC_Exp_Write(CAN_SLAVE۱_NODE, ۰x۱۰۱۷,
۰x۰۰, (uint۸_t*)buffer, ۲);
```

تا دریافت ACK از طرف اسلیو منتظر می‌مانیم

```
while(!(CANopen_SDOC_State == CANopen_SDOC_Succes ||
CANopen_SDOC_State == CANopen_SDOC_Fail));
```

عملیات برگرداندن مقدار هارت بیت برای فعال سازی مجدد آن انجام می‌شود.

```
if(CANopen_SDOC_State == CANopen_SDOC_Succes)

WatchList[۰].value = (WatchList[۰].NodeID)<<۱۶ |
WatchList[۰].ConsumerTime;/* node ۱ */
```

مقدار هارت بیت مصرف کننده به آدرس ایندکس ۱۰۱۶x۰ و زیر اندکس ۰۱x۰ که برای]value۰WatchList[است از طریق SDO_EXP یا تسریع شده برای اسلیو اول ارسال می‌شود.

```
buffer[۰] = ۰x۰۰;

buffer[۱] = CAN_NODE_ID;

buffer[۲] = (WatchList[۰].ConsumerTime >> ۸) & ۰xFF;

buffer[۳] = (WatchList[۰].ConsumerTime >> ۰) & ۰xFF;
```

```
CANopen_SDOC_Exp_Write(CAN_SLAVE۱_NODE, ۰x۱۰۱۶, ۰x۰۱, (uint۸_t*)buffer, ۴);
```

تا دیافت ACK از طرف اسلیو منتظر می‌مانیم

```
while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));
```

۶- انجام تنظیمات اسلیو دوم

برای انجام تنظیمات هارت بیت را غیر فعال می‌کنیم.

```
WatchList[۱].value = ۰x۰۰; /* Disable heartbeat checking while configuring */
```

مقدار هارت بیت تولید کننده به آدرس ایندکس ۱۰۱۷x۰ و زیر اندکس ۰ از طریق SDO_EXP یا تسریع شده برای اسلیو دوم ارسال می‌شود.

```
buffer[۰] = (WatchList[۱].ProducerTime >> ۸) & ۰xFF;

buffer[۱] = (WatchList[۱].ProducerTime >> ۰) & ۰xFF;

CANopen_SDOC_Exp_Write(CAN_SLAVE۲_NODE, ۰x۱۰۱۷, ۰x۰۰, (uint۸_t*)buffer, ۲);
```

تا دریافت ACK از طرف اسلیو منتظر می‌مانیم

```
while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));
```

عملیات برگرداندن مقدار هارت بیت برای فعال سازی مجدد آن انجام می‌شود.

```
if(CANopen_SDOC_State == CANopen_SDOC_Succes)
```

```
WatchList[۱].value = (WatchList[۱].NodeID)<<۱۶ | WatchList[۱].ConsumerTime;/* node ۲ */
```

مقدار هارت بیت مصرف کننده به آدرس ایندکس ۱۰۱۶x۰ و زیر اندکس ۰۱x۰ که برای]value۰WatchList[است از طریق SDO_EXP یا تسریع شده برای اسلیو دوم ارسال می‌شود.

```
buffer[۱] = CAN_NODE_ID;

buffer[۲] = (WatchList[۱].ConsumerTime >> ۸) & ۰xFF;

buffer[۳] = (WatchList[۱].ConsumerTime >> ۰) & ۰xFF;

CANopen_SDOC_Exp_Write(CAN_SLAVE۲_NODE, ۰x۱۰۱۶, ۰x۰۱, (uint۸_t*)buffer, ۴);
```

تا دریافت ACK از طرف اسلیو منتظر می‌مانیم

```
while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));
```

۷- نوشتن اطلاعات در اسلیو اول

دریافت اطلاعات یک رشته از پورت سریال

```
fgets(buffer, sizeof(buffer), stdin);

i = ۰;

while(buffer[i])

i++;
```

نوشتن در آدرس ایندکس ۲۲۰۰x۰ و زیر ایندکس ۰۰x۰ در اسلیو اول توسط تابع SDO منقطع

```
CANopen_SDOC_Seg_Write(CAN_SLAVE۱_NODE, ۰x۲۲۰۰, ۰x۰۰, buffer, i-۱);
```

تا دریافت ACK از طرف اسلیو منتظر می‌مانیم

```
while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));
```

۸- نوشتن اطلاعات در اسلیو دوم

دریافت اطلاعات یک رشته از پورت سریال

```
fgets(buffer, sizeof(buffer), stdin);
i = ۰;
while(buffer[i])
i++;
```

نوشتن در آدرس ایندکس ۲۲۰۰x۰ و زیر ایندکس ۰۰x۰ در اسلیو دوم توسط تابع SDO منقطع

```
CANopen_SDOC_Seg_Write(CAN_SLAVE۲_NODE, ۰x۲۲۰۰, ۰x۰۰, buffer, i);
```

تا دریافت ACK از طرف اسلیو منتظر می‌مانیم

```
while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));
```

۹- خواندن ازاسلیو اول

برای این منظور خواندن از آدرس نود اسلیو اول و ایندکس ۲۲۰۰x۰ و زیر ایندکس صفرم و نوشتن در آرایه بافر از دستور زیر استفاده می‌شود.

```
CANopen_SDOC_Seg_Read(CAN_SLAVE۱_NODE,۰ x۲۲۰۰, ۰x۰۰, buffer, sizeof(buffer));
```

برای این منظور خواندن از همان آدرس و چاپ در پورت سریال دستور

```
Print_SDO_SEG_Entry(CAN_SLAVE۱_NODE, ۰x۲۲۰۰, ۰x۰۰);
```

۱۰- خواندن ازاسلیو دوم

برای این منظور خواندن از آدرس نود اسلیو دوم و ایندکس ۲۲۰۰x۰ و زیر ایندکس صفرم و نوشتن در آرایه بافر از دستور زیر استفاده می‌شود.

```
CANopen_SDOC_Seg_Read(CAN_SLAVE۲_NODE,۰ x۲۲۰۰, ۰x۰۰, buffer, sizeof (buffer));
```

برای این منظور خواندن از همان آدرس و چاپ در پورت سریال دستور

```
Print_SDO_SEG_Entry(CAN_SLAVE۲_NODE, ۰x۲۲۰۰, ۰x۰۰);
```

۱۱- نوشتن یک مقدار عدد هگزا دسیمال بر روی LEDها اسلیو اول

دریافت اطلاعات یک رشته از پورت سریال

```
fgets(buffer, sizeof(buffer), stdin);
i = sscanf(buffer,"%X", (uint۳۲_t*)&buffer[۰]);
```

نوشتن در آدرس ایندکس ۲۰۰۰x۰ و زیر ایندکس ۰۰x۰ در اسلیو اول که برای نمایش LED از قبل تعیین کرده‌ایم توسط تابع SDO تسریع شده

```
CANopen_SDOC_Exp_Write(CAN_SLAVE۱_NODE, ۰x۲۰۰۰, ۰x۰۰, (uint۸_t*)buffer, ۱);
```

تا دریافت ACK از طرف اسلیو منتظر می‌مانیم

```
while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));
```

۱۲- خواندن وضعیت LED های اسلیو اول توسط مستر

خواندن از آدرس ایندکس ۲۰۰۰x۰ و زیر ایندکس ۰۰x۰ از اسلیو اول که برای نمایش LED از قبل تعیین کرده‌ایم توسط تابع SDO تسریع شده

```
CANopen_SDOC_Exp_Read(CAN_SLAVE۱_NODE, ۰x۲۰۰۰, ۰x۰۰, (uint۸_t*)buffer, NULL);
```

تا دیافت اتمام دریافت منتظر می‌مانیم

```
while(!(CANopen_SDOC_State == CANopen_SDOC_Succes || CANopen_SDOC_State == CANopen_SDOC_Fail));
```

اگر ارسال به درستی انجام شده باشد مقدار متغییر LEDArray با وضعیت LED در اسلیو اول برابر خواهد شد.

