CANopen with redundant communication channels

Holger Heinemann, Ralph Mösle, Vector Informatik GmbH

Abstract

In some areas in which CAN is utilized the systems involved are subject to more stringent safety requirements. Such systems need to be fault tolerant to a certain extent, and the redundant design of communication buses offers the required property. Using CANopen as higher layer protocol requires some supplements to the standard communication profile since this does not explicitly provide support for redundant communication channels. This paper looks at the possible error types occuring in a CAN network, the strategy of redundancy and the CANopen protocol enhancements. Any additional protocol definitions described here are proprietary so far. Further standardization work done by a CiA interest group may be expected.

Introduction

In some areas in which CAN is utilized the systems involved are subject to more stringent safety requirements. On the one hand these requirements are derived from potentially serious damage (to human life or the environment) when the system fails. On the other hand individual components may be exposed to more severe operating stresses and therefore carry a higher risk of defects. To deal with this situation effectively it is necessary to design such systems to be fault tolerant to a certain extent, and consequently to introduce redundancies.
In the maritime industry too, in which ships are being equipped with more and more electronic equipment (machine controllers, doors, alarms, etc.), reliable communication between the individual components is essential in such "difficult areas". In the CAN networks which are used for this reason, i.e. because they have implicit fault-tolerant properties and can compensate for temporary disturbances, the bus wires in marine applications are especially subjected to high mechanical and chemical stresses (salt water!) which can lead to permanent damage. Therefore in an environment that does not offer any apparent "failsafe" mechanisms in these cases further preventive actions must be taken to enhance protection against failure. In this case the redundant design of communication buses whereby their bus wires are laid out to be spatially separate offers the required expansion of fault tolerant properties needed for the system.
Vector Informatik, in a joint project with the University of Ulm, has developed a redundant CAN system especially for such requirements. The constraints for this project were the requirements specified by SIG Maritime Electronics within CiA, in which "redundant CAN" is a very current topic in relation to implementations on vessels:

- CANopen as Higher Layer Protocol [1]
- Full compatibility to CANopen standard is assured
- Special hardware is avoided; only standard components are used
- Integrity and real-time behavior are considered
- Reintegration of a node after a fault has been corrected

Possible Error Types

Temporary faults on the bus are handled very well by CAN itself, since many error situations are detected and this results in automatic retransmission of the faulty message. However when there are permanent faults that require human intervention for repair, the system is (completely) unavailable for longer periods of time, which may have serious consequences. Some of these permanent faults may indeed permit a continuation of communications, but the signal-to-noise ratio is significantly impaired. This makes communications more susceptible e.g. to electromagnetic disturbances and more errors occur during communication resulting in more error frames and message repetitions, and this has a negative influence on system time behavior. However, in the case of a complete wire break system-wide communication is no longer possible. In such situations only an available redundant CAN channel can assure the preservation of data interchange. The redundant CAN is designed to protect a system against the bus faults defined in ISO 11519-2 [2] shown in Figure 1.

