Håkan Sivencrona, Chalmers,
Johan Hedberg, SP
Håkan Röcklinger, Atlas Copco

Comparative Analysis of Dependability Properties of Communication Protocols in Distributed Control Systems

PALBUS Task 10.2

April 1, 2001

Abstract

The objective of this report is to introduce the field of dependability and discuss some common distributed communication protocols and their parameters for design of a safety critical system. The report contains a brief comparison and evaluation of some protocol principles and their profiles to provide fault tolerance. It furthermore discerns and addresses important properties such as deterministic and predictable behavior as well as composability.

Some typical protocols are reviewed more thoroughly in the report; these are CAN, TTP/C, Sercos and Profibus DP and Flexray and TTCAN are discussed more briefly.

Table of Contents

Preface

The influence by computers of almost every aspect of our life motivates the need for highly reliable computer systems.
Many control systems are based on a network of controllers communicating on a bus system. The objective of the PALBUS project is to evaluate and disseminate methods for design and evaluation of communication buses used in distributed control systems with requirements for safety and reliability. PALBUS is financed by NUTEK (the Swedish National Board for Industrial and Technical Development) and by the participating industries. The project is a part of AIS program for Active Industrial Co-operation.

Revision	**Date**	**Changes made**	**Distribution**
0	99-12-02	Outline	Task-internal
1	99-12-09	Draft	PALBUS partners
2	00-04-04	Final	External
3	01-04-01	Revised	External

1 Introduction

Distributed embedded computer system solutions have seen an increasing use for many applications. Areas or classes could be for example body electronics, powertrain control, infotainment systems or x-by-wire systems in cars, where x can stand for brake, steering or suspension etc. As a result of this, several communication protocols have emerged with different objectives. These objectives could be functionality, dependability, performance, space saving or cost efficiency.

Any development process of a computer-controlled system must be carefully planned and assessed before it can be released to the market. This is especially important and obvious for safety critical applications where a component or system failure could result in loss of goodwill and market shares, and also, most important, failure to meet personal safety criteria.
Assessment of a control system must be thoroughly done at several levels and through wise utilization of test methodology and techniques etc. Against this stands, for example, time-to-market for the product.

The dependability property can be prepared through wise utilization of the system and components at several areas. The basic fundament or area is the communication protocol. The protocols have emerged to allow nodes connected to communicate with each other. This communication must be able to meet timing constraints, redundancy handing and possibility to detect and handle faults as well as being able to continue operation in degraded mode.

A distributed control system could be divided into several levels such as the physical-, node-, application- and system level. Depending where on these levels, different methods and mechanisms can be applied to reach system requirements.
The communication protocol is the servant, where fundamental properties, such as bus access methods and message scheduling, task allocation as well as topologies are implemented. They all affect the dependability and the implementation of dependability increasing functions. These functions could be membership agreement and fault handling mechanisms.
Protocols must accordingly, implement and concentrate upon different techniques and levels to solve a specific task reliably.

The objective of this report is to investigate arguments for implementing dependability in the design and to compare different protocols in view from this. Furthermore it presents a number of existing protocols with respect to their ability to meet dependability demands.

2 Definitions of dependability properties of protocols

Most companies have thorough investigations before they introduce any product when it comes to reliability of the product and effects of possible faults, although there seem to be reinvention of the wheel in many cases. It is common to have different names for the same thing. What one single person calls a failure can by another person be called a fault. Even the term “fault tolerant” is used ambiguously to designate properties such as “the system has well-defined failure behavior” or “the system masks component failures”.

2.1 Dependability considerations for distributed control systems

Implementation of satisfying dependability in a system depends on what the hazard and requirement analysis has resulted in. The requirements in development to reach a certain level of dependability also relate to which kind of demands that must be fulfilled in the system. It is a question of fault detection, fault handling, fault recovery and masking redundancy used at different abstraction levels in the system.

Before describing dependability considerations more in detail, it is important to have understanding of the impairments of the dependability.

- Fault, the cause of an error
- Error, incorrect condition or state in a part of a system
- Failure, condition where the system does not completely fulfill the specification

The source of the fault is mostly unknown, the error in the system could be latent for a long time, and during some circumstances, it creates a failure in the system. It is impossible to avoid all errors but by using relevant development- and design principles it is possible in some cases to avoid that the faults that leads to failures. Many of these faults and failures are handled, although differently, by the communication protocol and provide fault-handling mechanisms.

In order to discern the dependability features of a system following attributes as described in PALBUS work package 10.1 “Definitions“. These can be used for requirements and specifications.

- Availability, which means readiness for usage
- Reliability, which means continuity of service
- Safety, which means avoidance of catastrophic consequences on the environment
- Security, which means prevention of unauthorized access and/or handling of information

It is important to quantify the different attributes of dependability in a specification. When a company is about to introduce a distributed control system, probably all these attributes must be considered, and some of them in more detail.

Next step in the process when the dependability requirements of the distributed system are specified is to decide what must be implemented i.e. fault avoidance, fault tolerance, fault forecasting mechanisms etc to fulfill dependability demands in the specification.

In the specification of the application not only the attributes should be quantified but also, the behavior when any fault, error and failure occurs should be defined. This sets certain demands with respect to dependability functions. Some faults may be considered as minor while faults that may lead to a failure of a task or failing nodes are treated as more serious.

The system performance when faults occur can be described from following viewpoints:

- The system must tolerate and still provide full service for a specified number of arbitrary faults. This could be a steer-by-wire system for example.

- The system should, for a specified number of faults not be degraded more than to a certain level and still maintain some service, the quality of service. This could for example be a flight control system. A failing node for a rudder is very critical and may result in other rudders taking care of its function.

- The system should at least acknowledge (could be solved by membership agreement) faults that may lead to failures and shut failing blocks/nodes off.

- Assume that some faults are temporary and transient and continue with the function as long as extrapolation from old values can be done satisfyingly. If not the system will be turned off.

- The system shuts down, if possible hopefully in a fail-safe state.

Depending on the application and the criticality of the task, it should be able to combine these above-mentioned scenarios. For some systems failing nodes are assumed to become fail-silent. This is an issue for deciding about used protocol.

Ways to act after serious faults are:

- Graceful degradation, mode changes etc.

- Use shadowing or redundant nodes/circuits (example brake systems).

- Reset of the system or recovering failing circuits/nodes.

Depending on features of different protocols more or less of above-mentioned points are implemented in the protocols. If the protocol does not have these mechanisms implemented they must either be implemented in a high-level protocol and/or using redundancy.
It is impossible to avoid all faults but following list consist of ways to decrease the probability that faults should occur.

- Use MTBF lists for the hardware to chose “good“ components

- Use formal methods for the software

- Investigate the interactions within the system

- Thorough tests

For applications with high requirements of dependability, it is relevant to request that all of the following demands as described above should be implemented based on the specification of the distributed control system:

- The system architecture should be fault tolerant to some specific degree.
- No single failure (electrical, mechanical or software) must jeopardize the function
- Chose software/hardware development tools on all levels of the design to decrease the probability of faults on the final system.
- Common mode failures must be avoided
- Implement intelligent and relevant fault handling methods when faults have occurred.

Following questions can be of importance to discuss for the distributed control systems, depending on in which areas they are used:

- What kind of data is to be sent on the bus? (Picture, speech-, regulation or text information)
- Does it have any functions to restore information about the past events of the system?
- Is the system built up in such way that it is possible for a node that has lost the contact with the cluster to reintegrate?
- How well implemented are the possibilities to perform service at the system?
- To what degree is possible to test the performance of the system?
- Is it possible to use the system in environment with electrical disturbances?
- Gives the system opportunities to implement information security?
- How flexible is the system in general?
- Are there connected stubs and sub systems that may affect the performance?

2.2 Levels in distributed control systems to achieve fault tolerance

In distributed control systems, fault tolerance must be achieved throughout the embedded system. Levels where this must be achieved are mentioned below. These levels are following Jalotte's model. Building blocks and fault tolerant services together define a fault tolerant distributed system.

First, we have some sort of basic level, the distributed system with the bus, called the physical level

- No single point-of-failure bus
- Redundant circuits, etc.

Second, we have, among the building blocks, node level

- Fail-stop nodes, i.e. no failing processor will participate in the communication.
- Stable Storage
- Reliable communication (Atomic broadcast is that all decided nodes should have gotten a specific message or no node shall execute). It might be enough with reliable broadcast

Other levels are application level and system level but these are not handled in this report since it is not directly protocol related.

Fault tolerant services could be:

- Consistent State recovery
- Atom actions (an action is generally aborted if a failure occurs while executing and must have consistency under node failures)
- Data resiliency (continued service under node failures).
- Process resiliency (continued service under node failures).
- Fault Tolerant Software, This means continued service under design faults.

2.3 Parameters of performance

This sub chapter includes a number of demands, which are important to take into consideration during the development phase of a distributed system with dependability demands. The demands are split up in general demands; technical demands and demands connected to dependability.