```
if(CANopen_SDOC_State == CANopen_SDOC_Succes)
                                        ];۰LEDArray = buffer[
```

۱۳- فراخوانی تابع ۱ میلی ثانیه در روتین وقفه تایمر صفر

```
void TIMER۱۶_۰_IRQHandler(void)

{

static uint۳۲_t PrevButtons;

LPC_TMR۱۶B۰->IR = ۱;/* برای وقفه دوباره */

PrevButtons = (LPC_GPIO۰->DATA);

CANopen_۱ms_tick();

}
```

تابع main.c اسلیو

۱- فعال سازی تایمر صفر

```
/* Timer۱۶B۰ */

LPC_SYSCON->SYSAHBCLKCTRL |= (۱<<۷);    /* Enable clock for ۱۶-bit counter/timer ۰ */

LPC_TMR۱۶B۰->MCR = ۳;/* Interrupt and Reset on MR۰ */

LPC_TMR۱۶B۰->MR۰ = SystemCoreClock/۱۰۰۰; /* Interrupt and Reset on MR۰ */

NVIC_EnableIRQ(TIMER_۱۶_۰_IRQn);/* Enable the Timer۰ Interrupt */

LPC_TMR۱۶B۰->TCR = ۱;/* Enable Timer۰ */
```

۲- فرآخوانی تابع انجام تنظیمات سیستمی و تنظیمات CANopenInit که قبلاً توضیح داده شده است.

```
SystemInit();
CANopenInit();
```

۳- فرآخوانی تابع ۱ میلی ثانیه در روتین وقفه تایمر صفر

```
void TIMER۱۶_۰_IRQHandler(void)
{
static uint۳۲_t PrevButtons;
LPC_TMR۱۶B۰->IR = ۱;/* clear interrupt flag */
/* CANopen ۱ms tick */
CANopen_۱ms_tick();
 /* save button state */
PrevButtons = (LPC_GPIO۰->DATA &);
}
```

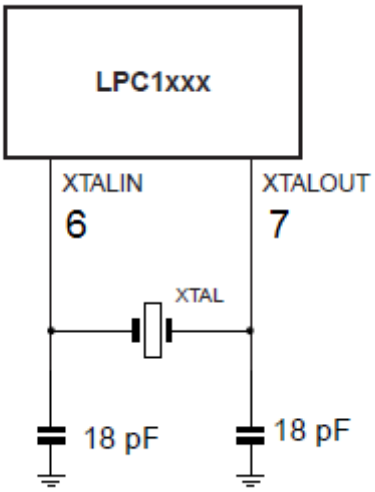
۴- در CAN_Node_Def.C این تنظیم درباره دیکشنری شیء اضافه بر آنچه در مستر وجود دارد درباره وجود LEDArray وجود دارد.

