FIRMWARE MANUAL

UIOLed firmware

Description

The UIOLed firmware is the generic firmware for the entire range of UniSwarm LEDs controllers. These controllers are designed to be compatible with the CANOpen protocol and offer a wide range of features and capabilities. However, the functionality may vary depending on the specific hardware board reference and revision, so it's important to consult the relevant hardware datasheet for additional information.

Features

- Compatibility with the CANOpen protocol, including the CiA DS401 profile
- Capability for low-level digital signal processing

Interfaces

- Compatible with the CAN Fd bus and the CANOpen protocol

Leds

- Support addressable leds

Effects

- Support various effects per segment on up to 10 segments

Please refer to the specific hardware datasheet for more detailed information on the features and capabilities of the UniSwarm LEDs controllers.

Contents

Chapter 1

Led profile

1 Leds segment

Leds strip segments are define recursively. The first segment start at the first led and goes up to s01_Led_count. The second one start at s01_Led_count + 1 and goes up to s01_Led_count + s02_Led_count. Up to 10 segments can be defined like that.

1.1 Getting started

The UIO1LED card can control 10 segments, each with a different effect.
The first important register is s01_Led_count, which defines the number of LEDs assigned to the first segment.
Next, you need to define an effect using s01_Effect: OFF = 0x0000
FIX = 0x0001
FLASH = 0x0101
BREATHING = 0x0102
FLOW = 0x300
FLOW_REVERSED = 0x301

Finally, you can set a color using s01_Color_01.
To turn on the first 10 LEDs in red and the next 5 LEDs in green: s01_Led_count = 10
s01_Effect = 1 (fixed)
s01_Color_01.R = 255
s02_Led_count = 5
s02_Effect = 1 (fixed)
s02_Color_01.G = 255

Please note that the instructions provided are based on the given specifications. Make sure to adjust them accordingly to the specific API or interface you are using to communicate with the UIO1LED card.

2 Multiple segments

The UIO1LED card can control 10 segments. Please check the hardware datasheet for the exact amount of segments supported. Objects of different segments can be accessed with the following addresses:

Axis	Objects index range
Segment 1	0x6000 to 0x60FF
Segment 2	0x6100 to 0x61FF
Segment 3	0x6200 to 0x62FF
Segment 4	0x6300 to 0x63FF
Segment 5	0x6400 to 0x64FF
Segment 6	0x6500 to 0x65FF
Segment 7	0x6600 to 0x66FF
Segment 8	0x6700 to 0x67FF
Segment 9	0x6800 to 0x68FF
Segment 10	0x6900 to 0x69FF

Table 1.1: Index range of specific board object

3 Leds configuration and control

3.0.1 0x6000 s01_Led_type

Data type	Acces	Default	Range	Unit
UINT8	RW	0	-	-

Unused in this firmware version.

3.0.2 0x6001 s01_Led_count

Data type	Acces	Default	Range	Unit
UINT16	RW	100	-	-

Number of led in the current segment.

3.0.3 0x6010 s01_Value

Data type	Acces	Default	Range	Unit
INT16	RW,RPDO	0	-	-

Value of the effect. This value may be written directly or driven by Speed to control the effect over the time.

3.0.4 0x6011 s01_MinValue

Data type	Acces	Default	Range	Unit
INT16	RW,RPDO	0	-	-

Minimal value of Value when Speed is different of zero.

3.0.5 0x6012 s01_MaxValue

Data type	Acces	Default	Range	Unit
INT16	RW,RPDO	32767	-	-

Maximal value of Value when Speed is different of zero.

3.0.6 0x6013 s01_Speed

Data type	Acces	Default	Range	Unit
INT16	RW,RPDO	0	-	-

If zero, the speed controller has no effect. Anyway, the value of speed is added to value each millisecond.

3.0.7 0x6020 s01_Effect

Data type	Acces	Default	Range	Unit
UINT16	RW,RPDO	0	-	-

Value	Definition
0x0000	OFF
0x0001	FIX
0x0101	FLASH
0x0102	BREATHING
0x0300	FLOW
0x0301	FLOW_REVERSED

Table 1.2: Effects enum

Effect on the current strip segment.

3.0.8 0x6024 s01_Smooth

Data type	Acces	Default	Range	Unit
UINT16	RW,RPDO	0	-	-

Somme effects can be affected to smooth effect leds per leds instead of direct light up.

3.0.9 0x6025 s01_Dimmer

Data type	Acces	Default	Range	Unit
UINT8	RW,RPDO	255	-	-

Dimmer of complete segment from 0 (0%) to 255 (100%).

3.0.10 0x6030 s01_Color_01

Index	Name		Object type
0x6030	s01_Color_01		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT8	RW,RPDO	-	-

Subindex	Name
1	R
2	G
3	B
4	W

First color of effect.

3.0.11 0x6031 s01_Color_02

Index	Name		Object type
0x6031	s01_Color_02		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT8	RW,RPDO	-	-

Subindex	Name
1	R
2	G
3	B
4	W

Second color of effect.

3.0.12 0x6032 s01_Color_03

Index	Name		Object type
0x6032	s01_Color_03		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT8	RW,RPDO	-	-

Subindex	Name
1	R
2	G
3	B
4	W

Third color of effect.

4 Board configuration and status

4.0.1 0x2000 Board_voltage

Index	Name		Object type
0x2000	Board_voltage		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT16	RO,TPDO	-	0.01 V

Subindex	Name
1	Board_voltage_input

Input power board voltage in 0.01 V. This value is used for under / over-voltage detection. Useful to monitor battery voltage.

4.0.2 0x2001 Manufacture_date

Data type	Acces	Default	Range	Unit
USTRING	RO	YYYY/MM/DD hh:mm:ss	-	-

Date of board manufacturing.

4.0.3 0x2002 Calibration_date

Data type	Acces	Default	Range	Unit
VSTRING	RO	YYYY/MM/DD hh:mm:ss	-	-

Date of board calibration (if needed).

4.0.4 0x2003 Firmware_build_date

Data type	Acces	Default	Range	Unit
VSTRING	RO	-	-	-

Date and time of firmware build. In case of pre-release version, thanks to indicate this value to support team.

4.0.5 0x2004 Board_led

Data type	Acces	Default	Range	Unit
UINT8	RW	1	-	-

On board LEDs settings.

