CANopen on CAN FD

Dr. Martin Merkel, Ixxat Automation GmbH

Following the presentation of the new CAN FD technology by Bosch during the international CAN Conference in 2012, CAN in Automation has initiated work to extend the CANopen standards to incorporate the new possibilities offered by this enhanced CAN standard.

To support this standardization effort, the technical activities previously performed within the interest group CANopen have been spun off to a new CANopen SIG application layer which currently works on an update of CiA 301, the basic CANopen specification.

This paper presents the current status of the standardization process within the SIG application layer and also discusses possible enhancements of different device profiles with respect to the new technology.

Introduction

At close to 20 years old, CANopen continues to be one of the most popular higher level protocols available for CAN, the Controller Area Network.
Despite the bandwidth limitation of 1 Mbit/s – which seems to be outdated in a world of Gigabit Ethernet and fast wireless networks – CAN offers many advantages as the data link layer for industrial communication protocols such as CANopen due to the event driven communication and the CSMA/CR bus arbitration scheme.
The main reason of the success of CANopen appears to be its flexibility that resulted in the development of an ever increasing number of device and application profiles in areas that were not considered at all during the initial design work on CANopen. Admittedly in some of these application domains higher communication bandwidth becomes more and more important. This resulted in the increasing popularity of Ethernet based industrial protocols, with some of them essentially reusing the device or application profiles originally developed for CANopen.
CAN based communication networks are however still the solution of choice for applications that do not inherently have the highest bandwidth requirements for a number of reasons:

- Large number of available micro controllers with integrated CAN controller(s)
- Significantly lower connection costs per CAN node compared to Ethernet
- High communication reliability due to advanced error detection functionality
- Efficient fault confinement

The presentation of the CAN with flexible data rate (CAN FD) data link layer protocol by Robert Bosch GmbH at the international CAN Conference in 2012 [1], promises to significantly reduce the performance disadvantage of CAN compared to Ethernet based technologies.
In the sections below we first give a brief introduction into CAN FD and then discuss the standardization process with the SIG application layer for a new version 5.0 of CANopen based on the CAN FD technology.

CAN FD in Brief

Retaining the core features of CAN, Bosch has improved the CAN protocol in two different areas:

- Increase of the transmission speed in the data field
- Increase of the data payload to up to 64 byte

CAN FD can operate with two alternate bit rates. In the arbitration phase and during acknowledgment of the CAN FD frame the bit rate of the bus is still limited to 1 Mbit/s, depending on the overall network length.

This is a result of the requirement that the signal propagation time between any two nodes in the network is less than half of one bit time.
After the bus arbitration the bit rate may be switched to a higher bit rate for the data field. The CAN FD frame format uses the reserved bit *r0* in the CAN control field to select the modified frame format. Due to the backwards compatibility of the new protocol, CAN FD controllers can operate in standard CAN communication networks, which again allows for a gradual integration of CAN FD capable devices into existing CAN systems.

Work plan of the SIG Application Layer

The discussions within the CiA working group defining a CANopen communication protocol based on CAN FD started on the assumption that the maximum payload available for user data will be 64 byte. The configuration of the higher bit rate during the data field of the CAN FD frame has been considered to be outside the scope of CiA 301 and therefore was referred to the task force LSS (Layer setting services) which will work on an update of [6] that includes additional services and protocols to configure the CAN FD bit timing. During the inaugural meeting of the SIG application layer in March 2013 the following work plan was defined:

- Clarification of CANopen NMT and error FSAs (not related to CAN FD)
- Revision of PDO services and protocols
- Enhancement of SDO services
- Other protocols and services
- Introduction of new object dictionary entries (partially related to CAN FD)

Not addressing the FSA issues and additional object dictionary entries, the following sections will focus on PDO and SDO protocols, followed by a brief discussion of the other protocols such as NMT, SYNC, EMCY, and TIME.
We also like to point out that this paper represents the status of the discussions within the SIG application layer at the time of writing of this paper and may not necessarily correspond to the final protocols that will be published in CiA 301 version 5.0.