```
CAN_ODENTRY myOD [] =
{
/* index, subindex,access_type | length,        value_pointer */
//master and Slave
//{     ۰x۲۰۰۰,     ۰x۰۰,     OD_EXP_RW     |     ۱,     (uint۸_t
*)&LEDArray},//slave
//{ ۰x۲۲۰۰, ۰x۰۰,OD_SEG_RW, (uint۸_t *)&SDOS_۲۲۰۰},//slave
}
```

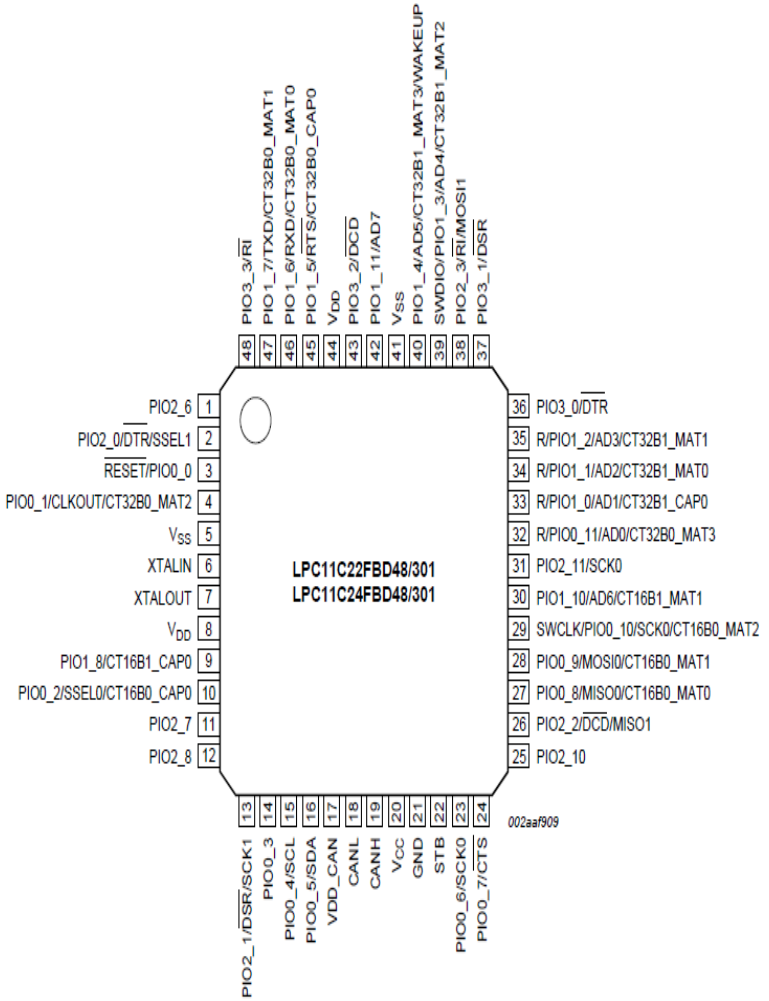
معرفی تراشه LPC۱۱C۲۴

جهت پیاده سازی پروتکل CANopen از این تراشه استفاده خواهد شد.

این تراشه یک پردازنده ARM Cortex-M۰ ۳۲ بیتی ساخت شرکت NXP است با ۳۲ کیلوبیت حافظه فلش و ۸ کیلوبیت حافظه Sram و با حداکثر فرکانس کاری ۵۰ مگاهرتز می‌باشد کلاک این میکرو کنترلر می‌تواند توسط اسیلاتور RC داخلی به اندازه ۱۲ مگاهرتز و یا توسط کریستال خارجی ۱ تا ۲۵ مگا هرتزی و PLL جهت افزایش آن تأمین شود. تغذیه این تراشه ۳.۳ ولتی است که باید دو پایه ۸ و ۴۴ را به ۳.۳+ و دو پایه ۵ و ۴۱ را زمین کنیم بهتر است در بین این پایه‌ها دو مدل خازن ۱۰۰ نانو فارادی قرار گیرند. برای قسمت CAN نیز تغذیه ۵ ولتی توصیه شده و موردنیاز است. به علت حساسیت تراشه به ولتاژ منفی در ورودی یک دیود قرار گیرد و ولتاژها از رگولاتورهای ۵ و ۳.۳ تأمین شده باشد.



این تراشه دارای ۴۸ پایه به فوت پرینت LQFP است و شامل ۳۶ پایه GPIO در ۴ پورت به نام‌های صفر، ۱, ۲, ۳، یک واحد USART، یک واحد C2I، دو تایمر کانتر ۱۶ بیتی، دو تایمر کانتر ۳۲ بیتی، یک واحد ADC، یک واحد WTD، دو واحد SPI و یک واحد کنترلر C_CAN می‌باشد.


LPC11C22FBD48/301
LPC11C24FBD48/301
002aaf909
1 PIO2_6
2 PIO2_0/DTR/SSEL1
3 RESET/PIO0_0
4 PIO0_1/CLKOUT/CT32B0_MAT2
5 VSS
6 XTALIN
7 XTALOUT
8 VDD
9 PIO1_8/CT16B1_CAP0
10 PIO0_2/SSEL0/CT16B0_CAP0
11 PIO2_7
12 PIO2_8
13 PIO2_1/DSR/SCK1
14 PIO0_3
15 PIO0_4/SCL
16 PIO0_5/SDA
17 VDD_CAN
18 CANL
19 CANH
20 VCC
21 GND
22 STB
23 PIO0_6/SCK0
24 PIO0_7/CTS
25 PIO2_10
26 PIO2_2/DCD/MISO1
27 PIO0_8/MISO0/CT16B0_MAT0
28 PIO0_9/MOSI0/CT16B0_MAT1
29 SWCLK/PIO0_10/SCK0/CT16B0_MAT2
30 PIO1_10/AD6/CT16B1_MAT1
31 PIO2_11/SCK0
32 R/PIO0_11/AD0/CT32B0_MAT3
33 R/PIO1_0/AD1/CT32B1_CAP0
34 R/PIO1_1/AD2/CT32B1_MAT0
35 R/PIO1_2/AD3/CT32B1_MAT1
36 PIO3_0/DTR
37 PIO3_1/DSR
38 PIO2_3/RI/MOSI1
39 SWDIO/PIO1_3/AD4/CT32B1_MAT2
40 PIO1_4/AD5/CT32B1_MAT3/WAKEUP
41 VSS
42 PIO1_11/AD7
43 PIO3_2/DCD
44 VDD
45 PIO1_5/RTS/CT32B0_CAP0
46 PIO1_6/RXD/CT32B0_MAT0
47 PIO1_7/TXD/CT32B0_MAT1
48 PIO3_3/RI

پایه‌های SPI:

MISO0 (Master In Slave Out for SSP0. input SSP0 master/output from SSP0 slave) [P0.8]
MISO1 (Master In Slave Out for SSP1. input SSP1 master/output from SSP1 slave) [P2.2]
MOSI0 (Master Out Slave In for SSP0. out from SSP0 master/input to SSP0 slave) [P0.9]
MOSI1 (Master Out Slave In for SSP1. out from SSP1 master/input to SSP1 slave) [P2.3]
SCK0 (Serial clk for SSP0. SSP0 clk out from master/input to slave) [P0.10]
SCK0 (Serial clk for SSP0. SSP0 clk out from master/input to slave) [P0.6]
SCK0 (Serial clk for SSP0. SSP0 clk out from master/input to slave) [P2.11]
SCK1 (Serial clk for SSP1. SSP1 clk out from master/input to slave) [P2.1]
SSEL0 (Slave Select for SSP0. Selects the SSP0 interface as a slave) [P0.2]
SSEL1 (Slave Select for SSP1. Selects the SSP1 interface as a slave) [P2.0]

پایه‌های C۲I:

SCL (I2C clk I/O. Open-drain output) [P0.4]
SDA (I2C data I/O. Open-drain output) [P0.5]

پایه‌های ADC: این واحد شامل ۸ کانال ورودی آنالوگ و ۱۰ بیتی با زمان تبدیل ۴۴.۲ میکرو ثانیه و یا ۴۰۰ kSamples/s می‌باشد و به صورت معمول از صفر تا VDD که معمولاً ۳.۳ ولت است کار می‌کند.