Value	Definition
0	Disables all status LED
1	Default LED display status

Table 1.3: Board LED options

4.0.6 0x2041 Board_voltage_config

Index	Name	Object type
0x2041	Board_voltage_config	RECORD

Subindex	Name	Data type
1	Undervoltage	UINT16
2	Overvoltage	UINT16

Over / under-voltage detection configuration. If the input board voltage fall below this under-voltage value, an under-voltage fault is generated. The same thing happens for over-voltage with dedicated parameter.

4.1 CPU status

4.1.1 0x2020 CPU_temperature

Index	Name		Object type
0x2020	CPU_temperature		ARRAY
Data type	**Acces**	**Range**	**Unit**
INT16	RO,TPDO	-	0.1 °C

Subindex	Name
1	CPU1_temperature

CPU temperature. Some boards can have multiple CPU with independent temperature sensor.

Note: The number of elements in the array may vary depending on the controller model.

4.1.2 0x2021 CPU_life_cycle

Index	Name		Object type
0x2021	CPU_life_cycle		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT16	RO,TPDO	-	-

Subindex	Name
1	CPU1_life_cycle

A life time counter, incremented each CPU cycle. Can be used to monitor CPU speed or if a secondary CPU is frozen.

Note: The number of elements in the array may vary depending on the controller model.

4.1.3 0x2022 CPU_error

Index	Name		Object type
0x2022	CPU_error		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT16	RO	-	-

Subindex	Name
1	CPU1_error

Future usage.

Note: The number of elements in the array may vary depending on the controller model.

4.1.4 0x2023 CPU_min_cycle_us

Index	Name		Object type
0x2023	CPU_min_cycle_us		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT16	RO	-	μs

Subindex	Name
1	CPU1_min_cycle_us

Future statistics usage.

Note: The number of elements in the array may vary depending on the controller model.

4.1.5 0x2024 CPU_max_cycle_us

Index	Name		Object type
0x2024	CPU_max_cycle_us		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT16	RO	-	μs

Subindex	Name
1	CPU1_max_cycle_us

Future statistics usage.

Note: The number of elements in the array may vary depending on the controller model.

4.1.6 0x2025 CPU_mean_cycle_us

Index	Name		Object type
0x2025	CPU_mean_cycle_us		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT16	RO	-	μs

Subindex	Name
1	CPU1_mean_cycle_us

Future statistics usage.

Note: The number of elements in the array may vary depending on the controller model.

4.2 Communication configuration

4.2.1 0x2040 Communication_config

Index	Name	Object type
0x2040	Communication_config	RECORD

Subindex	Name	Data type
1	Node_ID	UINT8
2	Bit_rate	UINT8

0x2040.1 ***Node_ID***

Data type	Acces	Default	Range	Unit
UINT8	RW	0	-	-

CANOpen node id. To change the node id, set the new value in this object, store manufacturer parameters or all parameters and restart the node.

`0x2040.2` ***Bit_rate***

Data type	Acces	Default	Range	Unit
UINT8	RW	0	-	-

Value	Definition
0	Default speed, 1 Mbit/s
1	10 kbit/s
2	20 kbit/s
3	50 kbit/s
4	100 kbit/s
5	125 kbit/s
6	250 kbit/s
7	500 kbit/s
9	1 Mbit/s

Table 1.4: CAN bus bit rate options

To change the can speed, set the new value in this object, store manufacturer parameters or all parameters and restart the node.

800 kbit/s is not yet supported.

Chapter 2

CANOpen

1 Introduction

CANOpen is based on bus CAN (Controller Area Network) which is ISO-11898 standardized. CANOpen protocol is standardized by the CAN In Automation (CIA) under the name of CiA301.

1.1 CAN for CANOpen

CANOpen uses standard CAN frames to communicate on the bus. The frames are identified with an Id : CAN-ID. It is coded on 11 bits. The CAN-ID with the lowest value has priority if multiple frames are sent at the same time from different devices.

The simplified CANOpen frame:

11 bit	0	1	2	3	4	5	6	7
CAN-ID	Data							

Table 2.1: CAN frame

Note: A CAN message can contain a maximum of 8 bytes of useful data.

Note: The data is always sent on the bus in **little-endian format**.

1.2 CANOpen node

Each device, also called Node, is identified by a unique identifier in the network called: Node-Id. The Node-Id can take a value in the range [1-127].

All descriptions and functionalities of a Node are described in the EDS (Electronic Data Sheet) file.

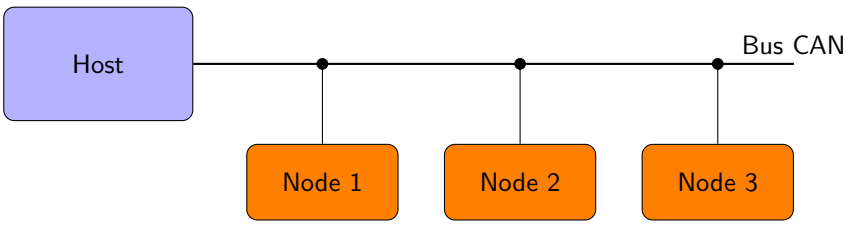


Table 2.2: Bus CAN

1.3 Communication

CANOpen protocol defines three communication protocol sequences :

- Master/Slave protocol
- Client/Server protocol

- Producer/Consumer protocol

1.3.1 Master / slave protocol

This protocol works on a network where only one CANOpen master is present for a specific functionality. And therefore the other CANOpen devices are slaves. The master sends a request and the slave responds.

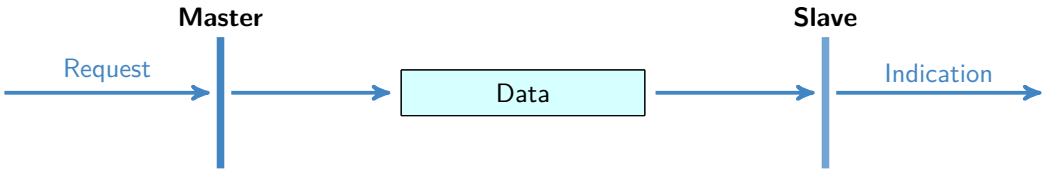


Figure 2.1: Unconfirmed master / slave protocol

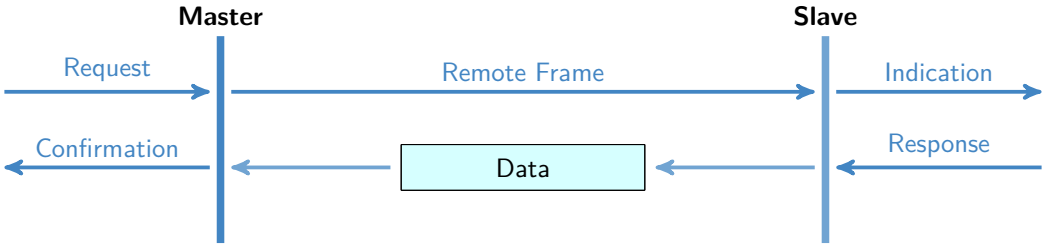


Figure 2.2: Confirmed master / slave protocol

1.3.2 Client / server protocol

This protocol is used between a client and a server. When the client makes a request (download / upload), the server triggers the processing of the request. The server responds to the request when the task is completed.