PDO Services in CAN FD Networks

As already discussed during the international CAN Conference 2012 [2], extending the specification of the PDO services and protocols to make use of the extended CAN FD payload is straightforward. The current PDO mapping parameter records at the object dictionary indices following 1600_h and $1A00_h$ can be retained without modification. Each of these records represents an array of individual mapping entries of data type UNSIGNED32 that contain index, sub-index, and length of the mapped application object.

Index		Sub-index	Length
31	16	15 8	7 0
3	2	1	0

Figure 1: Structure of PDO mapping entry

In the published CiA 301 specification [3] the number of mapping entries is limited to 64 allowing for up to 64 BOOLEAN objects to be mapped. Theoretically, this number could be increased to 253, considering that the values 254 and 255 for sub-index 00_h are reserved to indicate that the corresponding PDO is formatted as a multiplexed PDO. The only use case that would profit from such an increase is the requirement to map a significant number of BOOLEAN objects.
The discussions within the SIG application layer resulted in consent that mapping of BOOLEAN objects has a negative impact on the device performance. Given that transmission of single bits is neither supported by classical CAN nor CAN FD it was therefore suggested to introduce a minimum PDO mapping granularity of one byte. Doing so would also solve the issue with the complexity of mapping a number of BOOLEAN objects that is unequal to 8 followed by objects of a size which is a multiple of one byte.
As some CiA device profiles currently specify BOOLEAN parameters (see for example [7], [9], [10], [11], [12], and [13]) that are partially also used as part of the default PDO mapping [14], discontinuing support for single bit granularity will not be possible without changes to those device profiles.

It is therefore suggested to retain the current definitions with respect to the PDO mapping but encourage the SIGs to refrain from using data types with a length that is not an integer multiple of one full byte.

Another aspect of CAN FD is that not all controllers may support 64 byte data fields. Configuration software therefore cannot assume that the maximum data field size will be available for mapping of application objects. At run-time this is addressed with the SDO abort code 0604 0042_h indicating that the accumulated length of the mapped application objects exceeds the available PDO length. During offline system configuration this information is not available. The SIG application layer will therefore propose a new object in the communication profile area of the object dictionary that shall indicate the CAN FD capabilities of a device.

Due to the variable granularity of the CAN FD data field length (see Table 1), run-time verification of the correct PDO length by the receiver will not be possible either or only of limited use.

Table 1: Supported lengths of a CAN FD data field

DLC	Number of data bytes
0	0
..	
8	8
9	12
10	16
11	20
12	24
13	32
14	48
15	64

In the case the configured PDO mapping does not use the entire data field up to the next possible length, SIG application layer proposes to set the trailing bytes to the value 00_h. A PDO with a configured data content of 14 bytes will therefore be transmitted as a frame with 16 data bytes, including 2 trailing 00_h bytes. As 00_h may constitute valid data the receiver cannot detect if a configuration error is present or not. Emergency messages with error codes 8210_h (PDO not processed due to length error) or 8220_h (PDO length exceeded) may therefore become obsolete.

In summary, the specification of PDO services can be retained as is. 64 entries in the PDO mapping record will allow mapping of 64 byte parameters – which is the typical data size used for digital channels – into one PDO. More important, up to 32 INTEGER16 values – the representation typically used for analog data – will fit into one PDO, compared to the 4 analog signals that could be mapped into one data frame with classic CAN.

SDO Services in CAN FD Networks

The specification of SDO services that make use of the extended frame format available with CAN FD is not as obvious as with PDO services. In the original design of CANopen, SDO services were optimized for the maximum of 8 data bytes supported by classic CAN within one frame. Extending these services to profit from the maximum data field length of 64 bytes offers interesting new possibilities but also poses some challenges that are inherently given by the current protocol specifications.

The initial requirements for SDO services based on CAN FD as discussed within the SIG application layer were:

- Backwards compatibility with currently specified SDO services
- CAN FD based SDO transfer shall be based on classic SDO block transfer
- Data transfer shall already be possible during initialization
- The frame length shall be adapted to the size of the application data
- Reading respectively writing of entire complex objects should be possible

After the first round of discussions within the SIG application layer members expressed concerns to base an enhanced CAN FD capable SDO protocol on the block transfer mode as this may impose high requirements on micro controller performance and buffer size to handle long bursts of 64 byte CAN FD frames.