AD0.0 (A/D converter 0 input 0) [P0.11]
AD0.1 (A/D converter 0 input 1) [P1.0]
AD0.2 (A/D converter 0 input 2) [P1.1]
AD0.3 (A/D converter 0 input 3) [P1.2]
AD0.4 (A/D converter 0 input 4) [P1.3]
AD0.5 (A/D converter 0 input 5) [P1.4]
AD0.6 (A/D converter 0 input 6) [P1.10]
AD0.7 (A/D converter 0 input 7) [P1.11]

پایه‌های USART:

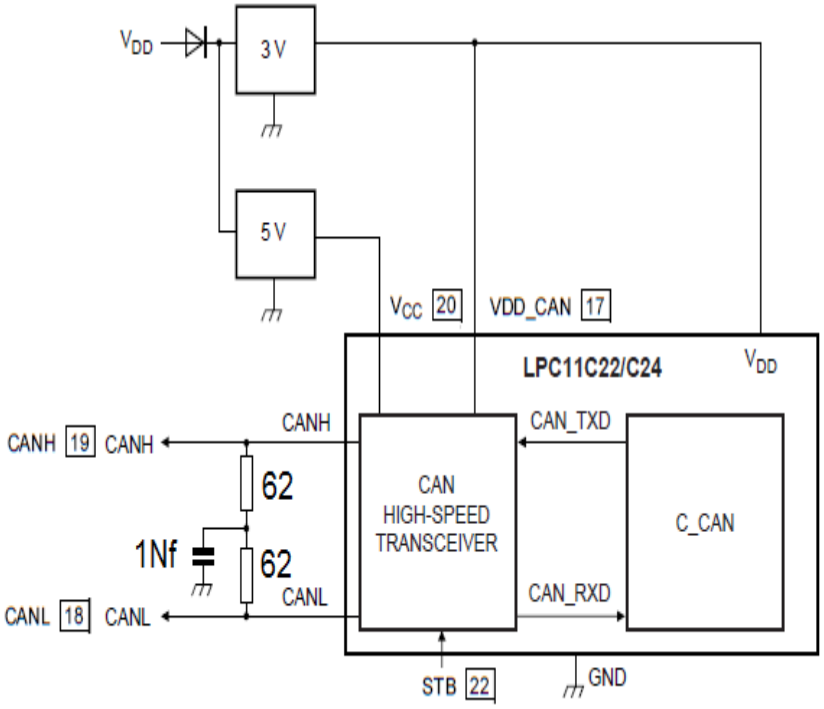
RXD (Receiver input for UART) [P1.6]
TXD (Transmitter output for UART) [P1.7]

پایه‌های کپچر و برابری با مقدار خاص تایمرها به صورت زیر است:

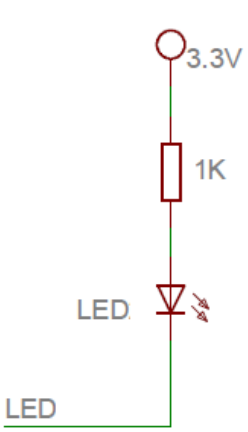
CAP0.0 (Capture input for Timer 0 channel 0 [CT32B0]) [P1.5]
CAP1.0 (Capture input for Timer 1 channel 0 [CT32B1]) [P1.0]
CAP2.0 (Capture input for Timer 2 channel 0 [CT16B0]) [P0.2]
CAP3.0 (Capture input for Timer 3 channel 0 [CT16B1]) [P1.8]
MAT0.0 (Match output for Timer 0 channel 0 [CT32B0]) [P1.6]
MAT0.1 (Match output for Timer 0 channel 1 [CT32B0]) [P1.7]
MAT0.2 (Match output for Timer 0 channel 2 [CT32B0]) [P0.1]
MAT0.3 (Match output for Timer 0 channel 3 [CT32B0]) [P0.11]
MAT1.0 (Match output for Timer 1 channel 0 [CT32B1]) [P1.1]
MAT1.1 (Match output for Timer 1 channel 1 [CT32B1]) [P1.2]
MAT1.2 (Match output for Timer 1 channel 2 [CT32B1]) [P1.3]
MAT1.3 (Match output for Timer 1 channel 3 [CT32B1]) [P1.4]
MAT2.0 (Match output for Timer 2 channel 0 [CT16B0]) [P0.8]
MAT2.1 (Match output for Timer 2 channel 1 [CT16B0]) [P0.9]
MAT2.2 (Match output for Timer 2 channel 2 [CT16B0]) [P0.10]
MAT3.1 (Match output for Timer 3 channel 1 [CT16B1]) [P1.10]

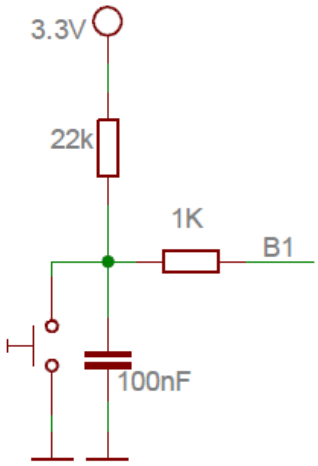
واحد کنترلر C_CAN این امکان را فراهم می‌سازد تا توسط توابع نرم‌افزاری مناسب شبکه CANopen را با سرعت Mbit/s ۱ پیاده سازی گردد. این واحد دارای یک لایه سخت‌افزاری داخلی پرسرعت CAN نیز می‌باشد. پایه STB نیز در حالت کار نرمال باید با مقاومتی ۲ کیلو اهمی Pull Down یا صفر باشد. حالت یک این پایه مربوط به مد Silent است که در آن فرستنده خاموش و گیرنده فعال است و این مد خاص برای جلوگیری از بروز خطا و از

کار افتادن سیستم کنترلر استفاده می‌شود. البته VDD_CAN و VCC می‌توان ۵ ولت نیز باشد و برای آن خازنی ۱۰۰ نانو فاراد زمین کنیم. مقاومت‌های روی بأس ۶۰ اهم معمولاً هستند.

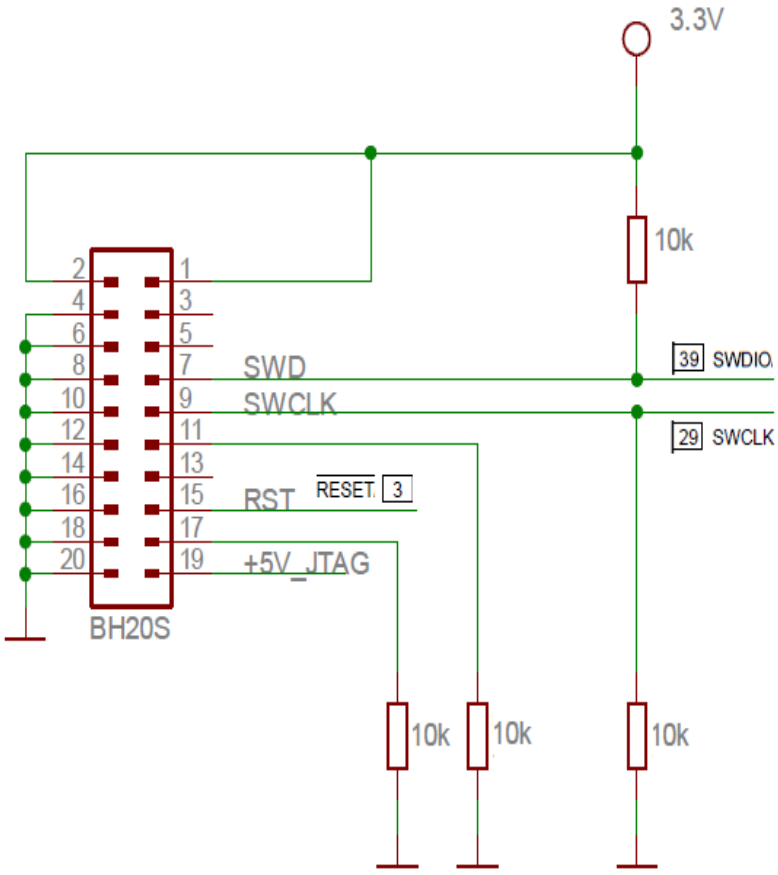


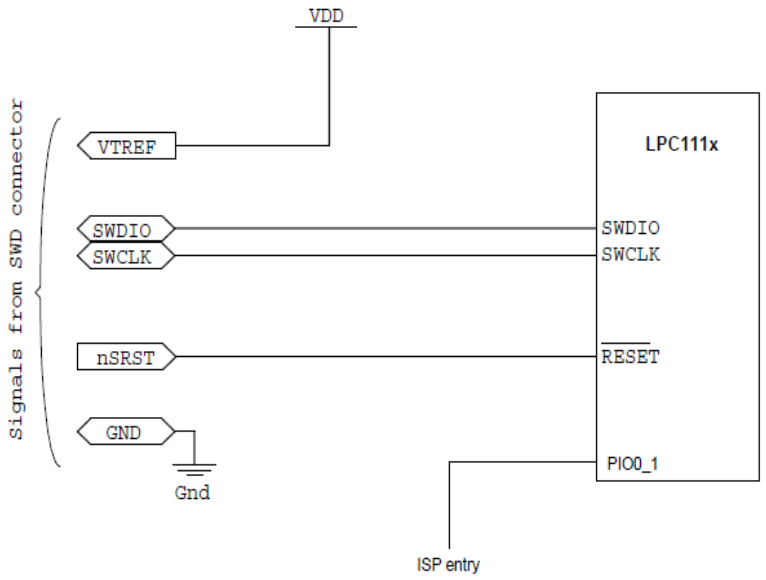
مدارات مناسب برای LED و کلید در برد راه‌انداز:





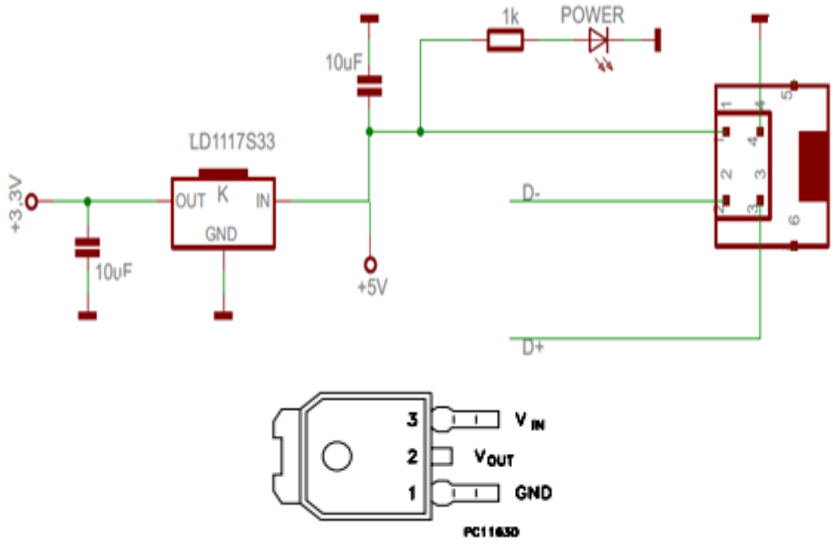
محل اتصال دیباگرJlink :



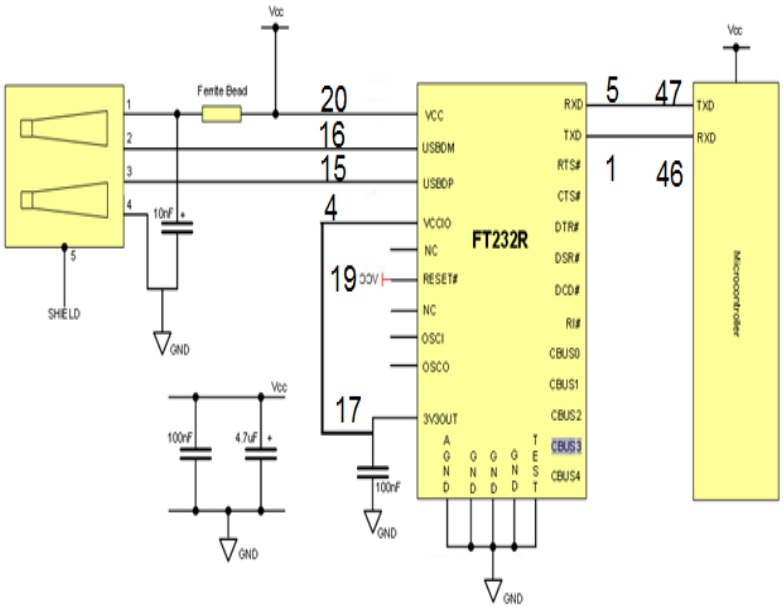


The VTREF pin on the SWD connector enables the debug connector to match the target voltage.

جهت تغذیه ۵ و ۳.۳ می‌توان از مدار زیر استفاده کرد. البته هسته فریت و خازن نباید فراموش شود.



برای آنکه پایه‌های Usart دارای مشخصه ولتاژ مناسب برای میکرو کنترلر ۳.۳ ولتی باشد از این آرایش استفاده خواهیم کرد.


Vcc
Ferrite Bead
1
2
3
4
5
SHIELD
10nF
GND
20
16
15
4
19
17
VCC
USBDM
USBDP
VCCIO
NC
RESET#
NC
OSCI
OSCO
3V3OUT
FT232R
AGND
GND
GND
GND
TEST
RXD
TXD
RTS#
CTS#
DTR#
DSR#
DCD#
RI#
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
100nF
4.7uF
5
47
1
46
TXD
RXD
Microcontroller

25 21 18 7 26

فصل دوم

ارتباط سیستم کنترل و مانیتورینگ با سیستم های سیگنالینگ

مقدمه

آنچه در زیر مشاهده می‌فرمایید، نتیجه‌ی انجام مطالعات بر روی محصولات شرکت‌های توانمند جهان در حوزه ساخت اجزای مختلف سیستم کنترل و مانیتورینگ است. همان‌طور که از نام‌گذاری سیستم کنترل و مونیتورینگ مشخص است در یک ناوگان ریلی این سیستم باید با یکایک سیستم‌های قطار در ارتباط بوده و بر کار آنها نظارت داشته باشد.