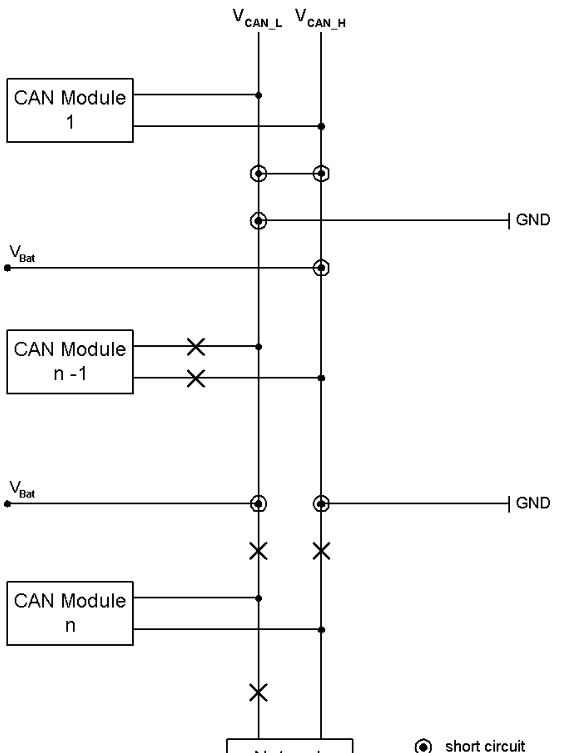

Figure 1: ISO 11519-2 bus faults

Hardware

To keep costs as low as possible and permit the use of various types of hardware, only standard components should be implemented. Handling of the buses would occur in the software of the nodes. Such a system has two completely separate CAN buses, whereby the individual nodes each have two CAN controllers and transceivers. There is no need to rely on special features of the specific hardware, since timeouts can be parameterized to detect communication problems on the bus. This achieves independence from the underlying hardware, instead relying directly on the CAN driver.

Strategy

Because continuous checking of both CAN buses for their availability is required, and there are tight time constraints which only permit short delays, it is not possible to implement a Cold Standby concept [4]. This conclusion is reached based on values specified in various certification rules, which prescribes a maximum failure time of 2 seconds specifically for maritime applications.
A strict Hot Standby variant on the other hand introduces new problems: Proper sequencing of the twice transmitted messages requires sequence numbers, which are associated with an expensive administrative mechanism (implementation expense). Furthermore, these sequence numbers would have to be coded in the individual messages, and the CANopen protocol does not provide any leeway for such coding.
Therefore, a Warm Standby concept was implemented which communicates regularly over both buses, but does not transmit any messages twice, with the exception of the network management commands (NMT). This is achieved by allocating the various communication objects to the available physical buses shown in Figure 2.
Even if practical possible, this design will not provide routing functionality to act a module as a router between two partly damaged CAN busses, since there is no network layer defined in the CAN reference model [3].

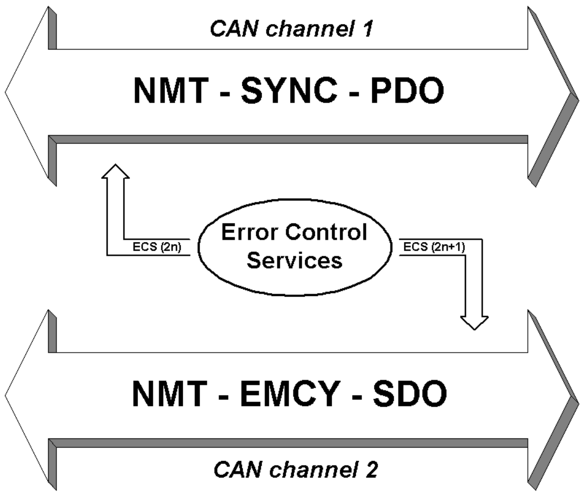

Figure 2: default channel allocation

Regular monitoring of the communication capability of a Slave on the two buses is performed by Node Guarding (or Heartbeat), whereby this mechanism is performed in alternation between the two channels and the system therefore discovers any occurring faults within a clearly defined time period.
The detection of a fault on one of the buses is not based on the evaluation of various status information of the CAN controller, rather it is achieved independent of this information by means of separate timeouts. An individual timeout can be assigned to each communication object, and the timeout is adapted to the priority or cyclic period of the specific message. A certain degree of flexibility is achieved by this method, whereby a communication problem does not necessarily imply a fault of the CAN hardware, rather it might also be caused by the delay of a (low priority) message due to high bus load and arbitration losses. In such a situation a type of load distribution is conceivable, whereby care must be taken to only generate the designed base load on one bus, since in the event of failure of one CAN bus the entire communication definitely only has one bus available. Otherwise the system tolerates temporary disturbances on the transmission medium within the framework of the specific timeout and utilizes the protection mechanisms of the CAN protocol (transparent) in this way.
When a timeout elapses a Slave reacts autonomously by deactivating the responsible bus and shifting its entire communications to the remaining bus which is indicated to the system by transmission of an emergency message (EMCY). If it involves a local disturbance of a Slave (e.g. loose contact in the connector) the remaining nodes might remain unaffected by this under some circumstances (no Slave-Slave communications) and only the Master would need to change its configuration for this Slave. Otherwise, the Master would perform a suitable reconfiguration of the indirectly affected nodes.
Also contributing to the flexibility of the system is the individual configurability of the Slaves which includes their utilization of the buses and the distribution of individual message types to these buses. Expansion of the object dictionary and new NMT commands and error codes permit the configuration, control and diagnosis of the buses.
Another feature is the checking and subsequent reintegration of a CAN channel that failed for one or more Slaves. This could be performed by having a node that only communicates over <u>one</u> bus periodically check the other bus by attempting to transmit a special message; if successful the Master would restore it to its regular operational state.

