Using CAN with flexible data-rate in CANopen systems

Heinz-Jürgen Oertel, *port* GmbH

Shortly after the Controller Area Network developed by Bosch was introduced in cars, it started to be used in industrial communication networks. The most successful Higher Layer Protocol which is based on CAN is still CANopen. CANopen tried as good as possible to use the technical parameters given by the underlying CAN chips.

Now, after decades, Bosch has extended the CAN protocol to be much more faster. It seems to be possible to overcome the most criticized parameter in CANopen - bus speed and available bandwidth. The paper discusses the new opportunities for CANopen given by the »CAN with Flexible Data-Rate«.

Introduction

Today CANopen is established as a communication network in many different application fields. Even it might not be the fastest network looking only at the bit speed. It has a lot of performance advantages due to the underlying CAN protocol and the event driven communication principle. In the last 15 years a large community of CANopen users has developed more than 50 device and application profiles for CANopen. This is the largest number of profiles a single protocol can offer.

During the last years some applications were demanding higher communication speeds. Especially motion control applications with short cycle times. Ethernet based networks are now going to provide the bandwidth for such applications — and surprisingly or not — some of them with the profiles defined and introduced by CANopen. But still CANopen based networks have advantages making it the better suited for most applications.

- Connection costs per node are significantly lower compared with Ethernet based networks .
- Network reliability is higher because of the complete passive structure of the network. No need for active components like switches or hubs.
- High immunity against electromagnetic interferences.
- Ease of use.
- Countless number of micro controllers with integrated CAN controller.

Now with CAN FD the only weak point in CANopen, the offered bandwidth, can be increased significantly. The following chapters will show how CANopen can benefit from the higher speed and higher payload CAN FD offers.

What is CAN FD?

This paper is not going into detail how CAN FD is working on the bit-level. For all the following meditations the two main advantages are considered

- CAN FD can have a payload of 32 or maybe 64 bytes
- CAN FD increases the bit speed of the data field by up to 8

For more information on CAN FD refer [7] or the original White Paper[1] .

Increased PDO efficiency

The most relevant CANopen network status for process control and bandwidth usage is OPERATIONAL. Nearly 100% of the communication services will be PDO transfers. Because most of the CANopen systems are using the 11 bit base identifier format the efficiency of an default PDO is calculated for this frame format, taking into account that a CAN FD frame can have more data bytes and increased speed while transferring data bytes.

data bytes	CAN	CAN FD	CAN FD$_{max\_speed}$
1	0.1	0.1	1.1
4	0.4	0.4	3.2
8	0.6	0.6	4.5
12	—	0.7	5.3
16	—	0.7	5.8
32	—	0.8	6.7
64	—	0.9	7.3

Table 1: efficiency comparison

(CAN - CAN ISO11898-1; CAN FD - increased payload only; CAN FDmax_speed - increased payload and increased bit speed by factor 8)

The efficiency is calculated as follows:

$$e = (t_{payload} * x) / t_{frame}$$

$$e = (n * 9 * x) / 55 + (n * 9)$$

Efficiency 1 means the overhead in bits, or more accurately time spend in protocol overhead, is going to be zero compared with the payload bits in the frame. Additional to counting bit times in this formula, the payload time only is multiplied with the factor *x*, representing the speed factor of the data phase in CAN FD.

A more exact evaluation of the protocol can be found in [7] .

By using a simple approach to take care of stuff bits included in CAN frames. The worst case of stuff bits is *n*/4, every 5th bit can be a stuff bit. In this table it is assumed that bits in the data frame are distributed in such a way, that the stuff bit to data bit ratio is approximately 1/8.

It was not taken into account, that by the longer date field the size of the CRC field is slightly increased in CAN FD. And 5 bits of the DLC are already sent with the optional high bit-rate. The different rules for stuff-bit inclusion into the CRC field.