سیستم‌های مختلفی که در هر ناوگان ریلی TCMS با آنها ارتباط دارد:

۱. سیستم سیگنالینگ
۲. سیستم کنترل ترمز
۳. سیستم ترکشن موتور
۴. سیستم تهویه هوا
۵. سیستم کنترل درب‌های قطار
۶. سیستم اطلاع‌رسانی صوتی
۷. سیستم اطلاع‌رسانی تصویری
۸. سیستم تأمین برق داخلی قطار و باتری شارژر

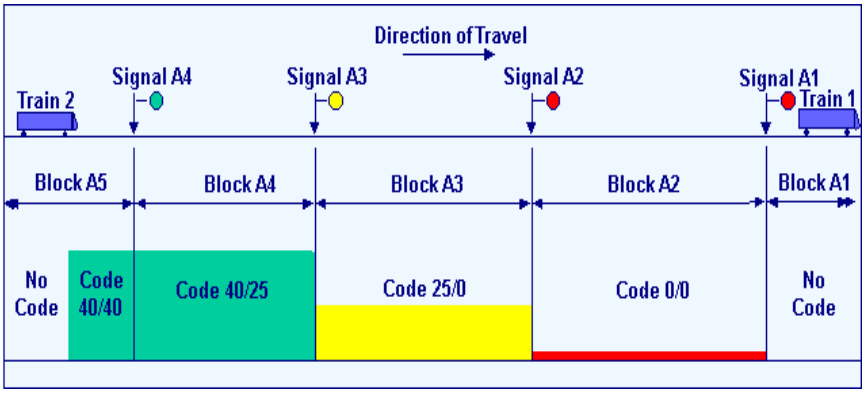
یکی از مهم‌ترین سیستم‌های همکار با TCMS سیستم سیگنالینگ است در این فصل و در ادامه با این سیستم و نحوه کار آن و در نهایت به ملزومات سخت‌افزاری اشاره خواهد شد.

سیستم‌های سیگنالینگ مورد استفاده در خطوط ریلی دارای بخش‌های مختلفی می‌باشند. یک سیستم سیگنالینگ کامل Automatic Train Control (ATC)نامیده می‌شود. این سیستم شامل بخش‌های مختلفی از قبیل ATP (Automatic Train Protection), ATO (Automatic Train Operation) و Automatic Train Supervision (ATS) می‌باشد که در ادامه به معرفی هر کدام از آنها خواهیم پرداخت. هر سیستم سیگنالینگ شامل دو بخش از تجهیزات می‌باشد. بخشی از تجهیزات که در قطار نصب می‌شوند که اصطلاحاً به آنها تجهیزات On-board گفته می‌شود و بخش دیگر تجهیزات که در خط نصب می‌گردد و به آنها تجهیزات Wayside گفته می‌شود.

سیستم ATP

مهم‌ترین وظیفه یک سیستم ATP در سیستم‌های سیگنالینگ، حفظ فاصله ایمنی لازم یک قطار از قطار جلوئی برای جلوگیری از هر گونه برخورد می‌باشد. بدین منظور هر سیستم ATP دارای یک سیستم کنترل در هر بلاک می‌باشد. سیستم کنترل هر بلاک، اطلاعات را از سیستم کنترل بلاک جلوئی دریافت کرده و با پردازش این اطلاعات، محدودیت سرعت را برای بلاک خود محاسبه می‌کند. این اطلاعات با استفاده از یکسری کدهای استاندارد از طریق آنتن‌های ATP به تجهیزات نصب شده در قطار منتقل شده و نهایتاً این محدودیت سرعت توسط راننده یا به صورت اتوماتیک توسط سیستم سیگنالینگ در حرکت قطار اعمال می‌گردد. در ATP های اولیه، کدها شامل

سه کد سرعت نرمال، سرعت هشدار (Caution speed) و سرعت صفر (Zero speed) بود که به صورت رنگ‌های سبز، زرد و قرمز در کابین راننده برای او نمایش داده می‌شد با پیشرفت سیستم‌های ATP کدهای سرعت به صورت سرعت‌های مجاز در هر بلاک محاسبه و در مقابل راننده نمایش داده می‌شد. با پیشرفت بیشتر این سیستم‌ها و به منظور داشتن حرکت نرمتر قطار، علاوه بر محاسبه سرعت مجاز در بلاک فعلی، سرعت مجاز در بلاک بعدی نیز محاسبه و برای راننده نمایش داده می‌شود.

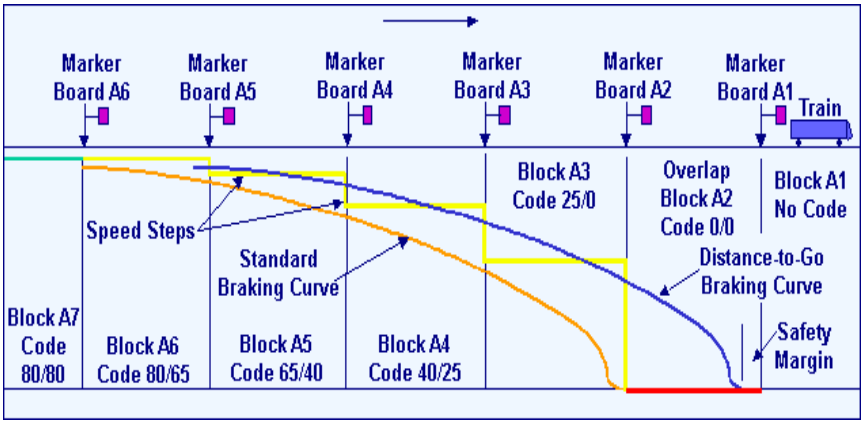


شکل ۲-۲-۱ کدهای سرعت مورد استفاده در سیستم‌های ATP پیشرفته‌تر

همان‌طور که در شکل ملاحظه می‌کنید، قطار شماره ۲ واقع در بلاک ۵A سرعت مجاز ۴۰/۴۰ (۴۰ روی ۴۰) را دریافت می‌کند و مفهوم آن این است که قطار با سرعت ۴۰ کیلومتر بر ساعت می‌تواند در بلاک ۵A حرکت کرده و با همان سرعت وارد بلاک ۴A گردد. در بلاک ۴A محدودیت سرعت ۲۵/۴۰ اعلام می‌گردد و مفهوم آن این است که قطار باید با سرعت ۲۵ کیلومتر بر ساعت وارد بلاک ۳A گردد. در سیستم‌های بدون ATO این محدودیت

سرعت‌ها در روی نمایشگرهایی برای راننده نمایش داده می‌شود و راننده بر اساس این اطلاعات حرکت قطار را کنترل می‌نماید؛ اما در سیستم‌های مجهز به ATO این محدودیت‌ها توسط سیستم ATO در حرکت قطار اعمال می‌گردد.

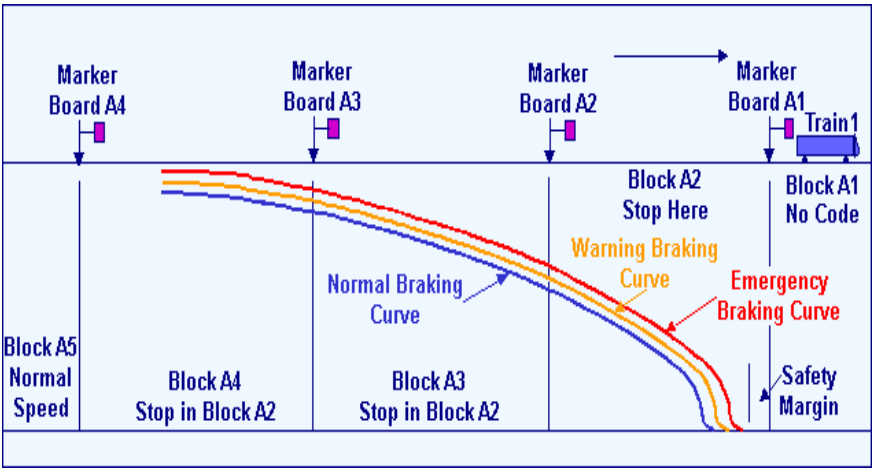