Protocol Expansions

The expansion of the object dictionary involves the following entries that can be accessed by the service data object (SDO):

- Transmit Table: Allocations of individual transmit messages (Communication Objects) to the CAN channels
- Timeout Error: Acquires elapsed timeout of transmit message groups
- Timeout Reload Values: Configurable timeout values for individual transmit messages
- Switchover: Activate switchover to <u>one</u> bus when timeout has elapsed

- Check CAN: Activate periodic check of the second CAN channel after switchover

The NMT Master can also control - with supplemental NMT commands - the utilization of the available buses, whereby each Slave would report any status change that occurs by means of an EMCY message, thereby informing the system of its status.

Object Dictionary Entries

Object 5000h: SDO Server Timeout

The SDO server timeout defines the current and the default communication channel as well as the timeout value for the SDO response message. If the timeout value is 0 the corresponding entry is not used. The timeout has to be a multiple of 1ms. Valid channel numbers are 1 (first channel), 2 (second channel) and 0 (alternation between the two channels). Each entry is linked to the dedicated SDO server parameter (Objects 1200h-127Fh).

Bits	31-24	23-16	15-0
Value	current channel	default channel	timeout
Encoding	UNS.8	UNS.8	UNS.16

Figure 4: Structure of the Timeout Configuration Entry

OBJECT DESCRIPTION

INDEX	5000h
Name	SDO server timeout
Object Code	ARRAY
Data Type	UNSIGNED32
Category	Mandatory

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1..128
Default Value	No

Sub-Index	1h - 80h
Description	timeout configuration
Entry Category	Conditional; Mandatory for each supported server SDO
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (see Figure 4)
Default Value	No

Object 5001h: SDO Client Timeout

The SDO client timeout defines the current and the default communication channel as well as the timeout value for the SDO request message. Each entry is linked to the dedicated SDO client parameter (Objects 1280h-12FFh).

OBJECT DESCRIPTION

INDEX	5001h
Name	SDO client timeout
Object Code	ARRAY
Data Type	UNSIGNED32
Category	Conditional; Mandatory if client SDOs supported

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1..128
Default Value	No

Sub-Index	1h – 80h
Description	timeout configuration
Entry Category	Conditional; Mandatory for each supported client SDO
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (see Figure 4)
Default Value	No

Object 5002h: TX PDO 1-128 Timeout

The transmit PDO timeout defines the current and the default communication channel as well as the timeout value for the PDO message. Each entry is linked to the dedicated transmit PDO communication parameter (Objects 1800h-187Fh).

OBJECT DESCRIPTION

INDEX	5002h
Name	TX PDO 1-128 timeout
Object Code	ARRAY
Data Type	UNSIGNED32
Category	Conditional; Mandatory if transmit PDOs supported

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1..128
Default Value	No

Sub-Index	1h - 80h
Description	timeout configuration
Entry Category	Conditional; Mandatory for each supported PDO
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (see Figure 4)
Default Value	No

Object 5003h: TX PDO 129-256 Timeout

Different from object 5002h each entry is linked to the transmit PDO communication parameter (Objects 1880h-18FFh).