Figure 2.3: Client / server protocol

1.3.3 Producer / consumer protocol

This protocol works with a producer that sends a message that can be received by one or more devices on the network. The producer does not receive confirmation.

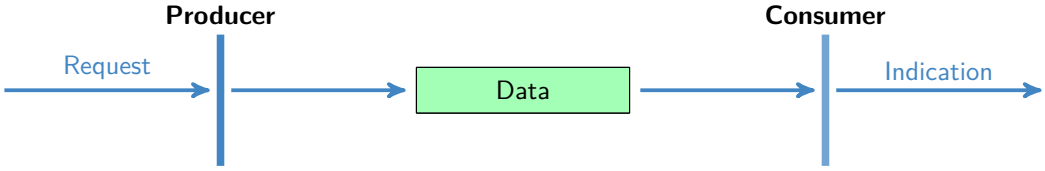


Figure 2.4: Push producer / consumer protocol

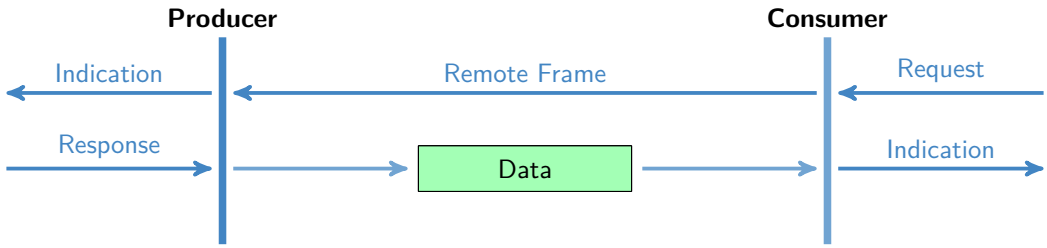


Figure 2.5: Pull producer / consumer protocol

2 CANOpen Services

Services provided by the CANOpen stack allow a standard communication between the devices on the network. These services allow communication object exchanges. There are 5 types of services:

- Network management services
  - Network management services (NMT)
  - Node guarding, network equipment monitoring
  - Heartbeat, network equipment monitoring
  - Boot-up
- Service Data Object (SDO), providing read and write access to the dictionary of objects
- Process Data Object (PDO), allowing to transmit data in real time process:
  - TPDO Transmit-PDO for use in data transmission
  - RPDO Receive-PDO for use in data reception
- SYNC, synchronization object used by PDO.
- Emergency (EMCY), emergency

Each type of message is defined by the allocation of the following CAN-IDs:

Services	Default CAN-ID	CAN-ID configurable
NMT	0x000	
SYNC	0x080	yes
EMCY	0x080 + Node-Id	yes
TPDO1	0x180 + Node-Id	yes
RPDO1	0x200 + Node-Id	yes
TPDO2	0x280 + Node-Id	yes
RPDO2	0x300 + Node-Id	yes
TPDO3	0x380 + Node-Id	yes
RPDO3	0x400 + Node-Id	yes
TPDO4	0x480 + Node-Id	yes
RPDO4	0x500 + Node-Id	yes
TSDO	0x580 + Node-Id	
RSDO	0x600 + Node-Id	
Boot-Up	0x700 + Node-Id	
Node-guarding and Heartbeat	0x700 + Node-Id	

Table 2.3: Default CAN-ID services

Some services can have a CAN-ID configurable by a communication object:

Services	Object
SYNC	0x1005
EMCY	0x1014
TPDOX	0x180X
RPDOX	0x140X

Table 2.4: Index of CAN-ID services

But the changing CAN-ID should not interfere with the following reserved CAN-IDs:

CAN-ID COB-ID	Description
0x000	NMT
0x000 0x07F	reserved
0x101 0x180	reserved
0x581 0x5FF	default SDO (tx)
0x601 0x67F	default SDO (rx)
0x6E0 0x6FF	reserved
0x701 0x77F	NMT Error Control
0x780 0x7FF	reserved

Table 2.5: Restricted CAN-ID

3 Object Dictionary (OD)

The object dictionary is a collection of all the data items which have an influence on the behavior of the application objects, the communication objects and the state machine used on the device. Each device on the network has its own object dictionary.

The object dictionary is divided into several areas:

Index range	Description
0x0000	Reserved
0x0001 to 0x025F	Data types
0x0260 to 0x0FFF	Reserved
0x1000 to 0x1FFF	Communication profile area
0x2000 to 0x5FFF	Manufacturer-specific profile area
0x6000 to 0x9FFF	Standardized profile area

Table 2.6: Object dictionary area

The **Communication profile area** contain the communication specific parameters. These objects are common to all CANOpen devices.

The **Standardized profile area** contain all data objects common to a profiles of CANOpen devices that may be read or written via the network. The objects from 6000 h to 9FFF h describe parameters and functionality.

The **Manufacturer-specific** profile area contains the objects for specific UniSwarm features.

3.1 Description of the object dictionary

The objects of the dictionary are described by several parameters. This description is materialized by an EDS file: Electronic Data Sheet. ASCII format respecting a strict syntax that can be used by the bus configuration software.

3.1.1 Index and sub-index

These form the unique identifier of an object in the objects dictionary in hexadecimal notation.

3.1.2 Object code

The object code denotes what kind of object is at a particular index within the objects dictionary. They can be one of the following:

Object name	Description	Code
NULL	An object with no data fields	0x00
DOMAIN	A large variable amount of data	0x02
DEFTYPE	A type definition for simple data type such as a Boolean, Unsigned16	0x05
DEFSTRUCT	Defines a new record type	0x06
VAR	A single value	0x07
ARRAY	A data area in which each entry is of the same data type.	0x08
RECORD	A data area that contains entries that are a combination of simple data types.	0x09

Table 2.7: Object code

3.1.3 Data type

The data type information indicates the data type of the object.