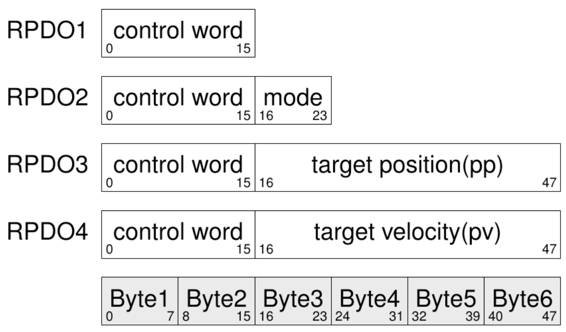
$$s_{max} = 8 + 2 * dlc$$

or to calculate with an average of stuff bits only:

$$s_{av} = 4 + dlc$$

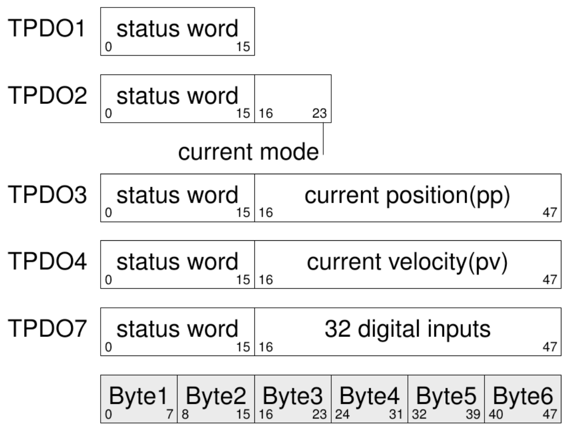
RPDO default mapping for CiA402 devices

A lot of electric drive parameters are 32bit values[3]. To have the full range of accuracy already in the early days the standard was developed, CiA decided to use high resolution for all parameters.

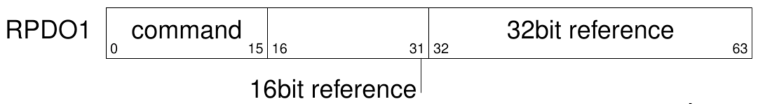


More RPDOs are defined according to the above schema mapping always the control word and 32bit parameters, depending on the drives mode.

TPDO default mapping for CiA402 devices

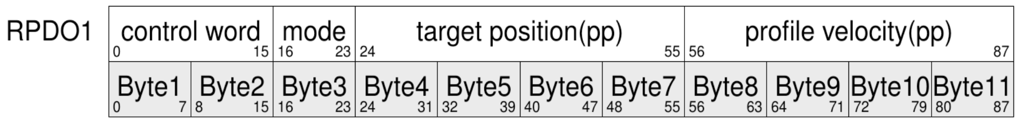


But what is desired? Drive applications need to have more parameters mapped. As a result the CiA profile 452 [5]
defines a new eight byte receive PDO for drives.



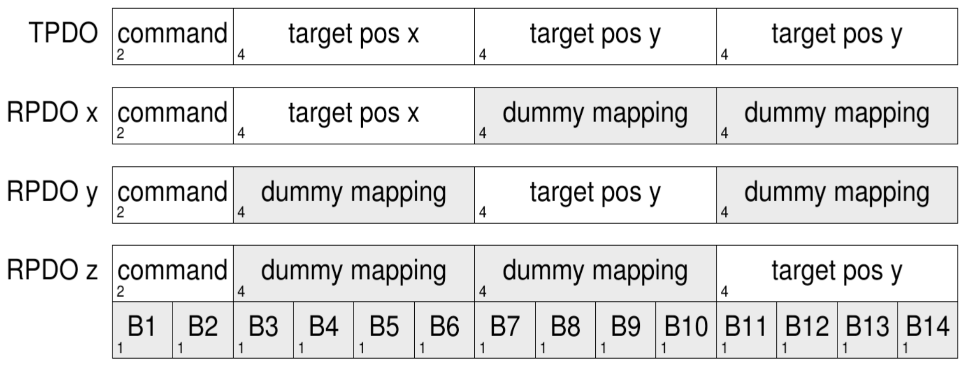
Where the meaning of the second and third mapping entry are defined by special *mode control*, *action* and *sub-action* fields in the *drive command*. This unfortunately led to a very different drives behavior and PDO interpretation compared to CiA 402 drives.