It was therefore suggested to initially pursue a path where a CAN FD SDO mode would be based on the classic normal (segmented) SDO transfer.

The introduction of an entirely new service, complementing the currently specified communication modes expedited, normal (segmented), and block transfer that covers the requirements listed above is not possible as all available command specifiers (Table 2) are already allocated for existing services.

Table 2: Command specifiers defined for SDO protocols

Command specifier	**Description**	**c/s**
Protocol SDO download		
1	Initiate download request	c
3	Initiate download response	s
0	Download segment request	c
1	Download segment response	s
Protocol SDO upload		
2	Initiate upload request	c
2	Initiate upload response	s
3	Upload segment request	c
0	Upload segment response	s
Protocol SDO block download		
6	Block download initiate	c
5	Block download initiate	s
5	Block download sub-block	s
6	Block download end	c
5	Block download end	s
Protocol SDO block upload		
5	Block upload initiate	c
6	Block upload initiate	s
5	Block upload sub-block	c
6	Block upload end	s
5	Block upload end	c
Protocol SDO abort transfer		
4	Abort transfer request	c/s
SDO network indication protocol[1]		
7	Network indication request	c
7	Network indication response	s

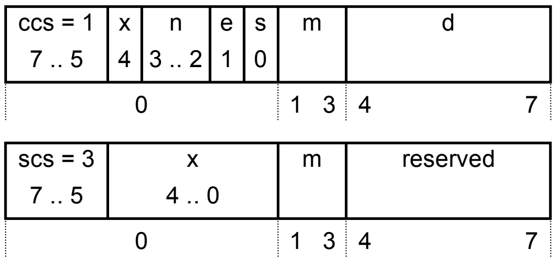
The only option available is therefore to extend one of the currently available SDO protocols to make use of the extended data field.
In the following sections we discuss the three different transfer modes.

Expedited SDO Transfer

SDO services represent point to point communication channels between a SDO client and a SDO server. The transfer is always initiated by the client, and the SDO server will either confirm or abort the transfer, depending on if it can serve the request.

[1] Specified in CiA 302-7 [5]

In a typical CANopen device the majority of object dictionary entries are of size 4 bytes or less. During the original design of CANopen it was therefore felt that an SDO transfer mode is required which is optimized to address such 4 byte or smaller objects. A second design goal was that all SDO transfer modes shall use a common initialization sequence after which it is decided, which transfer mode is finally used. Figure 2 is a representation of the protocol sequence SDO download initiate. For small objects of up to 4 bytes, this sequence constitutes the complete transfer.



ccs	Client command specifier
scs	Server command specifier
e	Transfer type
s	Size indicator
n	Only valid if e = 1 and s = 1, otherwise 0. If valid it indicates the number of bytes in d that do not contain data
x	Not used, always 0
m	Multiplexor, index and sub-index
d	Data

Figure 2: Protocol SDO download initiate

The command byte of the download protocol consists of a command specifier and a number of format flags that indicate the type of the requested transfer mode and – depending on the transfer mode – the size of the transferred application data. In the case of the expedited transfer, the size is indicated in a two bit field *n*, the value of which stands for the number of bytes in the data field *d* that do not contain valid data.
Even assuming that we will not always use the maximum frame length of CAN FD in an extended expedited SDO protocol, we still have the problem that the currently available field *n* will not be sufficient to indicate the data size of a transferred object, as up to 15 bytes may not contain valid object data (size difference due to the

frame length constraints between 48 and 64 byte minus 1 byte user data, see also Table 1).
As the SDO upload protocol (Figure 3) is very much identical to the download protocol, we conclude that using the full 64 byte data field is not possible without significantly modifying the existing SDO protocol definitions for the expedited transfer.