Type	Value	Size in byte
Boolean	0x0001	1
Integer8	0x0002	1
Integer16	0x0003	2
Integer24	0x0010	3
Integer32	0x0004	4
Integer40	0x0012	5
Integer48	0x0013	6
Integer56	0x0014	7
Integer64	0x0015	8
Unsigned8	0x0005	1
Unsigned16	0x0006	2
Unsigned32	0x0007	4
Unsigned24	0x0016	3
Unsigned40	0x0018	5
Unsigned48	0x0019	6
Unsigned56	0x001A	7
Unsigned64	0x001B	8
Real32	0x0008	4
Real64	0x0011	8
VISIBLE STRING	0x0009	Variable
OCTET STRING	0x000A	Variable
UNICODE STRING	0x000B	Variable
TIME OF DAY	0x000C	Variable
TIME DIFFERENCE	0x000D	Variable
Domain	0x000F	Variable

Table 2.8: Data types

Note: Data type with indices from 0x0001 to 0x0007, 0x0010, from 0x0012 to 0x0016 , and from 0x0018 to 0x001B may be mapped in order to define the appropriate space in the RPDO.

Note: Data type with indices from 0x0008 to 0x000F, 0x0011, from 0x0020 to 0x0023 shall not be mapped into RPDOs

3.1.4 Access usage

Access object:

- rw: read and write access
- wo: write only access
- ro: read only access
- const: read only access, value is constant

In addition, there are the access attributes for the PDOs:

- rww: read and write access and can be mapped on RPDO
- rwr: read and write access and can be mapped on TPDO

4 Network management services (NMT)

Network management (NMT) follows a master-slave structure. All devices are NMT slaves but the network must have a device master (device master, computer or other).

The service provides a tool to initiate, start, monitor, reset or stop the devices. Monitoring is made with the Node guarding and Heartbeat functionalities.

4.1 NMT Network management

The NMT master controls the state of each NMT slave. The state can be chosen among the following ones: Stopped, Pre-operational, Started, Initialization.

4.1.1 NMT state machine

The NMT state machine determines the behavior of the communication function unit.

- state **Initialization**
  - **Initializing**: the device enters this state after a power-on or an hardware/software reset.
  - **Application reset**: The object dictionary in Manufacturer-specific profile area, index range `0x2000` to `0xFFFF` is reset.
  - **Communication reset**: The object dictionary in Communication profile area, index range `0x1000` to `0x1FFF` is reset.
- state **Pre-operational**
- state **Started**
- state **Stopped**

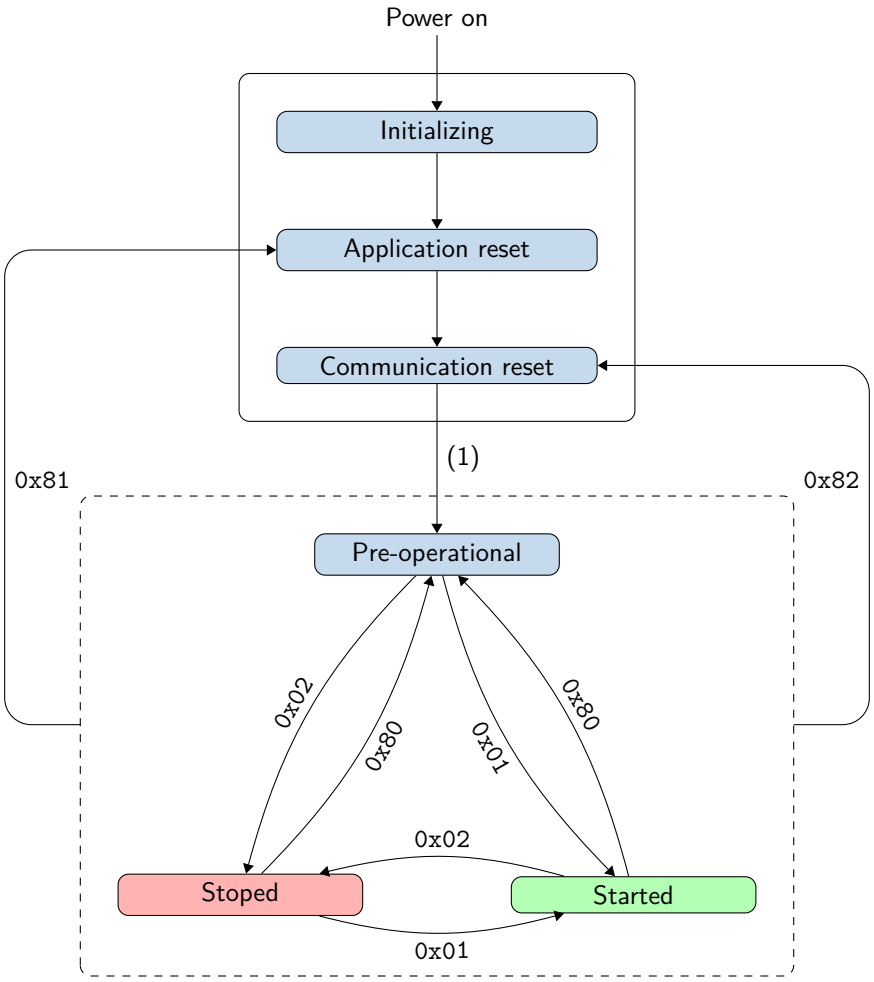
The following figure show the state diagram:

CAN-ID	Byte[0]	Byte[1]
0x000	Mode	Node-Id

Figure 2.7: NMT frame

Mode value	Description
0x01	Start node
0x02	Stop node
0x80	Enter in Pre-operational mode
0x81	Application reset
0x82	Communications reset

Figure 2.8: MODE values in NMT frame



(1) Automatic switch to Pre-operational and send Boot-up message

Figure 2.6: Different status of a CAN Open node

4.1.2 NMT frame

The NMT frame allow to change the state of device. An NMT frame have the CAN-ID 0x000 and a payload of two bytes. The first one is the mode and the second one the node id.