A better approach would be the following RPDO mapping:



It requires a 11 Byte PDO and allows mode control and specifying a new position combined with an associated velocity.

The following figure shows a possible approach for a three axis system. The controller is using a TPDO with the control word and new positions for each axis. Each of the axes is selecting the control word and its new position by using CANopen *dummy mapping*:



The numbers in the boxes above represent the number of bytes per field. Which leads to a 14 byte long PDO.

Possible use in other CANopen profiles

Another profile which could benefit from the increased PDO size is CiA454 [6] the application profile for energy management systems. It describes voltage converters in a very generic format. The AC side of such a converter is described by the voltage, the current and the power factor cos ϕ of three lines of three-phase units. The current standard describes three TPDOs which are send out periodically. To save PDO load, voltage and current are represented as two-byte values and an additional scaling factor.

All the values could be put in one CAN FD PDO and without the need of an additional scaling factor by using 4 byte for voltage and current.

But also CiA401 [4] generic IO modules can benefit from a larger PDO payload. While for digital in/out in most cases it will be sufficient to have 64 bits or say 64 in/out signals, with other modules like analog in/out that is not the case.

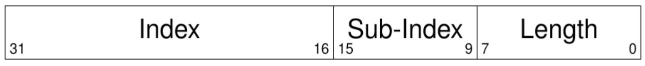
The second to fourth PDO mappings contain four mappings for 2 byte analog values. In real applications the number of analog signal on one modular CANopen node typically is higher. Such modular devices will benefit from the increased payload. Assuming only 32 byte payload, this means that one PDO can carry up to 16 channels instead of 12 channels using three PDOs today. If the devices support dynamic mapping, it will even be possible to have 64 digital in- or out-signals and 12 analog channels in one PDO freely configurable. Of course, PDOs carrying a payload of 64 Byte will double this number again. Assuming 4byte float values, the benefit is even bigger.

Besides all the advantages of a higher payload of course, there will be a drawback too. In event driven systems, changing one mapped object, toggling one digital input of a CiA401 device, will lead to the transmission of the whole PDO with all the digital and analog input signals. This means waste of bandwidth if the 32 byte payload message is larger than a conventional 8 byte PDO. And it will possibly lead to more processing overhead in the consuming devices, because they may need to check all objects mapped into the received PDO.

While extending the number of bytes used for a PDO increases the efficiency of a CAN frame, simply by improving the payload to overhead ratio, the additional increase of frequency also shortens the frame length. Using both leads to equally short frames like in the standard CAN protocol by increasing the payload and efficiency significantly.

Implementing the increased PDO payload in CiA 301

Implementing PDOs with extented payload in CiA301 is easily possible. The only affected objects are the mapping tables[2]. These object entries describe the mapping, and therefore the length of a PDO. Currently the mapping tables at 1600_h and $1A00_h$ are arrays, each entry describing a mapped object as:



CANopen using CAN FD still will use the data types defined in the area of 0000h to $0FFF_h$. The *Length* entry in the mapping table allows data types up to 255 bit −32 byte length. Theoretically the number of entries, currently 64, is the only limiting factor. This value was chosen in CiA 301 to allow each bit in an eight byte PDO to be a single addressed object in the object dictionary. Practically it makes no sense to map BOOLEAN objects. That means that with a maximum of 64 entries it will be possible to map up to 64 byte objects, the maximum number of bytes in the data field of a CAN FD frame. Even if there is a need to increase the number of mappings this change in CiA 301 will be compatible to older versions.

Using CAN FD with other broadcast CANopen services

While it is expected using CAN FD with PDOs will bring most benefits other CANopen services can use CAN FD as well. The following table gives a short summary if one of the services will benefit from the increased payload and speed.

	payload	speed
SYNC	no	no
TIME	no	no
Heart Beat	no	no
EMCY	no	slightly

Table 2: influence of CAN FD on CANopen services

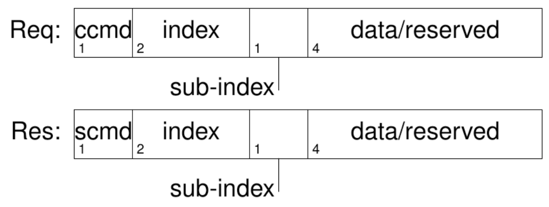
For all these services currently there is no need to increase the number of bytes and besides the 8 byte EMCY message there is no significant shortening of the frame duration using increased data rate.