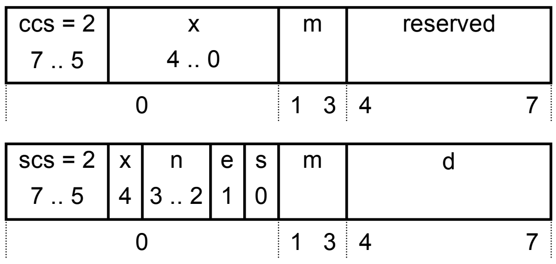
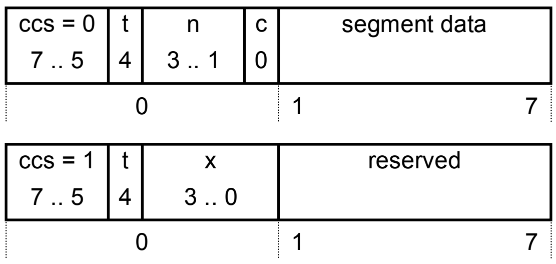


Figure 3: Protocol SDO upload initiate

Therefore, the SIG application layer proposes to retain the current specification of the expedited SDO protocol with frames containing exactly 8 data bytes as is.

Normal (segmented) SDO Transfer

Before CANopen version 4.0, the normal transfer mode was the only means to transfer data blocks larger than the size limit of 8 bytes imposed by classic CAN. With the segmented SDO protocols, the data transfer is continued after the initialization phase with the protocol SDO download (Figure 4) respectively upload (Figure 5).



t	Toggle bit
c	Indicates whether more segments are to be downloaded

Figure 4: Protocol SDO download segment

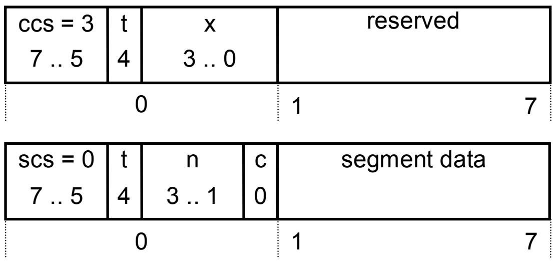
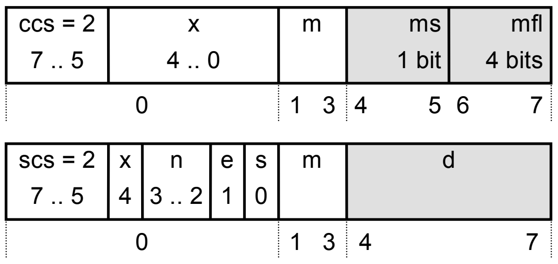


Figure 5: Protocol SDO upload segment

The segment protocol uses a data frame with one command byte and up to 7 bytes of user data and a second full length CAN frame that practically contains no information other than that the previous segment data have been received.
As a result of the significant protocol overhead the segmented transfer offers only limited throughput, but also does not impose high performance requirements on the target hardware as each individual segment is confirmed.
If transposed onto a CAN FD logical layer, one would likely abandon the requirement that both request and response frames in the SDO segment protocol have symmetrical data length, a feature that has been criticized by some already for the implementation on classic CAN.
The SDO specifications in [3] use the combinations [$e = 0$, $s = 1$], [$e = 1$, $s = 1$], and [$e = 1$, $s = 0$] to indicate transfer mode and size of the data to be transferred. For the combination [$e = 0$, $s = 0$] the data field *d* is reserved by CiA for future use.
A possible SDO upload request sequence for a CAN FD capable system could look as described in Figure 6. In the SDO upload initiate protocol, the SDO client needs to provide as additional information the maximum CAN FD frame length supported by the client (*mfl*), and if the upload request shall address a single entry or all entries of a complex object (*ms*). For the maximum supported frame length a coding as in the DLC field of the CAN FD frame (Table 1) will be used. We propose the combination [$e = 0$, $s = 0$] to indicate that those two parameters are be coded in the bytes 4 to 7 that are reserved in [3]. A possible coding with the proposed protocol enhancements highlighted in gray is shown in Figure 6.



n = 0	
e = 0	Segmented transfer
s = 0	Size is not indicated in d
d	See Figure 7

Figure 6: Protocol SDO upload initiate on CAN FD

Current SDO server implementations will ignore the information in the reserved field and continue the transfer using either expedited or normal transfer modes.