These parameters will be used below when we try to describe the different protocols in comparison to each other. Some of the parameters in the different lists are perhaps not relevant for all of the protocols.

The description of protocols in chapter 3.2 will be based on the parameters defined below:

General parameters:
- The cost to buy the system
- The complexity of the system must be such that it gives realistic development times
- Does equipment from several wendors exist?
- Industry acceptance, industry often demands that the system is well established. A new concept must be accepted by the market, which needs sales effort, education etc
- Found out which standards and directives which are relevant to use in the development phase
- User-friendly development tools
- The physical size of the hardware in each node, maybe this can put a limitation on a certain concept
- Well structured manuals of the systems performance
- Is the specification available for free
- License costs
- Users organization
- Cost for certification
- Certification needed

Technical parameters:
- Which protocol technique is used: single master, multi master, event-triggered, time triggered? And which communication technique: base band
- How large overhead does the protocol implement? For instance, the ratio between the total frame length and the data could be calculated and represented.
- What is the bandwidth of the communication bus, i.e. maximum transmission speed?
- Is it possible to implement different message length? Flexibility in message sending.
- Which kind of coding technique is used for each protocol?
- Bits per message?
- Maximum number of nodes?
- The maximum physical length of the bus?

- Open-/closed system. Is it possible to connect the protocol together with other existing protocols?
- Is the controller implemented in hardware or in software?

Dependability related parameters:

- Testability, in run time and off run time
- Does the protocol handle different modes and in such case which (how many, their purpose) and how does the mode change take place.
- Modifiability, for instance if different nodes can handle different tasks. In effect if a node stops working properly the protocol detects this and gives this task to another node in the system. The protocol can only support membership and these other functions must be taken care of at the system level.
- Does it exist methods to avoid “babbling idiot failure“? “Babbling idiot failure“ is a failure where faulty nodes occupy the bus all the time. Depending on the protocol, it is more or less possible to implement methods to avoid.
- Synchronization functionality, which is implemented in the protocols. Does it support a global time?
- Replica handling
- Message Acknowledgement
- Atomic Broadcast
- What is the size of the jitter in comparison to a specified value? The size of the jitter depends on the accuracy of the clocks and of the global synchronization of the nodes.
- How is the message protected from disturbances, for instance, by error detecting or by error correcting code?
- What is the response time of messages? Depending of application, different demands are required. In some applications a short worst case delay is preferable and in other the jitter should be minimized:

In PALBUS work package 10.5 methods for fault handling and fault detection are presented. Some faults, listed below, initiated by the communication system will have large impact on system dependability. Therefore is their failure important to analyze.

- Transient faults
- Omission/timing/crash/Byzantine faults
- Bus errors
- Data consistency errors
- Initialization and restart errors
- Configuration errors

3 Presentation of communication concepts with respect to dependability

This chapter analyzes topology, signal medium and bus access/control utilized by different protocols. It discusses these protocols, and their principle of operation. The chapter tries to point out advantages and disadvantages of some concepts related to the mentioned protocols. Special features of these ”typical” and fundamental principles are investigated.

3.1 Common topologies for distributed systems

A distributed computer system consists of a set of nodes connected to a communication means. The topology could be for example a broadcast, more or less fully connected, star-coupled, tree or a ring system.

The broadcast bus topology consists of a wire (bus) to which all nodes are connected. All connected nodes will see the same signal at the same time. If the bus should become short circuit for example, the whole bus would fail to provide any information. This topology is difficult to implement with optical fibers, although with mirrors and lenses etc, e.g. passive stars some of the broadcast bus features can be imitated. The broadcast bus medium is cheap and simple with a minimum of connections between nodes and bus. It is reliable and flexible when new nodes shall be added. The bus topology is often used in systems with double channels for redundancy purpose. A doubled bus should be drawn different path, to reach highest reliability.

A fully connected system has a topology where all nodes have wires between themselves and all other nodes. A system using this will become very complex and expensive with increased number of connected nodes. Using some mesh structure where the nodes have contact with maybe two or three other nodes could solve this. The communication delay times are in this kind of systems difficult to analyze. It is a very robust system with many redundant paths if one node or link should fail and the local system bandwidth can be increased if a large number of messages should be broadcasted.

A star-coupled system relies on a master node, in the center to which all nodes are connected. No direct coupling between other slaves/nodes exists. This feature makes the star in charge of all communication between different nodes. The star can shut down failing nodes and control the access to the bus. As in a broadcast bus all connected nodes will see the same signal at the same time, but it does not experience the same good reliability as the broadcast bus. Both electrical and optic buses are known to use this topology.

In a ring system, all nodes have two connections to other nodes. It means that there formally exists one redundant path for the signals if one connection gets broken. This feature exists only if messages can be transmitted in both directions and is not implemented in all ring-structured protocols. A ring system can feature larger system bandwidth than a broadcast bus, however, the timing analysis and scheduling to achieve this would be more complex. Furthermore, a ring is usually more reliable than a star connection. Ring systems are often used together with optic fibers.

In a tree structure system, all nodes have at least one connection with another node. These nodes will either direct incoming data up in the hierarchy or down. There is only one path between different nodes. The top node will be heavily occupied to pass messages on to other nodes, while the bottom nodes will only receive or transmit. This structure has limited use in dependable systems.

Redundant channels are often used for protocols that wish to achieve fault tolerance against more bus failures. Systems with redundant channels and nodes do not risk that the bus and nodes become single points of failures. If these double channels are combined with redundant nodes, it is possible to increase the reliability of the system.

The following paragraphs deals only with protocols for broadcast busses.

3.2 Protocols for distributed control systems

A distributed computer system consists of a set of nodes, e.g. actuators and sensors, connected to a communication means, an optical or electrical wire or even radio, e.g. bluetooth. The communication topology for this section is an electrical broadcast bus, briefly analyzed in chapter 3.1. The topology allows all connected nodes direct access to the messages transmitted on the bus.

This section reviews and compares feature of three ”common” protocols or concepts for embedded computer systems. The protocols handle the communication in very different ways and these principles could act as some kind of align for similar protocols and their features. Their special advantages, sometimes comes from how the access is controlled and when nodes are in charge (see chapter 3.2.1).

The objective of this section is to evaluate these protocol principles to make a trade off for an embedded application with dependability demands.

Important are also quality factors, such as testability and verification of protocols. These are not be handled in this section but discussed in other parts of the PALBUS report (work package 10:10, 10:11).

3.2.1 Bus access methods and message control

When distributed control systems were first introduced it was in most cases to run one single application. The protocol utilized was specified for this single application and not suitable for growing systems or other applications within the same product. Flexibility was something that was considered inappropriate for a dependable design. However, most protocols of today are feasible for a broader range of applications. It is a question of utilizing the properties wisely. It is believed that in future maybe six networks could coincide within the same vehicle. This introduces a hierarchy of protocols for example after their requirements on dependability. It is also possible to arrange these protocol types with respect to their handling of the bus access and on what base or constraints nodes connected to the bus may request to transmit. Depending on the bus access change and bus control, e.g. time constraints etc, the dependability of the whole system changes. These issues could be set up as below.

- When are messages sent via the bus?

  A transceiver connected to the broadcast bus can access the bus and send messages through two methods. On demand, asynchronous base, thus acting often events when something needs to be carried out. The transceivers that wish to send data must first “request” the bus. In a synchronous approach, acting on a periodic base the access is given in a specific time slot. This means that the communication is processed in a deterministic and pre-scheduled way even if the system does not have to solve any tasks or send any data except messages to run the system.

- Which node/nodes are controlling the communication on the bus?

  The decision when a message must and can be sent could be taken by several entities, transceivers connected to the distributed system. The system is sometimes designed so that more than one node can control the bus according to for example a medium access scheme. Some protocols have only one node for the control although a redundant node can step in, when for example the ”main” node fails.

A third type of protocol is token-based. A node that receives a token is given full authority over the bus.

These two issues are basic for the design of bus protocols and set limitations for possible applications and performance.

A node that acts on demand can receive the access by measuring the communication bus and try to send if the bus is free. If the bus is occupied or a collision between two concurrently sending nodes occurs, the node must halt its transmission and wait until the bus is idle or for some random time before it tries again. Ethernet uses this principle.

Access can also be decided by bit wise arbitration, contention based protocols. This scenario establishes a priority issue between concurrently transmitting nodes. This depends on dominant bits in the beginning of every message frame. If one node, A, sends a “1” and the bus do not respond with a “1” another higher prioritized node must have sent a “0” and therefore canceled A’s request to send. This procedure is established to decide which node that will continue to send. The other nodes have to cease their transmissions and retry later. This arbitration field can be static or dynamic and therefore be used to increase the timeliness of the system, e.g. through monotonic methods.

The system may also act on a token that is passed to give a certain node master abilities. The ‘token’ concept could be considered as a mixture between single-master and multi-mastering techniques. This technique is not investigated in this work package.