یکی از اشکالات ATP هایی که با روش فوق (Step speed) کار می‌کنند این است که همان‌طور که در شکل قبل ملاحظه می‌کنید، همواره بلاک بعدی قطار در حال توقف (بلاک ۲A) بلااستفاده است و قطار باید قبل از ورود به این بلاک توقف نماید. در ATP های جدیدتر این مشکل با استفاده از روش Distance-to-go مرتفع شده و بلاک بعد از قطار در حال توقف نیز قابل استفاده می‌باشد. با استفاده از این روش ظرفیت خط بسته به سرعت خط و طول بلاک‌ها تا ۲۰٪ قابل افزایش می‌باشد. در این روش فاصله تا نقطه توقف، به تجهیزات ATP نصب شده در قطار منتقل می‌گردد و پردازشگر موجود در سیستم، بر اساس فاصله تا نقطه توقف، منحنی کاهش سرعت قطار را محاسبه می‌کند به‌طوری‌که نهایتاً قطار در فاصله ۲۵ متری از قطار مقابل توقف نماید. این فاصله به عنوان فاصله ایمنی (Sfety margine) جهت پوشش دادن به خطاهای احتمالی در نظر گرفته می‌شود. با استفاده از این روش منحنی ترمز قطار یک بلاک به جلوتر شیفت داده می‌شود و قطار در بلاک ۶A با سرعت عادی به حرکت خود ادامه داده و کاهش سرعت را از بلاک ۵A آغاز نموده و نهایتاً در فاصله ۲۵ متری از قطار مقابل توقف می‌نماید.



شکل ۲-۱-۲ استفاده از روش Distance-to-go در ATP ها

روش Distance-to-go هم در حالت با راننده و هم در حالت بدون راننده استفاده می‌گردد. برای جلوگیری از بروز خطر، علاوه بر منحنی ترمز که برای راننده نمایش داده می‌شود و راننده موظف به رعایت آن می‌باشد، ۲ منحنی دیگر (منحنی حالت هشدار (Warning braking curve) و منحنی حالت اضطراری (Emergency braking curve) توسط سیستم کامپیوتری موجود در قطار محاسبه می‌گردد.

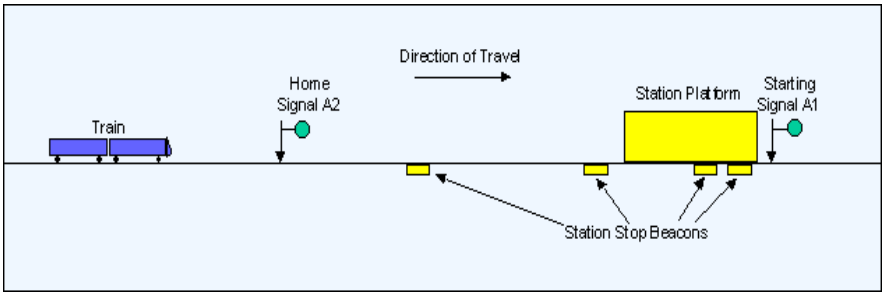
چنانچه راننده یا سیستم کنترل حرکت، به هر دلیل نتواند سرعت قطار را طبق منحنی محاسبه‌شده توسط سیستم کامپیوتری کاهش دهد و سرعت قطار به مرز منحنی اعلام هشدار برسد در این صورت هشدار صوتی یا تصویری، صادرشده و حتی می‌تواند ترمز سرویس از سوی سیستم اعمال گردد. چنانچه سرعت به محدوده مجاز برگردد، قطار به حرکت خود ادامه می‌دهد اما چنانچه سرعت به مرز منحنی ترمز اضطراری برسد، فرمان ترمز اضطراری از سوی سیستم به منظور جلوگیری از سانحه صادر خواهد گردید.



شکل ۲-۲-۳ منحنی‌های کاهش سرعت نرمال، اعلام هشدار و اضطراری در روش Distance-to-go

سیستم ATO

نقش ATO را در سیستم سیگنالینگ معرفی کنیم باید بگوییم که این سیستم جایگزین راننده در قطارهای اتوماتیک یا بدون راننده می‌باشد. از وظایف اصلی ATO می‌توان به نگه داشتن قطار در ایستگاه‌ها در مقابل سکوها و همچنین کنترل حرکت قطار در مسیر حرکت اشاره کرد. سیستم‌های ATO دارای تعدادی راهنما (ATO Beacon) در طول مسیر بوده (شکل زیر) و علاوه بر آن دارای تجهیزاتی می‌باشند که باید در داخل قطار نصب گردند. معمولاً راهنماهای نصب شده در طول مسیر شامل اطلاعات سرعت (معمولاً سرعت ثابت) در طول مسیر می‌باشند اما معمولاً آخرین راهنمای نصب شده در ایستگاه اطلاعاتی از قبیل زمان توقف در ایستگاه و حتی سرعت حرکت قطار به سمت ایستگاه بعدی را نیز شامل می‌گردد.



شکل ۲-۳-۱ راهنماهای ATO نصب شده در طول مسیر

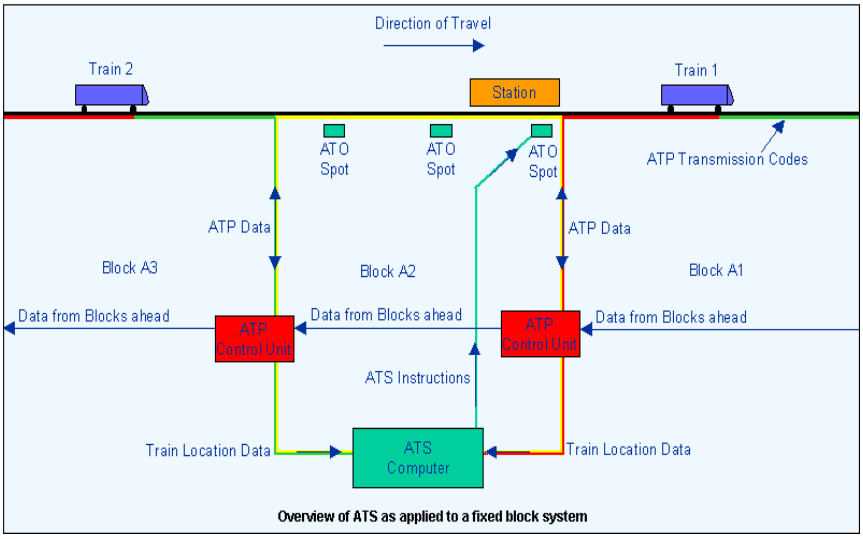
راهنماهای نصب شده در ایستگاه‌ها، اطلاعات لازم را در اختیار قطار جهت کاهش سرعت برای توقف در نقطه تعین شده قرار می‌دهد. تجهیزات On-board نصب شده در قطار این اطلاعات را پردازش کرده و با توجه به محدودیت‌های اعمال شده از سوی ATP منحنی کاهش سرعت قطار برای توقف در نقطه تعیین شده را مجدداً محاسبه نموده و در سیستم کنترل حرکت قطار اعمال می‌نماید. به وسیله ATO های امروزی قطار قادر است در نقاط تعیین شده با دقت ۱۵.۰± متر توقف نماید.

یکی دیگر از وظایف ATOها چک کردن شرایط باز و بسته شدن درب‌ها در ایستگاه‌ها و اینکه درب‌های کدام سمت قطار باید باز و بسته شوند می‌باشد. در مواقعی که ایستگاه‌ها مجهز به درب‌های محافظ (Screen door) می‌باشند، سیگنالی نیز به سیستم کنترل آنها ارسال می‌گردد.

هر چند که باز و بسته شدن درب‌ها جزء وظایف ATO شمرده می‌شود اما در برخی موارد با توجه به اینکه این موضوع یک موضوع امنیتی می‌باشد این وظیفه به ATP محول می‌گردد.

سیستم ATS