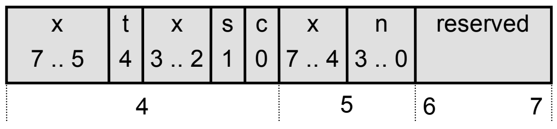
A SDO server that is implemented according to an updated, CAN FD aware CiA 301 specification may use the combination [*e* = 0, *s* = 0] to indicate that format flags are contained in the currently reserved field *d* of the SDO initiate upload response. The format flags need to contain the following information:

- Indication if object size information is provided or not
- Number of (in)valid data bytes in the response frame
- Indication if this response or segment is the last frame in the transfer

For reasons of a simplified implementation it may be attractive to define a format for *d* that could be reused as control word in subsequent segment protocols. This would imply that additional fields are reserved for a command specifier and a toggle bit.

Introducing a segment counter may also appear attractive but is not easily possible if the requirement for an unlimited transfer size is maintained.

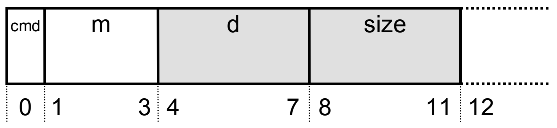
Figure 7 shows a proposal for the data field *d* in the response to the SDO upload initiate protocol. If bit *s* is set to a value of 1, the 4 bytes following *d* in the CAN FD frame (Byte 8 to 11) would contain the size of the object to be transferred (Figure 8).



x	Not used, always 0

Figure 7: Proposed layout for d in the response frame to the protocol SDO upload request

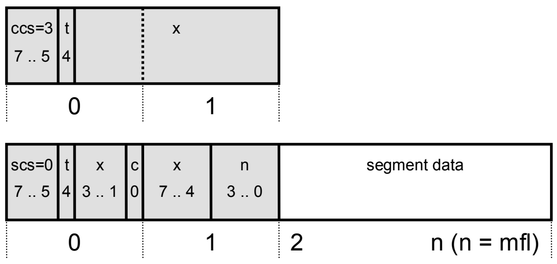
Bytes 2 and 3 currently remain unused and are reserved. The control information could be alternatively shortened to a 2 byte word.



cmd	Command word

Figure 8: Proposed location for the size information within a CAN FD frame

This implies that the proposed protocol will only be applicable to CAN FD controllers that implement the larger data frame format. For CAN FD controllers that only support the bit rate switch, but retain the 8 byte data field, it is suggested to reuse the normal transfer (see Figure 2 to Figure 5) specified in [3] as is.



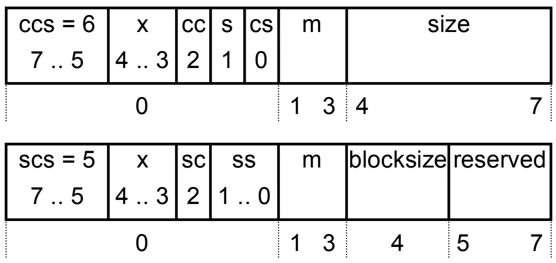
x	Not used, always 0
t	Toggle bit
c	Indicates whether more segments are to be uploaded

Figure 9: Proposed protocol SDO segment upload using CAN FD frames

The proposed enhanced segment protocol (Figure 9) will offer similar functionality as the currently specified normal transfer mode [3], but will offer significant performance improvements due to the reduced protocol overhead, even if less than the full 64 byte CAN FD payload is be used.

SDO Block Transfer

The SDO block transfer was introduced with CANopen version 4.0 to overcome the throughput limitations of the normal transfer mode introduced in the initial CANopen design. The main difference between the two transfer modes normal and block is that block transfer does not require each individual segment to be confirmed. Data are transmitted as a sequence of sub-blocks of 8 byte segments with a maximum burst length of at most 127 segments within a sub-block (Figure 11). Each segment contains a sequence counter and a flag to indicate if the current segment is the last segment in the transfer. A confirmation is only sent after each sub-block. The transfer is concluded with the protocol SDO block download end which contains an optional 16 bit CRC checksum.



ccs	Client command specifier
scs	Server command specifier
cc	Client CRC support
s	Size indicator
cs	Client subcommand
ss	Server subcommand
m	Multiplexor, index and sub-index
blocksize	Number of segments per block with 0 < blocksize < 128
x	Not used, always 0