Some protocols are utilizing both principles, asynchronous and synchronous access to the bus. The scheduled access is then used for close loop regulations and more critical tasks while some time in the cycle is designed for aperiodic and low priority use.

Who to be in charge thus depends on events, time and whether the system has one or many nodes that may control the communication. For a system with one controlling node, a single master system, the question of events does not have any great impact on the communication since the master controls the whole sending.

3.2.2 Principles of operation

A single-master system, that has only one node in charge of the communication, for example MIL-STD-1553B, relies on a bus controller, BC, to handle the broadcast communication. This master node solves and administrates possible access conflicts and errors that may arise. It is noteworthy that without a redundant master node, the system has a single point of failure. A permanent failure in this node would cause the complete system to go down. Since the master node takes care of the message administration, it is easy to analyze and monitoring the communication. The overhead cost and thus the complexity for this concept would become rather high because of all commanding messages to participating nodes. The overhead calculation related to a single-master system must count for control messages with agreement messages being sent every time data is needed from for example sensors and actuators separated from the master node.

A system that wishes to accomplish higher robustness may choose an approach that consists of many nodes that may control the transmission themselves, when they need to transmit. These nodes, so to speak, have split access to the bus. This divided access requires arrangements and agreements to give a single node control over the bus.
All nodes may control the communication in a multi-master system, and therefore it is easier to design a robust system.

The overhead cost for multi-master system can be low since they act as individuals and can keep track of the net bus status without being a part of the current transmission itself. The communication must be planned better to maintain a consistent system. In a fault tolerant system no failing nodes should degrade the communication and deadlines should be held. The medium access scheme is scheduled in all sending nodes memory.

Systems that act on demand, events, e.g. with bit arbitration/contention based protocols are typically non-deterministic, especially true for low priority tasks. This way to control whom to have access to the bus can end up with a non-deterministic behavior and in the end, missed deadlines. Close loop calculations will also become harder to achieve when there are no exact time for the arrival of messages. Non-predictable features of communication protocols need serious calculation and tests to decide if deadlines can be met by the system. If a system has a behavior based on discrete events, rather than continuous states from sensors and actuators then an event-triggered system would probably be the best choice. Opposite time-triggered communication are the optimal and maybe the only solution when very few frequently transmitted messages (sampled from continuous signals) should be handled.

An issue is furthermore, how old may the data in the messages become before it is appropriately received. This argument is valid for all event-triggered protocols. For some systems re-transmissions add to this unpredictable feature.
If the system consists of some few high prioritized nodes with short deadlines but random behavior, together with many low priority nodes. Then it would be beneficial to use an event-triggered protocol since tasks could be guaranteed faster delivery time compared to a round of a periodic protocol, which is longer than the required maximum time.
One benefit from this access method is that it is easy to modify the system with new functions and tasks, also while operating.

Clocks are needed for synchronization between nodes. In addition, since there are the messages themselves that provide the synchronization and these messages event controlled it is hard to get high precision.

Adding new applications, maintaining a fault tolerant behavior, to an event-triggered system is often hard without new thorough tests. This since the concept itself was optimized for the current setup and therefore does not offer composability. An event-triggered system is optimized for certain performance and changes to this may result that tasks cannot be fulfilled within given time limits. The extensibility level is therefore good only if it is planned.
A great advantage of event-triggered systems is that it is relatively easy to have external control over the system.

A time-triggered system on the other side has access to the bus in so-called time slots when the node's "temporal" firewall is not active. This node is repeatedly transmitting in the same 'time slot' at least once every TDMA-round (Time-Division-Multiple-Access). The scheduling is static and determined before run-time if a fault tolerant behavior should be achieved. This offers effective use of the bandwidth, but this efficiency is eaten up by information for handling the communication, for example what to do in when nodes fail or does not respond in the right time slot. A deterministic behavior is however achieved through this sampling, and this offers a good base for solving tasks that need this determinism to work out.

A time-triggered system can use higher transmission frequencies compared to contention-based protocols, and therefore meet higher response requirements. The flexibility to add

new nodes to the bus is rather low and extensibility must be planned for in advance. Composability is however easier to achieve with a time-triggered protocol than with an event-triggered protocol, because subsystem behavior and properties can be maintained during system integration. If some tasks must be guaranteed shorter response time, it is possible to "sample" these nodes more often, i.e. they are executed more than one time ever TDMA-round.

Time-triggered systems are because of this very good for continuous signaling and especially when actuators and sensors have high sampling rates. The system can be designed so that data arrives to a node when it is needed. This allows solving close loop algorithms since the jitter is known, small and the sampling time is known.

A time-triggered system needs certain initializing frames and start-up algorithms to synchronize. Synchronization algorithm is also needed to maintain the synchronization but no specific synchronization messages should be necessary. This can be handled through the knowledge about what node will send when. The accuracy of such synchronization can be as good as one microsecond.
The testability of a time-triggered system is high because of the predictability.
A small comparison can be seen in figure 3.2.1.

Whom in charge?	
Multi Master	**Central Master**
All nodes can control	One node can control
Robust	Single point failure
Low overhead	Easy to analyze
When to access?	
Event triggered	**Time triggered**
Control triggered by events	Control in time-slots, easy to make fail-silent
Efficient for discrete events	Efficient for continuous signals
Not predictable, not deterministic	Predictable, replica determinism
Dynamic scheduling	Static scheduling, before run-time
Flexible, easy to add tasks	Composability, no side effects, rigid
Synchronized via messages	Inherently synchronized, periodically
Low average delays, limited predictability	Known delays, small jitters
Time adequate	Resource adequate
	Easy to test

Figure 3.2.1. Protocol principles for bus communication, a comparison.

3.2.4 Protocol comparison

A communication protocol can typically be categorized into the areas shown below.

- Single master event triggered
- Single master time triggered
- Multi master event triggered
- Multi master time triggered
- Token based

One common protocol family used by the automotive industry is CAN. Although CAN itself does not put any direct constraints on the user it can still be considered a multi-master system with event-triggered bus even if it is possible by higher level protocols to even make it time-triggered. It is a protocol mainly for soft real-time systems.

An example from the time-triggered multi mastering system is TTP/C, a new protocol that emerged after the projects X-by-Wire and TTA. The objective of this protocol is to meet requirements from safety critical hard real-time systems.

MIL-STD-1553B is an example of a very well known and wide spread protocol, used in many aircraft applications. This is typically a soft real-time system.
Because of the widely use of the protocol components are cheap and easily available for this protocol and it is a single-master, event-triggered protocol. A timely behavior can be achieved.

These concepts have demands on the communication handling and require different overhead messages to organize the data flow. Some protocols have low administrative cost when no fault-tolerant features are implemented, but these tend to demand more bandwidth for increased functionality with respect to dependability. Some protocols do not need more bandwidth while implementing dependability because dependability is built into the protocol in a higher degree than others are.
This part does only count overhead that comes from a system without redundancy and fault tolerant features. Parity checks and checksums are not considered as redundancy in this comparison.

One goal is to see how actuators and sensor's data flow varies with chosen concept without being constructed to meet safety-critical demands. It is an ongoing discussion in the whole community, which protocol to use to reach best fault-tolerance. How large is the overhead in a non-redundancy system, depending on the chosen concept? It is likely that these concepts all have great features, therefore a trade off must be done.

A single-master system is theoretically either event-triggered or time-triggered system and it is hard to distinguish whether it should be categorized as such since even a time-triggered system may act from interrupts.

When a single-master system controls the communication, the master node may start a task with a message sent to a certain node that shall receive a message, data. Then the master sends another message meant for another node that should transmit the message to the first node. The information in these two first messages contains the address information and the kind of message that is to be sent, such as memory locations. The commands are the same if you have an event-triggered master-slave system or for a time triggered. A variation of this could be when a slave (remote terminal) is polled to send message for use in the master only and there are also other methods for controlling the communication.

A command word (MIL-STD-1553B) sent by the bus controller consists of the address of the remote terminal. The next field is if the node is to transmit or receive, then the sub-address to decide what data is to be sent from what position in the node, and then a field for the quantity that shall be sent. The last bit is a parity check over the preceding sixteen bits (odd parity). The master starts to commend the receiving node and then the transmitting node. After this, the communication may start between the transmitting and receiving nodes after an inter-gap time. This data is often formed as 16 bits (one word) and then a parity check, no checksums. Several data word frames may be sent between the nodes.
Furthermore, there is also a status word frame sent when the command word says so. This is done to be able to decide whether a node has received the data or not and if the communication has been successful.

In a multi-master time-triggered (TTP/C) system, the communication differs from the single master communication. A normal data frame (N-frame) consists of four fields. First the start bit and then one four-bit header and a data field between 0 and 128 bits, typically, and then finally a CRC field of 16 bits. In the CRC field a controller state (C-state) is calculated as well, to keep consistency in the communication. The system must have these special C-states to make re-integrating nodes consistent with the rest of the nodes. They contain information about the global time, membership vector and information about messages that shall be sent on the bus. Other messages are initialization frames, I-frames that contain the C-state and these must be sent periodically after initialization to support re-integration of temporary failing nodes.