اطلاعات دریافتی به‌وسیله سیستم کنترل ATP معمولاً محدود به این است که آیا قطاری در بلاک مقابل وجود دارد یا نه یا محدودیت سرعتی که باید در بلاک اعمال گردد تا مانع از برخورد با قطار مقابل گردد چقدر است. این اطلاعات و اطلاعات مربوط به سرعت قطار به سیستم کامپیوتری ATS در اتاق کنترل ارسال می‌گردد (شکل بعد). سیستم ATS با مقایسه سرعت قطار با جدول زمان‌بندی حرکت قطار تشخیص می‌دهد که آیا حرکت قطار طبق برنامه زمان‌بندی می‌باشد یا نه. برای تنظیم زمان‌بندی حرکت قطار، ATS فرامینی را به ATO های موجود در خط ارسال می‌کند و این ATOها این اطلاعات را در اختیار تجهیزات سیگنالینگ نصب شده در قطار قرار می‌دهد و نهایتاً سرعت قطار بر اساس برنامه زمان‌بندی حرکت قطار کنترل می‌گردد.



شکل ۲-۴-۱ ساختار یک سیستم سیگنالینگ کامل شامل ATP, ATO,ATS

طراحی ارتباط بین سیستم کنترل قطار و تجهیزات On-board سیستم سیگنالینگ

تجهیزات سیگنالینگ نصب شده در قطار باید بتواند با زیرسیستم‌های مختلف قطار از قبیل Master controller، سیستم کنترل درب‌ها، سیستم‌های کنترل ترمز، سیستم‌های کنترل ترکشن موتور و ... ارتباط برقرار نماید.

زیرسیستم‌های مختلفی از قطار با تجهیزات سیگنالینگ نصب شده در آن در ارتباط می‌باشند. یکی از مهم‌ترین بخش‌ها در طراحی ناوگان، طراحی ارتباط بین تجهیزات سیگنالینگ نصب شده در قطار و زیرسیستم‌های مختلف قطار می‌باشد. طراحی این ارتباط در حالی که تأمین کننده سیستم سیگنالینگ و سیستم کنترل و مونیتورینگ قطار یکسان باشد، کار ساده‌ای می‌باشد؛ اما در حالتی که تأمین کننده سیستم سیگنالینگ و سیستم کنترل قطار متفاوت باشد، در این صورت در مرحله طراحی ناوگان باید پیش‌بینی‌های لازم در طراحی سیستم کنترل و مونیتورینگ قطار صورت گیرد تا پس از مشخص شدن سیستم سیگنالینگ، ارتباط بین سیستم کنترل و مونیتورینگ با سیستم سیگنالینگ نصب شده در قطار با کمترین هزینه و دوباره‌کاری برقرار گردد.

بدین منظور لازم است پیش از طراحی سیستم کنترل قطار، تمام ارتباطات لازم بین سیستم کنترل قطار و سیستم سیگنالینگ، مستقل از تأمین کننده سیستم سیگنالینگ شناسایی گردد. پس از شناسایی ارتباطات لازم بین سیستم کنترل و مونیتورینگ قطار و سیستم سیگنالینگ، این اطلاعات باید در سیستم کنترل و مونیتورینگ قطار (TCMS) جمع‌آوری گردیده و نهایتاً ارتباط بین سیستم سیگنالینگ با زیرسیستم‌های مختلف از طریق TCMS برقرار گردد. بدین منظور لازم است تعدادی ورودی و خروجی دیجیتال و آنالوگ در TCMS برای ارتباط با سیستم سیگنالینگ به صورت رزرو در هنگام طراحی ناوگان پیش‌بینی گردد. در این صورت می‌توان پس از مشخص شدن سیستم سیگنالینگ و نصب تجهیزات On-board، ارتباط سخت‌افزاری سیستم سیگنالینگ و TCMS را برقرار نمود و سپس تنظیمات نرم‌افزاری لازم را در نرم‌افزار TCMS انجام داد.

سیگنال‌های دیجیتال ورودی به سیستم ATC و خروجی TCMS

در طراحی سیستم کنترل قطار باید تعدادی خروجی دیجیتال در TCMS پیش‌بینی کرد. این خروجی‌ها اطلاعات زیر را در اختیار سیستم سیگنالینگ قطار قرار خواهد داد:

۱- ADC(All Door Closed) : بسته بودن همه درب‌های قطار

۲- HCR(Head car Control Relay) : نشان دهنده master بودن کابین

۳- TCR(Tail car Control Relay) : نشان دهنده Slave بودن کابین

۴- DCF(Direction Control Forward) : نشان دهنده حرکت به سمت جلو

۵- DCR(Direction Control Reverse) : نشان دهنده حرکت به سمت عقب

۶- MCC(Master Control Coasting) : نشان دهنده وضعیت master control

۷- DPB(Departure Push Button) : نشان دهنده صدور فرمان حرکت از طرف راننده

۸- EBF(Emergency Brake Feedback) : نشان دهنده اعمال سیستم ترمز اضطراری

۹- EBRF(Emergency Brake Relay Feedback) : نشان دهنده عمل کردن رله ترمز اضطراری

۱۰- FSBF(Full Service Brake Feedback) : نشان دهنده اعمال سیستم ترمز سرویس