Figure 10: Protocol SDO block download initiate

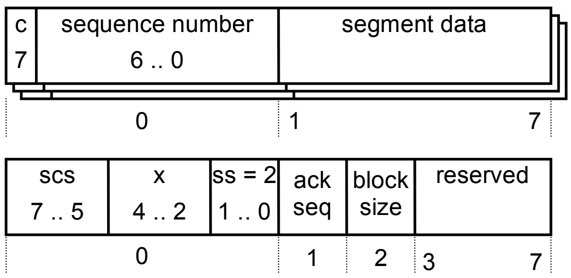
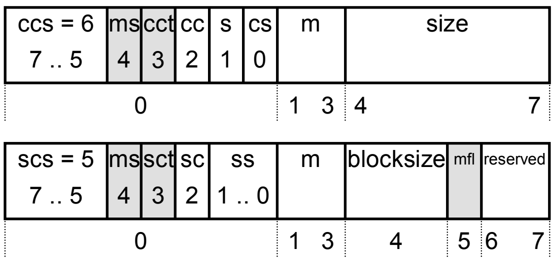


Figure 11: Protocol SDO block download sub-block

To use the extended data field with the SDO block transfer, the SDO client and the SDO server have to indicate their maximum supported frame length (*mfl*). This could be introduced in a similar way as with the normal (segmented) SDO transfer by using one of the reserved bytes in the protocol SDO upload initiate to code *mfl* based on the coding in Table 1. In addition to including the frame length information, it has been proposed to increase the size of the optional CRC checksum from 16 bit to 32 bit. Whether a 32 bit CRC is supported by both client and server or not has to be negotiated in the SDO block download or upload initiate protocol. As the currently available bit flags only allow the negation of using CRC verification or not, one of the reserved bits 3 or 4 in the initiate protocol is used. Here the client would indicate the CRC type it intends to use and the server would confirm if it supports the requested CRC type.

Finally, to indicate that the entire content of a complex object shall be transmitted a flag *ms* is introduced to inform the server which access mode is selected.

Current SDO implementation will not interpret the additional information and fall back to the functionality specified in [3].



ms	Multiple sub-indices selected, the server confirms if the functionality is supported	
cct	Client CRC type, only valid if cc = 1	
	0	16 bit CRC
	1	32 bit CRC
sct	Server CRC type, if unequal to cct, the request CRC type is not supported	
mfl	Maximum supported frame length	

Figure 12: Proposed enhanced protocol SDO block download initiate

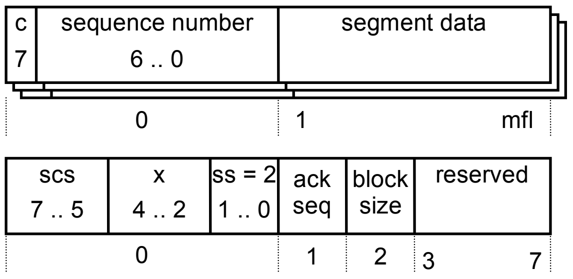


Figure 13: Protocol SDO block download sub-block with increased payload of up to (mfl - 1) byte

If the enhanced normal (segmented) SDO protocol or the enhanced SDO block transfer or both are introduced into the CANopen version 5.0 is currently under discussion within the CiA working group.

Multiple Object Access

SIG application layer has not yet started detailed discussion on how to support a service similar to the Complete Access mode available for example with CANopen over EtherCAT. In CANopen we have to consider that entries in a complex object may have different length, or that the length of the entries is not initially known. Also, it is not always the case that all sub-indices up to the highest supported sub-index are implemented by a device. This makes it inherently difficult to implement such a service, independently of such a service being used on classic CAN or CAN FD.
One possible solution would be to use an object format similar to the concise DCF format described in [4] which includes index, sub-index, object length, and object value in a binary data stream.
The design goals should be to avoid a complex initialization phase for the upload or download of structured objects and to render the service compatible with the SDO protocols discussed earlier in this paper.

CAN FD with other CANopen Services

As pointed out earlier, CAN FD will most likely affect only the PDO and SDO services specified in [3]. Below we enhance the table presented in [2] giving a short summary of the impact of CAN FD on other CANopen services:

Table 3: Impact of CAN FD on CANopen services

Service	Payload	Speed
SYNC	no	no
TIME	no	no
NMT node control	no	no
NMT error control	no	no
EMCY	to be discussed	marginal
LSS	possibly for LSS fastscan	not yet evaluated

Apart from the LSS protocols, which need to be discussed within the SIG application layer task force LSS and a possible payload extension of the emergency message, it is not expected that CAN FD will result in a redefinition of the currently specified services and protocols listed in Table 3.