Object 5004h: TX PDO 257-384 Timeout

Different from object 5002h each entry is linked to the transmit PDO communication parameter (Objects 1900h-197Fh).

Object 5005h: TX PDO 385-512 Timeout

Different from object 5002h each entry is linked to the transmit PDO communication parameter (Objects 1980h-19FFh).

Object 5006h: EMCY Timeout

The EMCY timeout defines the current and the default communication channel as well as the timeout value for the emergency message. The entry is linked to the emergency COB-ID (Object 1014h).

OBJECT DESCRIPTION

INDEX	5006h
Name	EMCY timeout
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (see Figure 4)
Default Value	No

Object 5007h: Error Control Timeout

The error control timeout defines the current and the default communication channel as well as the timeout value for the guarding/heartbeat message. The default channel should be set to 0.

OBJECT DESCRIPTION

INDEX	5007h
Name	error control timeout
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (see Figure 4)
Default Value	No

Object 5008h: SYNC Producer Timeout

The SYNC producer timeout defines the current and the default communication channel as well as the timeout value for the SYNC message. The entry is linked to the SYNC COB-ID (Object 1005h).

OBJECT DESCRIPTION

INDEX	5008h
Name	SYNC producer timeout
Object Code	VAR
Data Type	UNSIGNED32
Category	Conditional; Mandatory for SYNC producer

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (see Figure 4)
Default Value	No

Object 5009h: TIME Producer Timeout

The time stamp producer timeout defines the current and the default communication channel as well as the timeout value for the time stamp message. The entry is linked to the TIME COB-ID (Object 1012h).

OBJECT DESCRIPTION

INDEX	5009h
Name	TIME producer timeout
Object Code	VAR
Data Type	UNSIGNED32
Category	Conditional; Mandatory for TIME producer

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED32 (see Figure 4)
Default Value	No

Object 500Ah: Channel 1 Error Summary

Acquires elapsed timeout errors of transmit message groups on channel 1. Writing a 0 to any group timeout counter will reset this. Values higher than 0 are not allowed to write. This has to lead to an abort message (error code: 0609 0030h).

OBJECT DESCRIPTION

INDEX	500Ah
Name	channel 1 error summary
Object Code	ARRAY
Data Type	UNSIGNED16
Category	Optional

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0..7
Default Value	No

Sub-Index	1h
Description	total timeouts
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED16
Default Value	0

Sub-Index	2h
Description	SDO timeouts

Sub-Index	3h
Description	PDO timeouts

Sub-Index	4h
Description	EMCY timeouts

Sub-Index	5h
Description	error control timeouts

Sub-Index	6h
Description	SYNC timeouts

Sub-Index	7h
Description	TIME timeouts

Object 500Bh: Channel 2 Error Summary

Different from object 500Ah this object deals with channel 2.

Object 500Ch: Self-controlled Switchover

Activate self-controlled switchover to single bus communication when timeout has elapsed.

OBJECT DESCRIPTION

INDEX	500Ch
Name	self-controlled switchover
Object Code	VAR
Data Type	UNSIGNED8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	0..1
Default Value	1

Object 500Dh: Check Inactive Channel

Activate periodic check of the inactive CAN channel after switchover.

OBJECT DESCRIPTION

INDEX	500Dh
Name	check inactive channel
Object Code	VAR
Data Type	UNSIGNED8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	0..1
Default Value	1

NMT Module Control Services

Communicate on Both Channels

Through this service the NMT Master sets the communication of the selected NMT slaves to use both channels according to the default table (object dictionary entry 5000h).

Parameter	*Indication/Request*
Argument	**Mandatory**
Node-ID	selection
All	selection

Table 1: Communicate on both channels

The service is unconfirmed and mandatory. After completion of the service, both communication channels are active.

Communicate on Channel 1

Through this service the NMT Master sets the communication of the selected NMT slaves to use only channel 1 for all communication objects.