4.1.3 NMT States and Services

Authorized services according to the NMT state:

	Pre-operational	Started	Stopped
PDO		X	
SDO	X	X	
SYNC	X	X	
EMCY	X	X	
Node guarding	X	X	X
Heartbeat	X	X	X

Figure 2.9: NMT states and authorized services

4.2 Node Guarding - Heartbeat

Two services are available to detect an error on the CAN network: the **Node guarding** service and the **Heartbeat** service.

- **Node guarding**: the master sends a message periodically and each slave has to respond within a time limit.
- **Heartbeat**: each slave sends a message with his state without prior request from the master.

Note: The Heartbeat service has priority over the Node guarding service. Activation of the Heartbeat service results in the deactivation of the Node Guarding service.

4.2.1 Node guarding

This service monitors the status of devices on the bus and makes it possible to detect remote errors on the network.
The master and the slave monitor each other: the master cyclically requests the NMT status of the slave. In each response from the slave, the Toggle-bit (bit 7) is toggled.

Two monitoring functions are implemented:

- **Node guarding**: The master can react accordingly if these responses are not sent or if the slave always responds with the same bit Toggle.
- **Life guarding**: The slave monitors the reception of messages from the master, if the message is not sent within the allotted time, the Life Time, the slave triggers an Emergency (EMCY) message (with ode 0x8130) see table **??**.
  The Life time : *Life Time* = *0x100C Guard_time* x *0x100D Life_time_factor*.

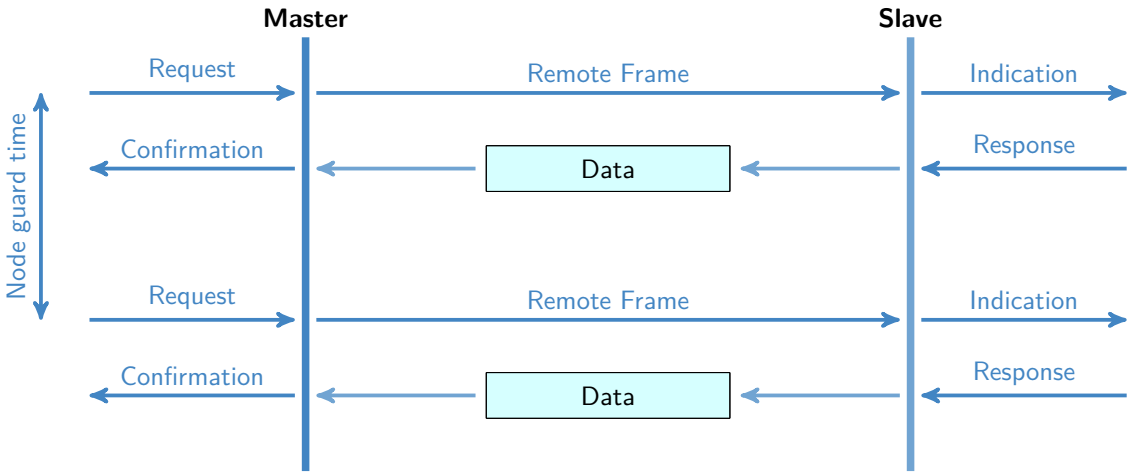
Note: This service is activated by setting value in object 0x100C Guard_time and 0x100D Life_time_factor other than zero.

Node guarding frame The node guarding service follows a master / slave architecture: the NMT master sends an RTR (Remote Transmission Request) message with CAN-ID 700 + node Id to a slave and the slave responds with an 8-bit message.

The response message is built with a Toggle-bit (bit 7) and the current NMT state of the slave in bits 6 to 0. For the first response and after a NMT reset, the toggle bit should be 0.

CAN-ID	Byte[0]
0x700 + Node-Id	0x00

Figure 2.10: Node guarding frame



- t: Toggle Bit
- Node state:
  - 4: Stopped
  - 5: Started
  - 27: Pre-operational

Figure 2.11: Node guarding protocol

4.2.2 Heartbeat

The Heartbeat works in Producer/Consumer mode with one producer and 0 minimum consumer.

The producer heartbeat send a heartbeat message periodically, with the time between two messages defined by the "Producer heartbeat time" object.

The consumer check if it received a message in the time defined by the object "Consumer heartbeat time".

Note: This service is activated by setting the producer heartbeat time object in object 0x1017 Producer_heartbeat_time to a value other than zero.

CAN-ID	Byte[0]
0x700 + Node-Id	0x00

Figure 2.12: Heartbeat frame

CAN-ID	Byte[0]
0x700 + Node-Id	0x00

Figure 2.14: Boot-up frame

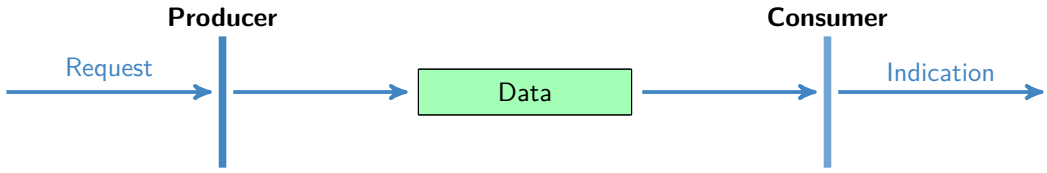
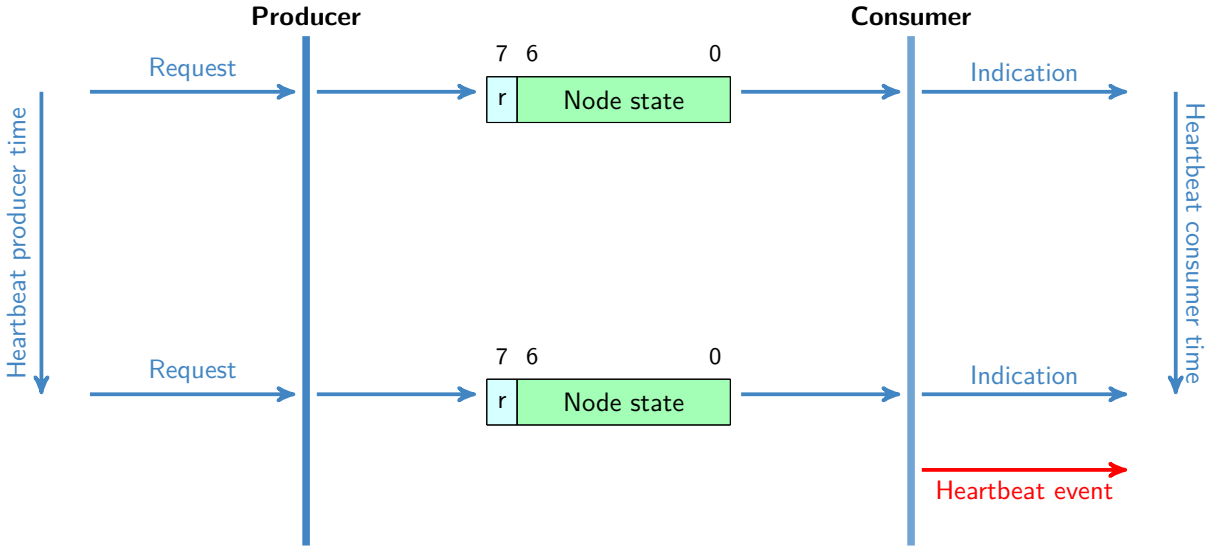