In a multi-mastered, event-triggered system (CAN), the communication frame contains four semi-frames, namely one data-frame, one remote-frame, one error-frame and one overload-frame. The communication frame does more specifically consist of seven fields. The start of frame bit, an 11-bit arbitration field for message identification followed with one bit remote-transmission-request (RTR). A 6-bit control field and then a data field follow. The data field consists of between 0 and 64 bits. A 16 bits CRC field follows up these fields and finally a 2 bit acknowledge field and a 7 bit end-of-frame field. Since the nodes see the same bits at the same time, several smart functions can be added, such as deadline monotonic and earliest deadline monotonic. This results in that the priority becomes dynamic. Re-transmissions due to priority conflicts will add to the overhead.

For a certain data, sent between two nodes, the overhead cost will depend on the protocol concept itself and not only due to the fundamental principles they work after. It is obvious that different protocols produce different frames but this comparison should be able to hold some valuable measure of what concept to chose. A comparison feels valid since MIL-STD-1553B, CAN, and TTP/C are all real time systems and should be able to compete in the same application field.

	Single-master	**Multi-master**
Event-triggered	Soft real-time Not deterministic None stop systems High bandwidth demands ***Mil-Std-1553B***	Soft real-time Not deterministic None stop systems Low bandwidth for discrete messages ***CAN***
Time-triggered	Safety critical Fail silence Hard real-time Predictable High bandwidth demands Easy to analyse ***OBDH***	Safety critical Fail silence Hard real-time Predictable Low bandwidth for continuos signals Cost effective ***TTP/C***

Figure 3.2.2 Characteristic of each generic protocol

In figure 3.2.2 the characteristics of each generic protocol group are listed. One typical example of standard protocol of each group is mentioned.

The discussion above illustrates how complex it is to trade-off different protocol principles for communication despite restricting to embedded applications, lots of compromises has to be done. Following basic rough rule can be applied:

- Time-triggered communication is useful for small systems with a few numbers of continuous signals (less than 200), especially when actuators and sensors have high sampling rates and when high system dependability is demanded. In a system based on time-triggered communication closed loop algorithms over several nodes can be optimized, because when sensing and activating are time deterministic with negligible jitter the delay times known and can be minimized.
- Event-triggered communication is useful for systems with large numbers of discrete signals randomly transmitting messages. This type of communication is best in non-safety-critical applications with soft real-time demands.

The application and the system design set different requirements for the protocol. In accordance with these, the cost for operating and administrating the communication will vary.

An important issue is how the acknowledgement mode controls should be organized. This varies depending on how the bus access is handled (see chapter 3.2.1). Other interesting issues for investigation and also included in this work package is, investigation of which concept that work most efficiently with respect to overhead, e.g. message overhead. Features that will increase this administration cost are parity checks and checksum as well as control messages to accomplish the transmission. Protocols may also work differently with respect to the overhead if the system mainly consists of active nodes that both sends and receives or if the system mainly includes passive nodes.

To be able to better appraise the overhead of the communication a small test case is investigated.
Lets assume a system where some data that needs to be sent, can be handled in one to two frames for following protocols TTP/C, CAN and MIL-STD-1553B. The data message is less or equal to 32 bits. How do these three concepts compete? This part investigates the overhead connected to this administration.
The frames from these three different concepts are shown in figure 3.2.3.

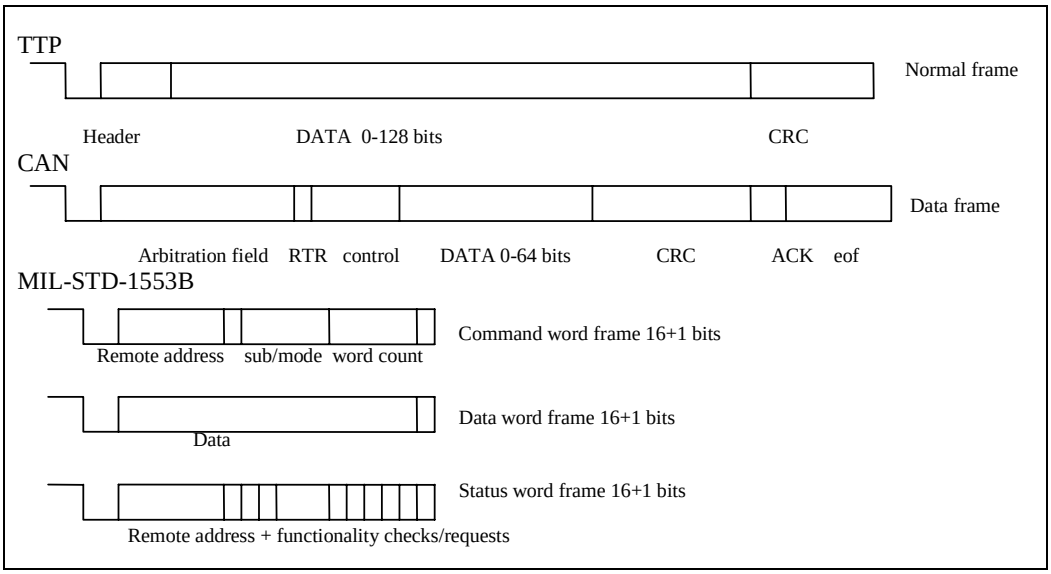


Figure 3.2.3 Typical appearance of the data-frames sent by TTP/C, CAN and MIL-STD-1553B

MIL-1553B then needs 3 overhead messages (2 command messages and 1 status message) and two data messages to fulfill and control the transmission. That would give:

32 (data) bits + 3 x 16 (control) bits +1 x 5 (parity) bits = **85** bits for a single-master system.

TTP/C uses N-frames and I-frames and depending on the size of the system the I-frame will affect the overhead lesser. Only N-frames communication gives:

4 (header) bits + 16 (CRC) bits + 32 (data) = **52** bits for every message + some I-frames.

CAN may also need to use re-transmissions but let's assume that one message is enough:

18 bits (arbitration and control) + 32 (data) bits + 16 (CRC) 9 (ack+eof) bits = **75** bits.

Judging from this, it would be best with respect to the overhead cost to choose a time-triggered concept. There are however other reasons to take under consideration. These are not investigated in this chapter.

3.3 Information of existing protocols for embedded distributed computer systems

This chapter sub chapter is a description and presentation of existing protocols outgoing from a number of parameters which are important in a distributed system with dependability demands as seen by the industry.

Each protocol description consists of three main parts, namely:

- General parameters including principle of operation
- Technical parameters
- Dependability related parameters

Technical- and dependability related parameters could be divided in the following sections depending on at what level the parameters are implemented:

- Physical transmission medium
- Controller level
- Software protocol
- Software profile (application layer)

With the definition protocol, we mean the following:

The structure of the messages sent on the bus and how they are handled in each node includes many steps, which will not be described in detail. Protocols are often divided into high-level protocols and low-level protocols. The low level protocols include mechanisms to take care of the bus near communication, while the high-level protocols communicate directly against the application software running in each node. This gives the application support, for instance, access to a global system clock, status of other nodes and so on.

In this presentation of “state of the art“ protocols the most important aspect is to describe how dependability is built into them but of course this presentation also include general features about the protocols to introduce the reader.

Following protocols will be described:

- TTP/C
- CAN
- CAN Kingdom
- CAN Open
- TTC
- SERCOS
- PROFIBUS

3.3.1 Time triggered, multi master protocol, TTP/C

Information about the TTP/C protocol is available at the following Internet address: <www.tttech.com>

Principle of operation:

TTP/C is a time-triggered protocol, which is supposed to be used in distributed systems with high demands on dependability. The quality that TTP/C is time triggered means that each node belonging to the distributed system has a specific timeslot where it is allowed to send information out on the bus. The rest of the time that specific node is just allowed to listen to and collect information from the other nodes in the system.

Following picture gives a description of how the time is shared between different modules.

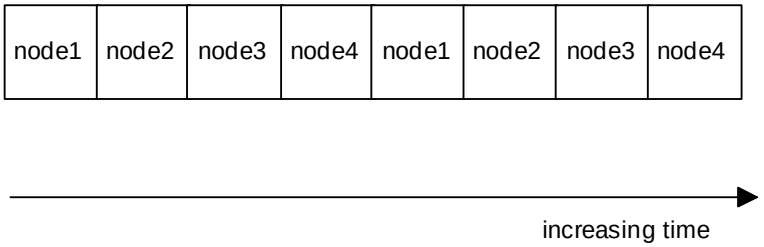


Figure 3.3.1. How time could be shared in a time triggered system.

The system in the picture consists of four different nodes. This pattern repeats periodically in the system. This is a basic way of scheduling between the nodes. It is also possible to have individual length of the sending slots for the different nodes and possible for nodes to have more than one sending slot before the communication round is repeated. This type of scheduling can be used when some nodes need more frequent access to the bus, for instance when nodes belonging to a distributed control loop requires shorter response times.