Parameter	*Indication/Request*
Argument	**Mandatory**
Node-ID	selection
All	selection

Table 2: Communicate on channel 1

The service is unconfirmed and mandatory. After completion of the service, only the first communication channel is active.

Communicate on Channel 2

Through this service the NMT Master sets the communication of the selected NMT slaves to use only channel 2 for all communication objects.

Parameter	*Indication/Request*
Argument	**Mandatory**
Node-ID	selection
All	selection

Table 3: Communicate on channel 2

The service is unconfirmed and mandatory. After completion of the service, only the second communication channel is active.

NMT Module Control Protocols

Communicate on Both Channels Protocol

This protocol is used to implement the 'Communicate on Both Channels' service.

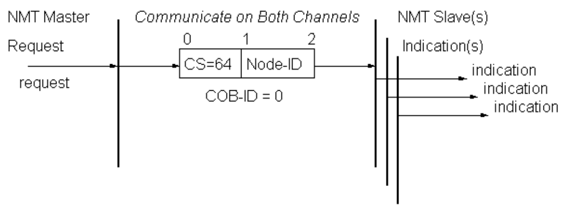

Figure 5: Communicate on Both Channels

- **cs:** NMT command specifier
  64: communicate on both channels
- **Node-ID:** The Node-ID of any NMT Slave or 0. If 0, the protocol addresses all NMT Slaves.

Communicate on Channel 1 Protocol

This protocol is used to implement the 'Communicate on Channel 1' service.

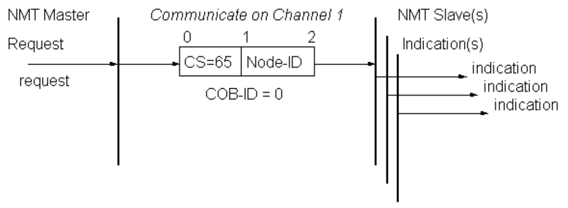

Figure 6: Communicate on Channel 1

- **cs:** NMT command specifier
  65: communicate on channel 1
- **Node-ID:** The Node-ID of any NMT Slave or 0. If 0, the protocol addresses all NMT Slaves.

Communicate on Channel 2 Protocol

This protocol is used to implement the 'Communicate on Channel 2' service.

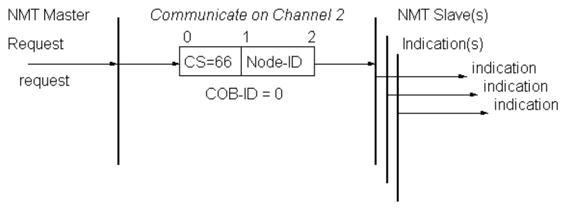

Figure 7: Communicate on Channel 2

- **cs:** NMT command specifier
  66: communicate on channel 2
- **Node-ID:** The Node-ID of any NMT Slave or 0. If 0, the protocol addresses all NMT Slaves.

EMCY Error Codes

Extending the appropriate EMCY standard error codes by the channel number will lead to the possible events in Table 4.

Error Code	Meaning
8101h	communication failure on channel 1
8102h	communication failure on channel 2
8141h	recovered from bus off, channel 1
8142h	recovered from bus off, channel 2

Table 4: Emergency Error Codes

Conclusion

The concept presented does not preclude a system expansion to more than two buses, and it offers a certain degree of flexibility on the application side. Since the use of special hardware is avoided and the administrative effort is quite low for the redundancy strategy presented, it would be possible to expand an existing CANopen protocol stack by the relevant functionality and to thereby implement a redundant system that runs on standard components.

References

[1] n.n.:
CANopen Application Layer and Communication Profile. CiA DS 301 V4.01, CAN in Automation e.V., June 2000.

[2] n.n.:
ISO 11519-2: „Road vehicles – Low-speed serial data communication – Part 2“, 1994.

[3] n.n.:
ISO 7498-1: „Information technology - Open Systems Interconnection - Basic Reference Model“, 1994.

[4] Montenegro, S.:
„Sichere und fehlertolerante Steuerungen“, Hanser Verlag, 1999.