۱۱- NRBD(Non Release Brake Detect) : عدم آزاد شدن ترمز پارک

۱۲- REB(Release Emergency Brake) : رها شدن ترمز اضطراری

۱۳- ATOF(ATO Feedback) : این سیگنال به ATC اعلام می‌کند که قطار شرایط لازم برای حرکت اتوماتیک را دارد.

با توجه به موارد فوق و رعایت اصل Redundancy در طراحی سیستم، لازم است حداقل ۱۶ خروجی دیجیتال سیستم TCMS به سیگنال‌های ورودی ATC اختصاص یابد.

سیگنال‌های دیجیتال خروجی از سیستم ATC و ورودی به TCMS

در طراحی سیستم کنترل قطار باید تعدادی ورودی دیجیتال در TCMS جهت دریافت سیگنال‌های دیجیتال خروجی از ATC پیش‌بینی کرد. این ورودی‌ها اطلاعات زیر را از سیستم سیگنالینگ قطار دریافت خواهد کرد:

۱- EB(Emergency Brake) : ارسال فرمان ترمز اضطراری از سوی ATP

۲- RDE(Right Door Enable) : سیگنال نشان دهنده اینکه درب‌های سمت راست می‌توانند باز شوند

۳- LDE(Left Door Enable) : سیگنال نشان دهنده اینکه درب‌های سمت چپ می‌توانند باز شوند

۴- FSB(Full Service Brake) : ارسال فرمان ترمز سرویس از سوی ATP

۵- PCO(Propulsion Cut Off) : ارسال فرمان قطع رانش از سوی ATC

۶- ATOB(ATO Brake Signal) : نشان دهنده اینکه set point ارسالی به صورت آنالوگ از سوی ATO مربوط به set point ترمز می‌باشد.

۷- ATOP(ATO Propulsion Signal) : نشان دهنده اینکه set point ارسالی به صورت آنالوگ از سوی ATO مربوط به set point ترکشن موتور می‌باشد.

۸- DEPL(Departure Lamp) : خروجی که از سوی ATC صادر می‌شود و با استفاده از آن لامپ DEPL مقابل راننده روشن می‌شود و بیانگر این است که راننده می‌تواند کلیدِ Departure را برای شروع حرکت فشار دهد.

۹- ATO: موقعی که ATO در حالت اتوماتیک (کنترل حرکت قطار بدون راننده) قرار می‌گیرد، این خروجی از سوی ATC صادر می‌گردد و مادامی‌که ATO در حالت اتوماتیک باشد این خروجی فعال می‌باشد.

۱۰- REBL(Release Emergency Brake Lamp): خروجی که از سوی ATC صادر می‌شود و با استفاده از آن لامپ REBL مقابل راننده روشن می‌شود و بیانگر این است که راننده می‌تواند کلیدِ آزادسازی ترمز اضطراری را فشار دهد.

با توجه به موارد فوق و به منظور در نظر گرفتن تعدادی سیگنال رزرو، بهتر است ۱۶ ورودی دیجیتال در سیستم TCMS به منظور ارتباط با سیستم سیگنالینگ در نظر گرفته شود.

سیگنال‌های آنالوگ خروجی از ATC (ورودی به TCMS)

وقتی که کنترل حرکت قطار به عهده ATC باشد، لازم است که ATC میزان سرعت و یا میزان نیروی ترمزی را به سیستم کنترل قطار اعلام نماید. سیستم‌های سیگنالینگ این موضوع را با یک سیگنال آنالوگ خروجی به عنوان set point اعلام می‌کنند؛ بنابراین لازم است که یک ورودی آنالوگ در TCMS برای دریافت سیگنال set point در نظر گرفته شود. لازم به ذکر است که چنانچه خروجی دیجیتال ATOP (set point تراکشن) فعال شده باشد، Set point ارسالی از سوی ATC به عنوان Set point سرعت در نظر گرفته خواهد شد و چنانچه خروجی دیجیتال ATOB (set point ترمز) فعال شده باشد، Set point ارسالی از سوی ATC به عنوان Set point ترمز در نظر گرفته خواهد شد. طبیعتاً سطح سیگنال ارسالی از سوی ATC به صورت سیگنال‌های استاندارد جریان یا ولتاژ خواهد بود و ورودی‌های TCMS نیز قادر به دریافت این سیگنال‌ها می‌باشند. سطح سیگنال‌ها و رابطه میزان آن با میزان set point مورد نظر بعد از مشخص شدن تأمین کننده سیستم سیگنالینگ به صورت سخت‌افزاری یا نرم‌افزاری قابل تنظیم می‌باشد.

با توجه به موارد فوق لازم است در طراحی سیستم، حداقل یک سیگنال آنالوگ ورودی در TCMS در نظر گرفته شود می‌توان به‌راحتی set point

سرعت و ترمز را از ATC دریافت نمود و آنها را در اختیار سیستم کنترل پروپالشن یا سیستم کنترل ترمز قرار داد. با توجه به اصل Redundancy بهتر است این تعداد بیشتر نیز باشد.