How time is shared between different nodes must be decided during design phase.

The only interface between the communication controller, which handles the interface against the bus, and the microprocessor, where the application program is running, is a dual ported random access memory called CNI (Communication Network Interface).

When the application wants to send something out on the bus, it places the information in CNI at the right point of time and after that the communication controller takes over the message handling. Also when a message is received, it is placed in a specific position in CNI and when time has reached a certain point the application program can get a correct value in this position.

CNI also contains control and status positions. In the control positions the application program can control the action of the communication controller and in the status positions the application program can get information about status of the cluster and its own communication controller.

TTP/C has been developed for a long time but it is just recently that semiconductor industry has started to develop this system on chip. The reason that this protocol is new makes it difficult to say something about some of following general parameters:

General parameters

- The cost to buy TTP/C, is unknown for us for the moment
- Complexity to introduce the system is unknown for the authors
- Industry acceptance is difficult to say something about because this protocol is recently introduced but it is clear that the industry has realized that TTP/C can offer important features regarding increased dependability
- Relevant standards and directives, see SAE standards for automotive industry
- Development tools, see website
- Physical size of the node. Latest status about this is also available at *TTT*ech: s website
- Well-structured manuals of the systems performance.
- The technical specification of TTP/C is available at <www.tttech.com> free of charge
- License costs is unknown
- Users organization is unknown
- Cost for certification is unknown
- Certification needed is unknown

Technical parameters:

- TTP/C has got specification for two separate channels to send information out on the duplicated bus
- TTP/C is a multi mater, time-triggered protocol
- In TTP/C the physical transmission medium is a twisted pair wire although the protocol itself does not limit the mediums.
- The hamming distance in TTP/C is 6 or 5 depending on the length of the frame
- TTP/C working with 3 different baud rates: 500 kbps, 1 Mbps, 2 Mbps
- TTP/C can work with different message lengths. When the scheduling is performed, it is up to the designer to decide how the total time should be divided between different nodes. If some nodes need to send more information than the other nodes, these nodes are given more time slots in the cluster cycle. A cluster cycle is defined as the time it takes for the frame pattern to repeat itself
- Bit encoding/decoding is based on modified frequency modulation code (MFM)
- TTP/C needs at least 52 bits for a 32 bits normal data message. However, to work properly TTP/C also has to send initialization frames because these include information for a node that is about to reintegrate in a cluster. Depending on application the schedule designer builds up a specific pattern of normal frames and initialization frames
- The maximum number of nodes that can be connected in the cluster is 64 nodes for the moment
- The maximum physical length of the bus is depending on what communication medium that is used in each application
- The controller in TTP/C is implemented in hardware
- The bus communication interface is controlled by a separate microprocessor, which communicates with the TTPC-controller through the CNI.

Dependability related parameters:

- The time triggered structure together with the time synchronization described above gives low jitter in the messages sent on the communication bus. The maximum jitter depends on the size of the system.
- The message is protected from disturbances by means of cyclic redundancy checksum
- When a message should be sent between two nodes following delays are specified:
- Maximum message delay
- Minimum message delay
- Mean delay
- These delays depend on specifications, bit rates etc.
- Modifiability is possible in TTP/C. for instance one node can work as an extra node. If any of the other nodes gets wrong this node can take over that nodes application
- Testability is high. Modifiability should be taken care of during the design task.
- In TTP/C, a maximum of 30 different modes can be defined. A few bits in the frame define in which mode each node is working in. The technical solution to work with different modes gives flexibility in the message transfer without adding more than a few numbers of bits. For instance following modes could be available: start-up mode, download mode, normal mode and so on. However the fault-tolerance will not be as high with allowed mode changes since this adds an extra parameter that may fail
- "Babbling idiot" means that an erroneous node occupies the bus constantly. This is a critical failure in a distributed system with a common bus. In TTP/C this problem is handled by means of a bus guardian, which is a separate hardware mechanism in each node that gives access to the communication bus just in its own sending slot. This hardware mechanism is near the bus.
- The synchronization for each node is performed in hardware in the controller. In a TTP/C system, synchronization between different nodes is done, by comparing the internal clocks in each node. Because of the time triggered architecture each node knows when it should arrive messages from other nodes and by calculate this difference between actual- and expected time for each message from the other nodes the actual node can get correct its internal clock. This gives each node in the distributed system access to the global clock without any need to implement clock information in the message frame
- The only interface between "host" and communication controller is the Communication Network Interface, which is realized as a dual ported random access memory. This gives the high level programmer a well-defined interface against the bus. The dependability is also increased when the communication handling an the normal node program are separated

3.3.2 Event-triggered multi master protocol, CAN

Information about the CAN protocol is available at the following Internet address, <www.kvaser.se> this site has several good links concerning CAN.

The structure of this section is divided into following sections. Low level CAN and higher level CAN.

- CAN's lower level, transmission medium and controller level with its parameters, general, technical and general.
- CAN Kingdom, CAN Open

Principle of operation:

The information given under this section is valid for all CAN based protocols.

ISO 11898, CAN, standardize how the bus-near parts in the system will be implemented. General for all protocols that are CAN based is that the lowest layers in the protocols are following the ISO 11898 standard: “Road vehicles-Interchange of digital information-Controller area network (CAN) for high-speed communication.

CAN is an event-triggered communication protocol. This means that when a node got something to send it tries to occupy the bus. To avoid collisions on the bus each message got a specific priority dependent code of how important that messages are. The access to the bus is regulated by a mechanism called bit wise arbitration. When a node wants to send something out on the communication bus it first listen on the bus in case it should be occupied by sending from another node. If the bus is idle the node start to send out its bit pattern. In addition, other nodes can have started to send out messages at the same time. The node or better frame, that has the lowest binary code in the identifier will “win” the arbitration and get access to the bus. The nodes that lose this arbitration will try again to get access to the bus by performing a retransmission.

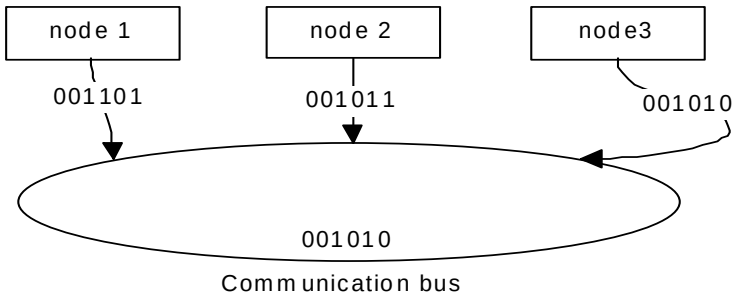


Figure 3.3.2. Bit wise arbitration mechanism.

The figure above describes a situation where three nodes try to send out their messages at the bus but at the same time but because of the bit wise arbitration is it just the message of node 3 that will be sent out on the bus. The other nodes must try to retransmit to get access to the communication bus.

A useful property while implementing priority is that this bit pattern also works as an identifier so that the other nodes in the cluster knows which messages on the bus that are relevant to extract. A disadvantage could be that nodes with low priority could be delayed because of that other nodes with higher priority wants to send its information out on the bus. This is normally not a problem because during system design each frame is given a specific priority outgoing from that node’s task at the moment in the distributed system.

The ISO 11898 standard is relevant for Controller Area Networks working at rates above 125 kbps up to 1 Mbps.

General for all CAN based protocols is that they have the structure for the physical layer and the data link layer as described in ISO 11898. The data link layer consists of the sub layers Logical Link Control (LLC) and Medium Access Control (MAC).

The physical layer consists of the functionality for bit encoding/decoding, synchronization and mechanisms to send the signals out on the bus. This functionality is split up in three different sub layers at the physical layer.
The physical layer including its sub layers will not be described in detail because this is not the point in the system where the “safety mechanisms“ are implemented. Encoding/decoding and synchronization also are important aspects of how the communication between different nodes is implemented

The LLC layer can generate two types of frames, namely LLC data frame and LLC remote frame. LLC data frame is the normal frame that is sent between a transmitter and a receiver. This frame consists of three parts:

- Identifier field, consists of 11 bits
- DLC field consists of 4 bits and informs about the length of the data message. Depending on how these bits are combined the following data field can vary between 0 and 8 byte
- LLC data field, this field can vary between 0 and 8 byte depending on the value in the DLC field

The other type of frame is LLC remote frame, which just consists of identifier field and DLC field. LLC remote frame is used when a node wants a remote node to send out information. The possibility to be able to send a LLC remote frame is of importance when a specific node needs fast information from another node in the cluster.

The Medium Access Control (MAC) sub layer is placed between the LLC sub layer and the physical layer.

In the MAC sub layer, the safety mechanism error detection is introduced in CAN. Of course other things are performed in this layer but the safety related function is the error detection mechanism. Outgoing from the data message and the CRC bit pattern this sub layer calculate if the transmission of the data frame has been correct or not.