Using CAN FD with SDO services

Using the extended data frame CAN FD provides with the SDO service is a major challenge. CANopen knows about three different SDO communication modes, each of them optimized for the traditional 8 byte data frame.

Expedited SDO transfer

SDO transfers are point to point client server connections.

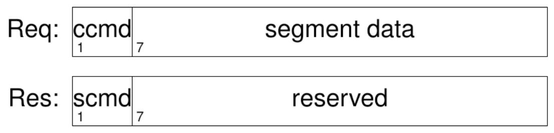


One design goal of CANopen was using always the same frame length for a fixed service. Expedited SDO like the other SDO services is using 8 bytes in each direction, for the clients request and for the servers response. Depending on the data direction, download or upload, the data field in one of the frames is unused (reserved). Increasing the data field for both frames, carrying the data and the frame with the reserved bytes leads to ineffective bandwidth usage because of the increasing number of unused bytes.

Secondly there is no need for changing this service. By addressing objects with data sizes from 1 to 4 byte more than 90% of the entries can be reached. Typical CANopen object dictionaries do not have so much entries with larger data size. To address these object segmented or block transfer can be used.

Segmented SDO transfer

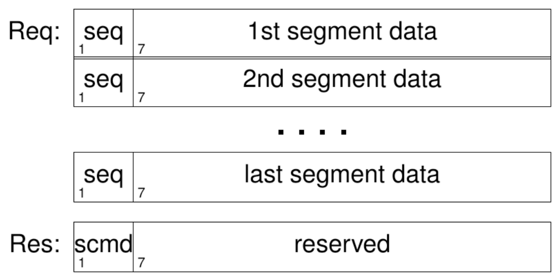
Segmented SDO transfer was originally the means to overcome the data size limit of a CAN frame to transfer objects with a larger data size. It still is in use but not very efficient because one of the frames, depending of the transfer direction, carries only little information in one byte and 7 byte unused. The next figure shows a typical sequence of a SDO download sequence:



Because of the poor performance of the segmented transfer CANopen introduced the SDO block transfer with CiA standard 301 version 4.0. Like with the expedited transfer there is no need for changing this service if using CAN FD.

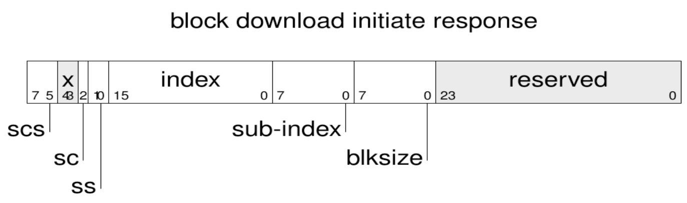
SDO block transfer

CANopen block transfer does not require that each frame is confirmed by the consumer of the data. Instead a block of data frames is sent by the producing side which in sum is confirmed by the consuming side.

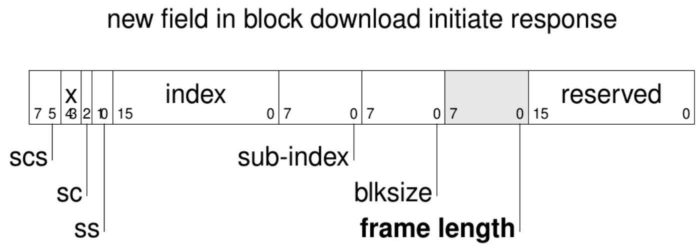


The date frame contains a one byte sequence counter plus a block-end flag and 7 byte payload data like the figure shows. The number of segments in one block can be up to 128. But how can CANopen use the extended data field with e.g. 32 byte in this block?