Figure 2.15: Boot-up protocol



- r: reserved (always 0)
- Node state:
  - 0: Boot-Up
  - 4: Stopped
  - 5: Operational
  - 127: Pre-operational

Figure 2.13: Heartbeat protocol

4.3 Boot-up

After power-up, the slave sends a Boot-up message to indicate that the Initializing phase is complete.

5 Emergency (EMCY)

The EMCY service is used to transmit application faults associated with each station. When a fault is detected the device send a EMCY message with Emergency error code, Error register and error code (optional).

CAN-ID	Byte[0]
0x080 + Node-Id	...

Figure 2.16: EMCY frame

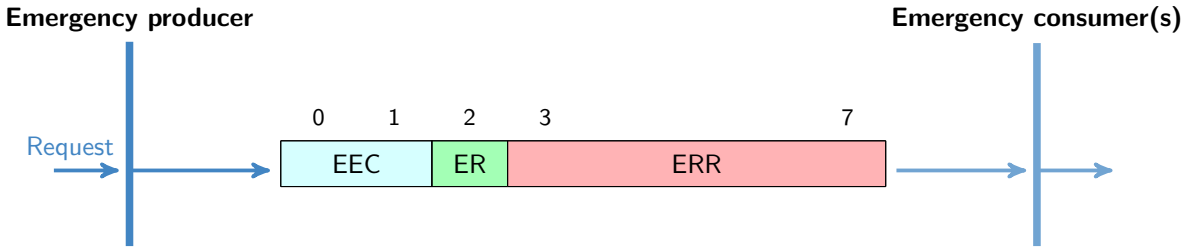


Figure 2.17: Emergency protocol with byte[0]

- EEC: Emergency error code
- ER: Error register (see object 0x1001 *Error_register*)
- ERR: Manufacturer-specific error code (Optional)

EEC	Description
0x0000	**Error reset or no error**
0x1000	**Generic error**
0x2000	**Current generic error**
0x2100	Current, CANOpen device input side generic
0x2200	Current inside the CANOpen device generic
0x2300	Current, CANOpen device output side generic
0x3000	**Voltage generic error**
0x3100	Mains voltage generic
0x3200	Voltage inside the CANOpen device generic
0x3300	Output voltage generic
0x4000	**Temperature generic error**
0x4100	Ambient temperature generic
0x4200	Device temperature generic
0x5000	**CANOpen device hardware generic error**
0x6000	**CANOpen device software generic error**
0x6100	Internal software generic
0x6200	User software generic
0x6300	Data set generic
0x7000	**Additional modules generic error**
0x8000	**Monitoring generic error**
0x8100	Communication generic
0x8110	CAN overrun (objects lost)
0x8120	CAN in error passive mode
0x8130	Life guard error or heartbeat error
0x8140	Recovered from bus off
0x8150	CAN-ID collision
0x8200	Protocol error - generic
0x8210	PDO not processed due to length error
0x8220	PDO length exceeded
0x8230	DAM MPDO not processed, destination object not available
0x8240	Unexpected SYNC data length
0x8250	RPDO timeout
0x9000	**External error generic error**
0xF000	**Additional functions generic error**
0xFF00	**Device specific generic error**

Table 2.9: General emergency error codes
Other specific EEC codes are defined in the operating modes

6 Error codes and error behavior

6.1 Definition of parameters

6.1.1 0x1001 Error_register

Data type	Acces	Default	Range	Unit
UINT8	RO,TPDO	-	-	-

Bit	Description
0	Generic error
1	Current
2	Voltage
3	Temperature
4	Communication error (overrun, error state)
5	Device profile specific
6	reserved (always 0 b)
7	manufacturer-specific

Table 2.10: Error code register of object 0x1001

6.1.2 0x1029 Error_behaviour

Index	Name		Object type
0x1029	Error_behaviour		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT8	RW	[0 - 2]	-

Subindex	Name
1	Communication_error

Sets the behavior in the event of a serious device failure in the Operational NMT state. By default, the device automatically enter the Pre-operational NMT state.

Failures include the following communication errors:

- CAN interface bus stop conditions
- NodeGuarding "time out"
- Heartbeat "time out"
- Serious errors can also be caused by internal device failures.

Bit	Description
0	Change to NMT state Pre-operational (default)
1	No change of the NMT state
2	Change to NMT state Stopped

Table 2.11: Error class values of 0x1029.1

7 Service Data Object (SDO)

This service provides access to the device object dictionary by an Index and Sub-index without time constraints in writing or reading.

Each network device is an SDO server, the one that holds the OD. The client refers to the node requesting to read or write an object value in the server's object dictionary.

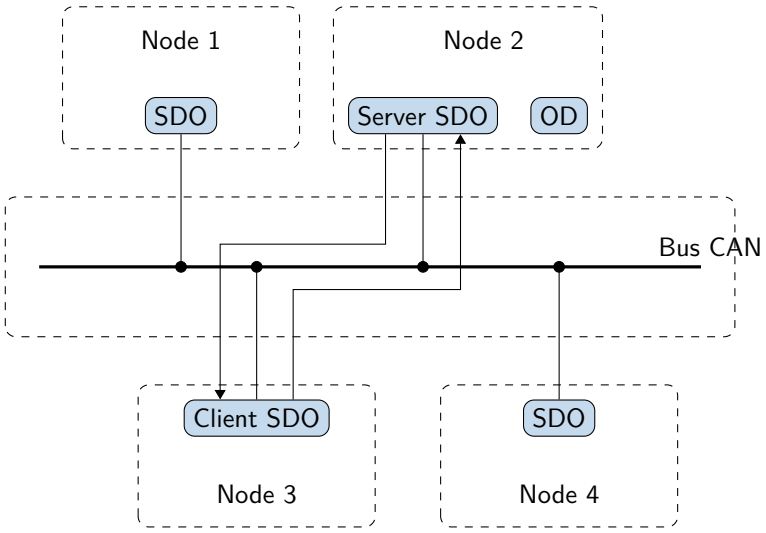


Figure 2.18: Server / client SDO

7.1 SDO message

CAN-ID	Byte[0-7]
CAN-ID+Node-Id	...