The task for the MAC layer in the up direction is to find faults in the received frame and if it finds faults to send this information together with the data to the LLC layer. In the opposite direction the task is to calculated the CRC pattern outgoing from the data message and adding the other bits that most be included in a complete CAN frame. The MAC layer also communicates with the physical layer to see how the communication works.

In CAN-systems four different kinds of frames can be sent on the bus:

- Data frame, the normal messages that are sent between the different nodes in the distributed communication system
- Remote frame, used when a specific node needs to have data information from another node very quickly
- Overload frame, this frame is sent when the a node is internally overloaded or upon certain error conditions
- Error frame, is sent when errors are detected in different nodes

As described above, for all CAN based systems the layers LLC and MAC are implemented in accordance to the ISO 11898 standard. These are the two layers nearest to the bus and their functions are to assure that correct messages are received and sent. To manage this, a number of different methods as described below are implemented.

Common for all CAN-related protocols are following functions, which should be implemented to accomplish demands specified in ISO 11898:

General parameters:
- The cost for a CAN system with development tools is depending on the application
- It is a well-known concept with many users and good support.
- Very high industry acceptance, widespread use.
- There are standards and directives concerning CAN
- There are many users organizations
- Low-level specifications cannot be found free of charge. Certain high-level specifications could be found at different developers, free of charge.
- Cost for certification is unknown

Technical parameters:
- Monitoring, is the digital signal that should be sent out on the bus equal with the signal that actually exist on the bus
- CAN is a multi-master event triggered distributed communication system
- CAN use stuff-bits when more than five bits with the same value is sent. For instance after five “0“ the controller puts in a “1“ in the message. The stuff bit is used to be able to maintain synchronization when messages are sent on the bus. Because if a message should contain of a very long sequence of bits with the same value should this be realized as a constant analogue signal on the bus without any edge. This stuff bit also implement fault detecting because when another nodes controller receive a message with more than five bits with the same value this is an indication that the message has been disturbed during transmission.
- Each node has the possibility to retransmit a frame if it has lost arbitration, or if disturbances on the bus have changed the frame. This degrades the dependability.
- CAN usually uses a twisted pair of electrical conductor as communication medium
- When electrical conductors are used in CAN the coding technique used is Non Return to Zero.
- The physical voltage level is constant over each bits transmitting time in this coding technique and the voltage level varies between high- and low level depending on which digital information that should be sent.
- In CAN, the hamming distance is 6.
- In CAN is the ratio between total frame length and data message 75/32 when the data is 32 bits.
- CAN contains 75 bits per message (if data word is 32 bits) but the data may be as much as 128 bits
- Messages in CAN could be sent with at least the following baud rates (specified) 125, 250 and 500 kbps.
- It is possible to implement different message length. The message length is defined in the DLC field in the CAN frame
- In CAN, the access to the bus is decided by means of arbitration. The length of the identifier in the frame decides the maximum number of nodes. The identifier consists of 11bits. This should give 2048 possible identifiers but some combinations of bits are not allowed to be used as identifier. Outgoing from this the maximum number of identifiers are 2032. This number of identifiers applies for CAN 2.0 A. A new version of CAN called 2.0 B is implemented. This version consists of 29 bits used for identification and this gives theoretically the possibility to connect 536 870 912

identifiers. Demands on an extended identifier frame depends on, if the user wants to have the possibility to handle much more signals in the distributed system.
- The maximum physical length of the bus is depends on the specific baud rate
- Typically around 1 Mbps for a 40 meter long cable
- The maximum number of nodes are 32 in both CAN 2.0 A and CAN 2.0 B
- In CAN the controller is implemented in hardware

Dependability parameters:
- To improve the predictability of a CAN system and therefore the dependability it is possible to use the bit arbitration field. Methods that can be used are DM, deadline monotonic, EM, Earliest deadline monotonic and RM, rate monotonic. This means that priority is due to this dynamic.
- All frames are equipped with a 15-bit cyclic redundancy check
- When a complete message frame is received in a node, hardware mechanism controls that the frame format is an accepted format. If an error occurs this will be flagged and the frame will be retransmitted
- Acknowledgment, when a frame is received correct from another node the receiver change the sign on the acknowledgement bit in the sent message and this is a indication for the sender that the message has been received correctly.
- When a specific node detects that it is defect it has got the possibility to switch off itself
- The testability of a CAN system is complex because of the difficulty to model the bus traffic from different nodes with realistic load.
- Following message delays are relevant in CAN: Maximum delay: Minimum delay: Mean delay: These depend on the system size, etc

Mechanisms to avoid “babbling idiot“, e.g. a failing uncontrolled node, are difficult to implement in CAN because it is not possible to know when each node wants to have access to the communication bus.

It is just the physical transmission medium and the controller part that is common for all protocols based on the lower level CAN specification. The other technique description which include software protocol and software profile must be described outgoing from each unique protocol structure. For these higher-level CAN protocols it will also be described in which kind of applications they are used.

The quality that only the bus related functionality is strictly defined in IS0 11898 makes CAN very flexible and outgoing from these qualities a number of different higher level protocols have been defined which each are designed in such way that it is specialized to a specific application area.

CAN Kingdom

General:

The specification of CAN Kingdom is available at <www.kvaser.se>

CAN Kingdom is a higher level protocol that implements many of the functions that are required to be implemented in a distributed system with dependability demands. The reason why not all needed features are implemented is that then the protocol should be specialized for just one or a few applications.

The CAN Kingdom solution gives the possibility to use the protocol in a wide range of applications. Then for each application it is up to the designer to complete the high-level protocol functionality in such a way that it gives optimized performance of the task that should be performed in each specific case.

One of the most important features about CAN Kingdom is that it is built on a principle called The Modules is to serve the Network (MSN). This means that a new node can get information from the bus as how to integrate into the cluster to be able to start working properly to fulfill the expected task. The opposite situation is that a new node that should integrate to the cluster must a priori be programmed so it knows as how to perform its specific task.

In CAN Kingdom the system is described as a kingdom with a capital and a number of cities and these phrases are used as a starting point while describing the CAN Kingdom protocol. This metaphor is used to make the complex communication system easier to understand for people without detailed knowledge about distributed systems.

Below some features of CAN Kingdom is mentioned:

- CAN Kingdom is specifically developed to be used for machine control systems
- CAN Kingdom implements the important feature that the maximum latency of any message can be predicted
- The specification of this higher level protocol is based on the ISO 11898 specification
- When a node has managed to integrate to the cluster and be initiated, a set up phase is performed in the node…
- It is up to the system designer to decide the priority between different messages
- Extended CAN identifiers can be used

CAN open

CANopen is used in industrial automation to transfer data for control of I/O units, motor controllers, HMI, actors and sensors. CANopen masters are available on the market both as PLC units and PC insert cards.

CANopen supports the transmission of synchronous and asynchronous messages. The synchronous transmission of message is supported by pre-defined communication objects (sync message, time stamp message). Synchronous message are transmitted with respect to pre-defined synchronization message, asynchronous message may be transmitted at any time.

Four types of message (object) are available:
- Service Data Messages: SDO (Service Data Object), point-to-point data exchange for configuration and monitoring.
- Process Data Messages: PDO (Process Data Object), used for module-to-module(s) real time data exchange.
- Pre-defined Messages: Synchronization-, Time Stamp-, And Emergency Messages.
- Administrative Messages: Layer Management, Network Management and Identifier Distribution Message.

CAN Open provides:
- Set-up with minimal configuration effort
- Easy access to all device parameters

- Device synchronization
- Cyclic and event driven process data transmission
- Simultaneous readied of inputs
- Simultaneous setting of outputs

Synchronization could be done with a Synchronization message sent with a cyclical period.

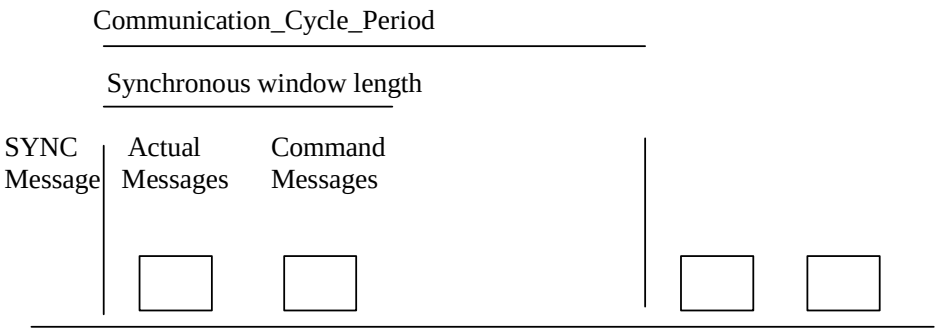


Figure 3.3.3 Message synchronization

In figure 3.3.4 is the bus configuration presented.