In the following we look at downloading data to the server. The SDO block transfer is initiated by a special sequence starting with a request done by the SDO client. The request contains the overall size of data in bytes and as usual the destination address on the server as index/sub-index pair. In the response the server will tell the client the block size it will accept and some additional flags. But there are some free bits and bytes which can be used to state that the server accepts other frame length than only the traditional 8 if a CAN FD capable CAN controller is used.



Without touching the two reserved bits 3 and 4 in byte 1. The server can use one of the reserved bytes, let's say byte 6 to code the data frame length which it will accept from the SDO client.



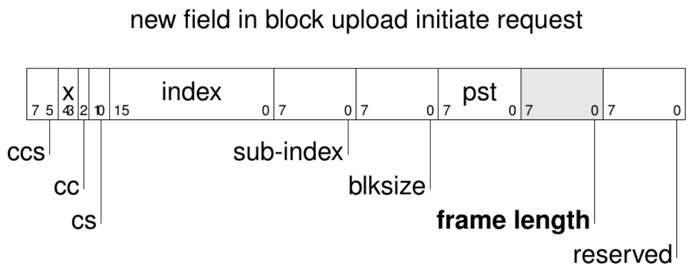
The following table contains the suggested values:

byte 6 value	frame length
0	8
9	12
10	16
11	20
12	24
13	32
14	48
15	64

Table 3: frame length coding

Increased frame length together with the increased data rate will lead to a highly improved transfer speed for task like program or firmware download to CANopen nodes.

SDO block upload can benefit from the increased frame length too. This time the SDO client can use some unused bytes to tell the server not only the number of segments he accepts in one block of data but also the frame length if CAN FD is used at data link layer.



Coding of the frame length field is the same as in table 3 for block download.

Other CAN based protocols

CANopen is only one of the protocols using CAN as data link layer. It seems that DeviceNet is not anymore developed by the ODVA. But J1939 has some market share. The usage of J1939 frames with its PGNs[1] can be compared with the CANopen PDOs. The "mapping" of a PGN is described completely in the standard and contains more than one "signal". J1939 can benefit from CAN FD by combining more signals in one PGN, and/or by increasing the data resolution of the signals.

Conclusion and future development

Interestingly adapting the CANopen communication profile to the extended data link layer of CAN FD needs surprisingly little effort. Adopting the available protocol stacks will too not be difficult. Besides the additional configuration of the data bit rate it needs a clever handling of all the frame buffers with the maximum increased frame size of up to 32 or 64 bytes which still can be important in embedded devices with small memory resources.

While loosing share in traditional automation to Ethernet based communication I'm sure we will see more CAN/CANopen based systems in the future in deeply embedded, may be closed, systems using FPGAs implementing CAN FD and other special functionality. With this transition to CAN FD, controller manufacturers will follow and in some years we will see as many controllers integrating CAN FD as we see today controllers with CAN.

1 PGN Parameter Group number

Heinz-Jürgen Oertel
port GmbH
Regensburger Straße 7b
06132 Halle (Saale)
oe@port.de
http://www.port.de

References

1. CAN with Flexible Data-Rate, Robert Bosch GmbH, Gerlingen, Germany. http://http://www.semiconductors.bosch.de/media/pdf/canliteratur/can_fd.pdf.

2. CAN in Automation, Application Layer and Communication Profile CiA 301 Version 4.2 (07 December 2007).

3. CAN in Automation, Drives and motion control device profile CiA 402-3 Version 3.0 Part 3: PDO mapping (14 December 2007).

4. CAN in Automation, Device profile for generic I/O modules CiA 401 Version 3.0 (03 June 2008).

5. CAN in Automation, Device profile PLCopen motion control CiA 452 Version 1.0 (30 August 2010).

6. CAN in Automation, Application profile for energy management systems CiA 454 Version 1.0.1 Part 3: PDO communication (30 August 2011).

7. CAN in Automation, Florian Hartwich, "CAN with flexible data-rate" in Proceedings of the 13th international CAN Conference.