CANopen Device Profiles and CAN FD

The most significant impact that CAN FD will have on device and application profiles is the possibility to configure PDOs with more than the traditional 8 data bytes. Apart from the increased payload, this will also result in a more efficient allocation of the CAN identifiers. In the default PDO mapping specified in CiA 401 [7], we require 4 CAN-IDs per transmission direction to transmit 64 digital and 12 analog channels. With CAN FD only 2 CAN-IDs will be required as the 12 analog channels could be mapped into one PDO. CiA 402 defines different operation modes for drives and their corresponding PDO mappings. In the current specification [8] which is based on the maximum CAN payload of 8 bytes, each PDO contains either a control or a status word – depending on the transmission direction – and an operation mode specific application parameter. With the increased payload of CAN FD frames, multiple application parameters, such as control word, operation mode, target position, and target velocity, could be transmitted with one data frame, thus simplifying the operation of drives.

As discussed earlier in the section on PDOs, the SIGs responsible for the device and application profiles should consider the necessity for application parameters data types with a length that is not an integer multiple of one full byte.

Conclusion

The technical work on an update of CiA 301 to support the enhanced functionality of CAN FD has started in June 2013. Whereas the initial discussions focused on the clarification of the CANopen communication FSAs, the actual work related to CAN FD begun in August 2013. At the time this paper was compiled, the working group has discussed PDO and SDO services for CAN FD. PDO services will likely not require any changes. For SDO services different proposals have been presented. A final decision which of these proposals will enter the final version of the new CiA 301 specification is expected for the end of 2013. The working group also concludes that no changes are required for the other services and protocols specified in [3].
A discussion if the enhanced payload of CAN FD can be utilized in the CANopen Safety specification [15] is outside the scope of the current work of the SIG application layer and needs to be addressed in a dedicated working group.

Acknowledgements

The author likes to thank the CiA office, the members of the SIG application layer, and the IXXAT CANopen group for the collaboration and the discussions that resulted in the proposals presented in this paper.

Dr. Martin Merkel
Ixxat Automation GmbH
Leibnizstraße 15
DE-88250 Weingarten
merkel@ixxat.de
http://www.ixxat.de/

References

[1] CAN in Automation, Florian Hartwich, Robert Bosch GmbH, CAN with Flexible Data-Rate, Proceedings of the 13th international CAN Conference

[2] CAN in Automation, Heinz-Jürgen Oertel, Using CAN with flexible data-rate in CANopen systems, Proceedings of the 13th international CAN Conference

[3] CiA 301, CANopen application layer and communication profile, Version 4.2.0 including Corrigendum 3

[4] CiA 302, CANopen additional application layer functions – Part 3: Configuration and program download, Version 4.1.0

[5] CiA 302, CANopen additional application layer functions – Part 7: Multi-level networking, Version 1.0.0

[6] CiA 305, CANopen layer setting services (LSS) and protocols, Version 3.0.0

[7] CiA 401, CANopen profile for I/O devices – Part 1: Generic I/O modules, Version 3.1.0

[8] CiA 402, CANopen drives and motion control device profile – Part 3: PDO mapping, Version 3.0

[9] CiA 404, CANopen device profile for measuring devices and closed-loop controllers – Part 1: Generic objects and generic PDO mapping, Version 2.0.0

[10] CiA 413, CANopen device profile for truck gateways – Part 1: General definitions, Version 3.0.0

[11] CiA 413, CANopen device profile for truck gateways – Part 2: Brake and running gear devices, Version 3.0.0

[12] CiA 413, CANopen device profile for truck gateways – Part 3: Other than brake and running gear devices, Version 3.0.0

[13] CiA 413, CANopen device profile for truck gateways – Part 5: Superstructure objects, Version 1.0.9

[14] CiA 850 Recommended Practice CiA 413 based truck gateway, Version 1.0.0

[15] EN 50325–5:2010 Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 5: Functional safety communication based on EN 50325-4