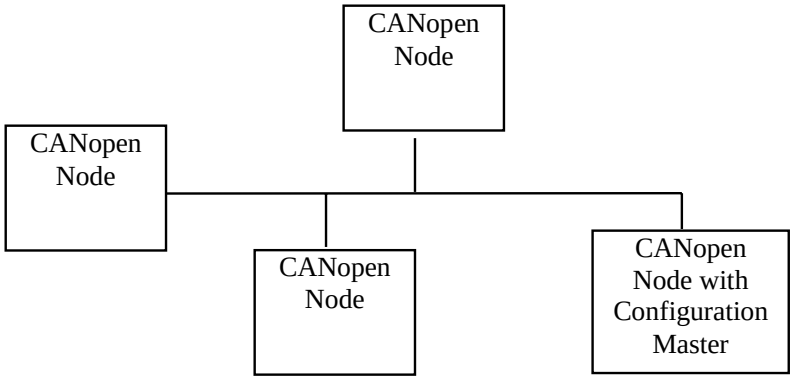


Figure 3.3.4. Bus configuration of CAN open.

Commercial issues

Cost of controller	CAN controller	10 USD
Cost physical interface	CAN driver	3 USD
Certification needed	no	
User's organization	Yes, CiA CAN in Automation e. V	
More information at	www .CANopen.com	

Technical description

Physical transmission medium

See CAN

Standard	ISO 11898
Transmission media	CAN driver, differential 2-wire related to GND potential
Transmission speed	125kbit/s to 1Mbit/s
Connector	type and pinning in according to CiA standard DS 102

Controller level

Standard	ISO 11898
Type	Module to module star network with or without master. Master only needed for configuration of the system.

Software protocol

Standard	CiA Draft Standard 301
Distance between nodes	maximum 25m at 1Mbit/s, 100m at 500kbit/s up to 500m at 125bit/s
Number of nodes	maximum 128
Information in a message	8 byte with PDO (Process Data Object) 5 byte with SDO (Service Data Object)
Message synchronization	Yes
Cycle jitter	--
Cycle time (ms)	Depending on number of nodes and transmission speed, example 10ms for 5 nodes at 1 Mbps.
Operation modes	Synchronous Cyclic Synchronous Event Driven Asynchronous Event Driven Asynchronous Remotely requested

Software profile

CANopen has Device Profile for some applications like:

- CANopen Device Profile Motion Control and Drives — Standard CiA WD-402
- CANopen Device Profile I/O modules — Standard CiA DSP-401

In the Device Profile will it be defined how the data that is sent over the bus shall be handle and used by the products regarding function, safety and real-time behavior. It also defines resolution and scaling for parameters sent over the bus.

Dependability description

Security and protection mechanisms

The CANopen protocol has mechanisms like Lifeguard (watchdog); error register and Emergency messages that could be used for building security and protection mechanisms. But the security and protection mechanisms have to be handle in the software implementation on the module using CANopen.

TTCAN-Time Triggered Can

This is an expansion of the first version of ISO11898. This new version contains a time triggered communication option. TTC enables the use of one or more prepared time masters (8 max) that may hold the time synchronization. A communication cycle consists of both synchronized time slots and arbitration slots.

The hardware needed to establish Time Triggered CAN (TTC) is included between Logical Link Control (LLC) and Medium Access Control (MAC).

To manage the synchronization in a distributed system all nodes in the cluster must have a common reference point. Possibly reference points, which are issued, are:

- Start Of Frame (SOF) bit
- Sample point of last bit of End Of Frame (EOF)

When time has reached one of the points above cyclic up counters are started in each node that supports TTC option. The length of the counter must at least be of length 16 bits.

It must be possible in each node to program the CPU in such way that when the counters has reached a certain value this could start an event in the node, in effect a message is sent out on the bus. By choosing different counter values in the different nodes it is possible to build up a time-triggered structure. If a counter with 16 bit is used it is possible to trig events at following counter values (0 to 65535). The time length between each counter value depends on the frequency at the internal or external clock connected to the counter.

Retransmission mechanisms must be disabled when the TTC option is used because a retransmission should occupy the next nodes sending slot and therefore destroy the time triggered structure

3.3.3 Central-master time-triggered protocol, Profibus DP

General description

Profibus DP is used in industrial automation to transfer data for control of I/O units, motor controllers, HMI, actors and sensors
Several different Profibus DP masters are available on the market both as PLC units and PC insert cards

For the security of perfect functionality of different appliances from different manufactures in an open system, independent test laboratories have been set up for testing the appliance in respect of the standard conformity.

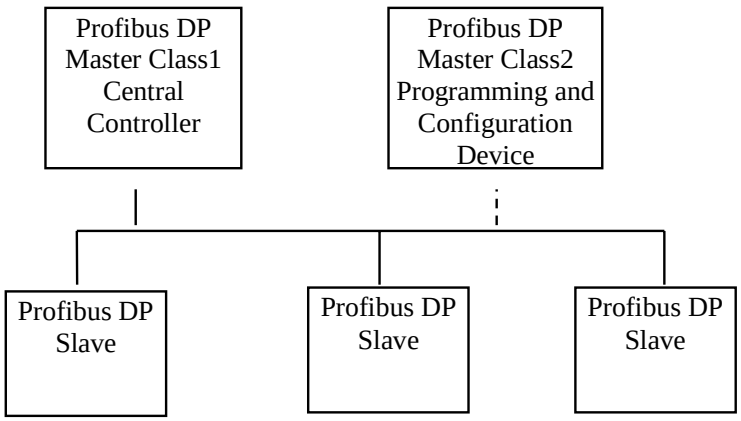
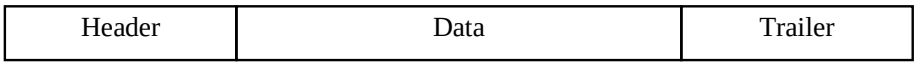


Figure 3.3.5 Bus configuration of Profibus

Frame form (message)



Commercial issues

Cost of controller	Profibus DP controller 20-30 USD
Cost physical interface	RS-485 3 USD Fiber optical 30 USD
Cost for certification	2000-4000 USD
Certification needed	yes, to be allowed to use the Profibus label
Product ID number	yes, all certified products get an ID number. Needed to do the bus configuration by the master.
User's organization	Yes, Profibus International (PI) and local Profibus National Organizations (PNO)
More information at	www.profibus.com

Technical description

Physical transmission medium

Standard	EN50170 (DIN19245 part 1)
Transmission media	RS485 twisted pair shielded or unshielded, IEC1158-2 or fiber optic
Transmission speed	9.6kbit/s to 12Mbit/s
Distance between nodes	maximum 100m at 12Mbit/s, up to 1200m at 93,75bit/s extendible with repeaters to approx. 10km
Number of nodes	maximum 32 without repeaters and 126 with repeaters
Connector	9-Pin D-sub Connector for RS485

Controller level

Standard	EN50170 (DIN19245 part 1)
Type	Master/Slave star network with Single Master or Multi Master

Communication	Peer-to-Peer or Multicast (synchronization)
Message length	--
Information in a message	max 64 byte, configurable
Message synchronization	Yes (new function)
Cycle jitter	< 1us
Coding	--
Cycle time (ms)	Depending on number of nodes and transmission speed, example 1ms for 32 nodes at 12 Mbps.

Software protocol

Standard	EN50170 (DIN19245 part 3)
Operation modes	Operate: cyclic transfer of input and outputs Clear: input are read and output cleared Stop: only Master-Master function is possible
Functionality	Cyclic user transfer between DP-Master(s) and DP-Slave(s) Activation or deactivation of individual DP-Slaves Checking of the configuration of the DP-Slave Powerful diagnosis mechanisms, 3 hierarchical levels of the diagnosis message Synchronization of inputs and /or outputs Address assignment for DP-Slaves over the bus Configuration of the DP-Master over the bus max 246 bytes input and output per DP-Slave, typical 32 bytes Connection and disconnection of node without affection of other nodes
Device Types	DP-Master Class 2 (DPM2) e.g. programming/ configuration device DP-Master Class 1 (DPM1) e.g. central controller like PLC, PC, CNC… DP-Slave e.g. Input / Output device digital or analog I/O, drives, sensors and actors…

Software profile

For some applications has the content in the message been defined in a product specific Profile.

- NC/RC controllers DP profile for numerical and robot controller.
- Encoders DP profile for rotary, angel and linear encoders.
- Variable Speed Drives DP profile for electrical drive technique.
- HMI Devices DP profile for Human Machine Interface devices
- Failsafe with Profibus DP profile for safety applications

In the Profile will it be defined how the data that is sent over the bus shall be handle and used by the products. It also defines resolution and scaling for parameters sent over the bus.

Dependability description

Security and protection mechanisms

- All messages are transmitted with Hamming Distance HD=4
- Watch-dog timer at the DP-slaves
- Access protection for input/outputs at DP-Salves
- Data transfer monitoring with configurable timer interval

Profibus-DP uses control mechanisms at the DP-Master and at the PD-Slaves. They are implemented as Watchdog timers. The control intervals are defined during the configuration of the system.

At the DP-Master
The DP-Master monitors the user data transfer of the DP-Slaves with the timer named Data_Control_Timer. A separate control timer is used for each slave. The time monitoring is tripped when correct data transmission does not occur within the monitoring interval. The user is informed when this happens. If the automatic error reaction has been enabled, the DP-Master exits it OPERATE state, switch the output of all assigned slaves to fail-safe status and change to the CLEAR status.

At the DP-Slave
The slave uses the watchdog control to detect failures of master or transmission line. If no data communication with the master occurs within the watchdog control interval, the slave automatically switches its outputs to the fail-safe status.
In addition, access protection is required for the input and output of the slaves operating in multi-master system. This ensures that only the authorized master has direct access. For all other masters the slaves offer an image of their input and outputs, which can be read from any master even without access rights.

3.3.4 Sercos

General description

SERCOS is an international standardized interface for digital drives in numerical controlled machines but it could also be used for more general control of devices in industrial automation where synchronized transfer data is needed.
Several different SERCOS master controllers are available on the market both as PLC units and PC insert cards.

SERCOS interface
- International standardized interface for digital drives in numerical machines.
- Is neither a purchasable product, nor a construction regulation for controls or drives, but a standard for transmitting medium, for topology, connection technology, telegram contents, data and scaling methods.
- Transmit command- and actual values with extremely short times and guarantees a microsecond exact synchronization for precise and coordinated moves with as many drives as needed.
- Additional contains a non-cyclic data transmission. This enables the display and input of all control terminals, as well as storage and loading of drive parameters.
- Ensure the communication of controls and drives of different manufactures by standardization of all data, commands and reactions exchanged between drives and controls in numerically controlled machines.
- For the security of perfect functionality of different appliances from different manufactures in an open system, independent test laboratories have been set up for testing the appliance in respect of the standard conformity.

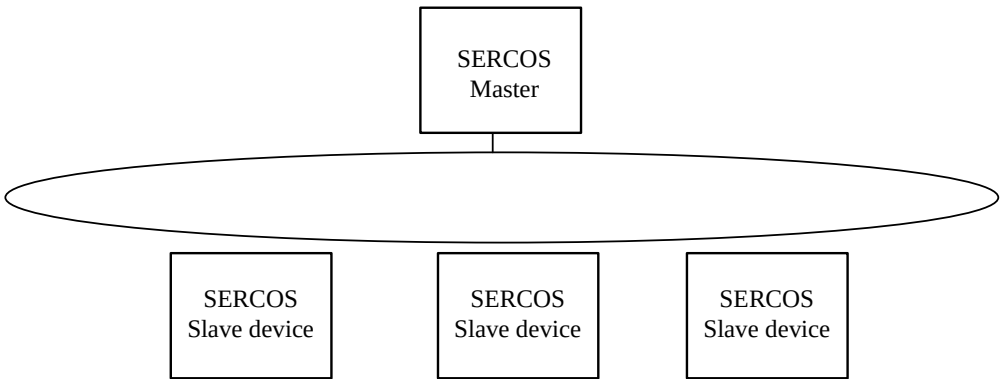


Figure 3.3.6. Bus configuration of SERCOS

Commercial issues

Cost of controller	SERCOS controller, 30 USD
Cost of interface media	fiber optic, 30 USD
Cost for certification	--
Certification needed	yes, to be allowed to use the SERCOS label
User's organization	Yes, IGS Interest Group SERCOS interface
More information at	www.sercos.com

Technical Description

Physical transmission medium

Standard	EN61491 (IEC1491)
Transmission media	Fiber optic or RS485 twisted pair
Distance between nodes	60m with plastic fiber, 250m with glass fiber
Number of nodes	max 126

Controller level

Standard	EN61491 (IEC1491)
Type	Master/Slave ring network with Single Master
Transmission speed	2 or 4 Mbps (extended 8 or 16 Mbps)
Message length	2 till 18 byte
Information in a message	0 to 16 byte configurable
Message synchronization	yes, broadcast
Sync. Jitter	1us max +/-7 us
Coding	NRZI-coded
Cycle time (ms)	0.062, 0.125, 0.25, 0.5, 1, 2 …
None cyclic data	yes, 8 bit
Control chip, ASIC	Yes, SERCOS 410 SGS-Thomson

Software protocol

The software protocols and the profile for SERCOS is defined the standard EN61491.

Functionality

In operation, communication takes place cyclically as master/slave communication at the cycle time selection at initialization. This can be 0.062ms, 0.125ms, 0.25ms, 0.5ms, 1ms or any whole multiple of 1ms. The cycle times are specified in order to obtain the necessary synchronization with fixed working cycle times in the controller and drives. The controller is always the communication master in a SERCOS interface.

Communication takes place via three different types of messages:

- Master Synchronization Telegram (MST). The master sync message is received and simultaneously by all drives and is used to synchronize all time-related actions in the controller and drives.
- Master Data Telegram (MDT). Like the master sync message, the master data message also received simultaneously by all drives. It contains the cyclical data and service data for all drives on the ring.
- Drive Telegram (AT). The drives send their message in succession in allocated time slots.

Software profile

Function for drives are defined in different modes like position control, Velocity control and Torque control. For the defined modes are the message with ID 1 to 32767 used to specify functional parameters and also their resolution and scaling.
ID number 1 to 32767 are reserved for data specification by the SERCOS interface association.

The ID numbers 32768 to 65535 are available to product manufactures for definition of data or parameters not covered by the standard but witch are required for the operation of the product.

Dependability description

Detection and handling of communication errors

In the SERCOS interface, communication errors are detected with high reliability by the HDLC protocol and additional monitoring of the known transmission times. The Hamming distance is greater then 4.
Errors in the transmission of service data (parameters and diagnostic signals) are corrected via handshaking procedure. Erroneous data is transmitted.
Real-time data (set-points and actual values) is corrected automatically by updating in each communication cycle. In the case of communication errors, the last valid set points are used until the next cycle. The drive (Slave device) is stopped in the event of two successive erroneous transmissions

System safety with SERCOS interface.

Digital intelligent drives with the SERCOS interface provide outstanding protection against uncontrolled drive movements and excessive speeds.
The drives' internal intelligence provides perfect self-monitoring with the aid of the position values and drive parameters, combined with forced shutdown in the event of a malfunction or failure of the drive processor.
Furthermore, excessive axis speed or run-away due to faulty or incorrectly transmitted position set points can be completely eliminated by logical monitoring in the drive processor of the set point received by the controller.
Safety redundancy is achieved by monitoring the actual value data feedback to the controller via the SERCOS interface.
Even in the event of a communication failure, safe shutdown of the drive is ensured by the drive's internal monitoring and also by the monitoring the controller in conjunction with a higher-order emergency-stop circuit.

Appendix A. References

[PALBUS] Project plan “Project description PALBUS - Validation of Dependable Bus Systems “ (in Swedish)
Technical notes (“Arbetsrapport“) no. SP-AR 1999:31
issued by SP Swedish National Testing and Research Institute

[1] Kopetz H., (1998) ”A comparison of CAN and TTP”, 15th IFAC Workshop on Distributed Computer Control Systems, (DCCS 98), Como, Italy

[2] Kopetz H., (1997), ”Design Principles for Distributed Embedded Applications”, Kluwer Academic Publishers

[3] Laprie J.C. (ed.), ”Dependability: Basic Concepts and Terminology, Dependable Computing and Fault-Tolerant Systems”, Vol.5, Springer-Verlag (1992)

[4] Christian, F., (1991) ”Understanding Fault Tolerant Distributed Systems”, Communication of the ACM

[5] SS-ISO 11898 Road vehicles- Interchange of digital information- Controller Area Network (CAN) for high-speed communication

[6] CAN Kingdom 3.01 specification

[7] Specification of the TTP/C Protocol; Specification version 0.5 of 21-Jul-1999; Document edition 1.0 of 21-Jul-1999

[8] Implementation of a Distributed Control Application Based on the TTP/C Architecture; Jonas Bolin, Johan Hedberg

[9] MIL-STD-1553, An Interpretation of MIL-STD-1553B

[10] EN 50170: Standard for a European Fieldbus Computer Standards Interfaces

[11] EN 61491, SERCOS specification

Appendix B. PALBUS Partners

Following partners participate in the PALBUS project:

Aros Elektronik, Mölndal
Arcticus Systems AB, Järfalla
InMotion AB, Stockholm
CR&T, Gothenburg
QRtech AB, Mölndal

KVASER, Kinnahult
NDC, Särö
NOB Elektronik, Älmhult
Scania, Södertalje
THOREB AB, Gothenburg

Volvo Teknisk Utveckling, Gothenburg
Chalmers University of Technology, Department of Computer Science, Gothenburg
IVF, Swedish Institute of Production Engineering Research, Mölndal
SP Swedish National Testing and Research Institute, Borås