Figure 2.19: SDO frame

- Cmd: Command
- Index: Index of object
- Subindex: Subindex of object

Figure 2.20: Expedited SDO upload frame

7.2 Expedited transfer

This mode of communication is used to write or read data in object. The object size must be inferior or equal to 4 bytes. An answer is expected after each request, either with data, with a confirmation or with an error message.

7.2.1 SDO reading

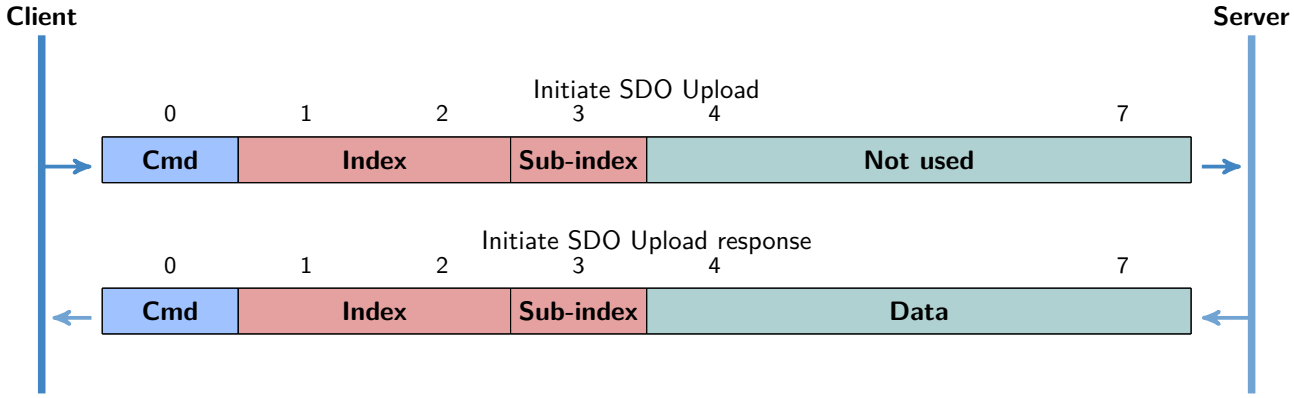


Figure 2.21: Expedited SDO Upload protocol

Cmd	Description
0x40	Upload/Reading request
0x4F	Upload/Reading response for data of size of 1 byte
0x4B	Upload/Reading response for data of size of 2 byte
0x47	Upload/Reading response for data of size of 3 byte
0x43	Upload/Reading response for data of size of 4 byte
0x2F	Download/Writing for data of size of 1 byte
0x2B	Download/Writing for data of size of 2 byte
0x27	Download/Writing for data of size of 3 byte
0x23	Download/Writing for data of size of 4 byte
0x60	Download/Writing response

Figure 2.22: List of commands (cmd)

Initiate SDO Upload/Reading request

0	1		2	3	4			7
Cmd	**Index**			**Sub-index**	**Data**			
0x40	lsb		msb	Sub-index	0	0	0	0

Initiate SDO Upload/Reading response

Read response for a 1 byte data:

0	1		2	3	4			7
Cmd	**Index**			**Sub-index**	**Data**			
0x4F	lsb		msb	Sub-index	data	0	0	0

Read response for a 2 bytes data:

0	1		2	3	4			7
Cmd	**Index**			**Sub-index**	**Data**			
0x4B	lsb		msb	Sub-index	lsb	msb	0	0

Read response for a 3 bytes data:

0	1	2	3	4			7
Cmd	**Index**		**Sub-index**	**Data**			
0x47	lsb	msb	Sub-index	lsb	···	msb	0

Read response for a 4 bytes data:

0	1	2	3	4			7
Cmd	**Index**		**Sub-index**	**Data**			
0x43	lsb	msb	Sub-index	lsb	···	···	msb

7.2.2 SDO Download/Writing

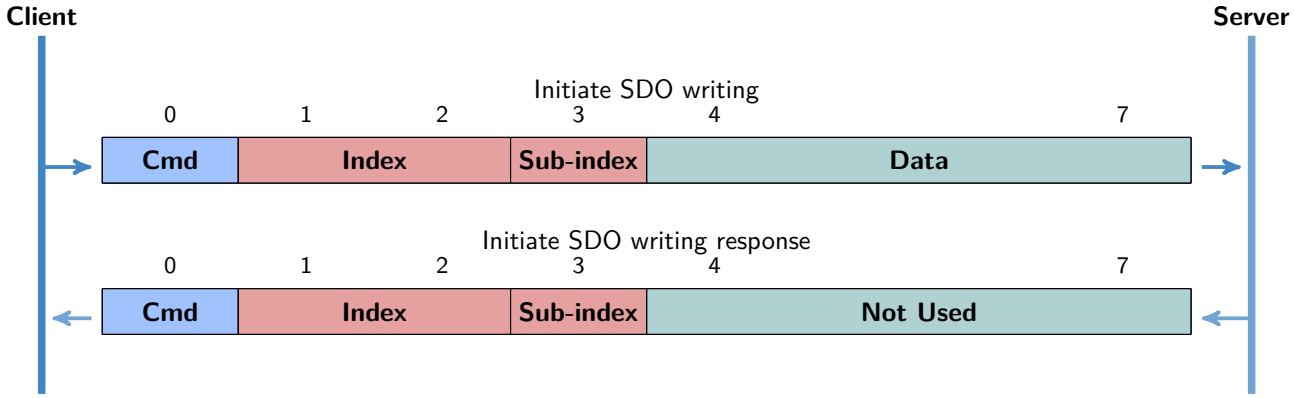


Figure 2.23: Expedited SDO Download protocol

Initiate SDO download / writing

Write request for 1 byte data:

0	1	2	3	4			7
Cmd	**Index**		**Sub-index**	**Data**			
0x2F	lsb	msb	Sub-index	data	0	0	0

Write request for 2 bytes data:

0	1	2	3	4			7
Cmd	**Index**		**Sub-index**	**Data**			
0x2B	lsb	msb	Sub-index	lsb	msb	0	0

Write request for 3 bytes data:

0	1	2	3	4			7
Cmd	**Index**		**Sub-index**	**Data**			
0x27	lsb	msb	Sub-index	lsb	···	msb	0

Write request for 4 bytes data:

0	1	2	3	4			7
Cmd	**Index**		**Sub-index**	**Data**			
0x23	lsb	msb	Sub-index	lsb	···	···	msb

Initiate SDO download / writing response

0	1	2	3	4			7
Cmd	**Index**		**Sub-index**	**Data**			
0x60	lsb	msb	Sub-index	0	0	0	0

7.3 SDO abort transfer

Error response:

0	1	2	3	4			7
Cmd	**Index**		**Sub-index**	**SDO abort codes**			
0x80	lsb	msb	Sub-index	lsb	···	···	msb

7.4 SDO abort codes

Error codes	Description
0x05030000	Toggle bit not alternated
0x05040000	SDO protocol timed out
0x05040001	Client/server command specifier not valid or unknown
0x05040002	Invalid block size (block mode only)
0x05040003	Invalid sequence number (block mode only)
0x05040004	CRC error (block mode only)
0x05040005	Out of memory
0x06010000	Unsupported access to an object
0x06010001	Attempt to read a write only object
0x06010002	Attempt to write a read only object
0x06020000	Object does not exist in the object dictionary
0x06040041	Object cannot be mapped to the PDO
0x06040042	The number and length of the objects to be mapped would exceed PDO length
0x06040043	General parameter incompatibility reason
0x06040047	General internal incompatibility in the device
0x06060000	Access failed due to an hardware error
0x06070010	Data type does not match, length of service parameter does not match
0x06070012	Data type does not match, length of service parameter too high
0x06070013	Data type does not match, length of service parameter too low
0x06090011	Sub-index does not exist
0x06090030	Invalid value for parameter (download only)
0x06090031	Value of parameter written too high (download only)
0x06090032	Value of parameter written too low (download only)
0x06090036	Maximum value is less than minimum value
0x060A0023	Resource not available: SDO connection
0x08000000	General error
0x08000020	Data cannot be transferred or stored to the application
0x08000021	Data cannot be transferred or stored to the application because of local control
0x08000022	Data cannot be transferred or stored to the application because of the present device state
0x08000023	Object dictionary dynamic generation fails or no object dictionary is present
0x08000024	No data available

Table 2.12: SDO error codes

8 Process Data Object (PDO)

The purpose of "Process Data Objects (PDO)" is to provide a transfer of data in real time during the operation of controller. This service is performed without protocol overload or confirmation. The size of a PDO frame is variable and

RPDO	CAN-ID COB-ID	Object Index	
		Index RPDO communication	Index RPDO mapping
RPDO1	0x200 + Node-Id	0x1400	0x1600
RPDO2	0x300 + Node-Id	0x1401	0x1601
RPDO3	0x400 + Node-Id	0x1402	0x1602
RPDO4	0x500 + Node-Id	0x1403	0x1603

Table 2.13: Index used of RPDOs

depends on the size of object.

The PDO provides an interface to the application objects in the object dictionary. Data type and mapping of object is determined by the corresponding PDO mapping structure in the object dictionary.

The PDO configuration process (PDO Mapping) allows to configure the number of objects in a PDO. This process uses the SDO service.

There are two types of PDO, the Transmit-PDO (TPDO) for use in data transmission and the Receive-PDO (RPDO) for use in data reception. The data transmission via the PDO service operates according to a producer / consumer relationship: RPDOs are frames received from the master or others nodes. TPDOs are frames transmitted to others.

PDOs are described by the PDO communication parameter and the PDO mapping parameter.

Note: A node can have a maximum of four TPDOs and four RPDOs.

There is an index couple for each PDO: the columns "Index RPDO communication" and "Index RPDO mapping" provide the indexes of specials objects used to read or modify the parameters of communication objects via an SDO object:

TPDO	CAN-ID COB-ID	Object Index	
		Index TPDO communication	Index TPDO mapping
TPDO1	0x180 + Node-Id	0x1800	0x1A00
TPDO2	0x280 + Node-Id	0x1801	0x1A01
TPDO3	0x380 + Node-Id	0x1802	0x1A02
TPDO4	0x480 + Node-Id	0x1803	0x1A03

Table 2.14: Index used of TPDOs

8.1 PDO message

There is two ways to transmit a PDO message:

- Synchronous transmission: the object are synchronized on SYNC.
- Event-driven transmission

This PDO is only activated if the status of CANOpen is "Started". It is necessary to activate the PDOs, for that the "valid" bit of CAN-ID must be to set to 0x0.

Example:
to activate or deactivate the TPDO1, the object with the index 0x1800 and sub-index 0x1 is used:

- Deactivate TPDO1: set sub-index 1 to 0x80000181
- Activate TPDO1: set sub-index 1 to 0x00000181

The value 0x181 is the id of TPDO1.

After each SYNC, two things happen in this order:

- for TPDOs: the slaves samples and copy the data into TPDOs which are then sent on the bus.
- for RPDOs: the previous received RPDO data from master is copied in the objects database and made available to the application.

8.2 SYNC

The master in a CAN Open network sends a unique sync frame for all the nodes. At reception, the nodes transmit theirs TPDOs and apply their RPDOs. A SYNC frame has the ID 0x080 and does not contain a payload.

CAN-ID	Data
0x080	-

Figure 2.24: SYNC frame

The frequency of SYNC frame can be variable, but is always sent by the master.

8.3 PDO dynamic mapping

General procedure:
The following procedure shall be used for re-mapping, which may take place during the NMT state Pre-operational and during the NMT state Operational, if supported:

1. Deactivate PDO: setting the valid bit to 0x1 of sub-index 0x01 of the PDO communication parameter.
2. Disable mapping: setting sub-index 0x00 to 0.
3. Modify mapping: changing the values of corresponding sub-indexes.
4. Enable mapping: setting sub-index 0x00 to the number of mapped objects.
5. Activate PDO: setting the valid bit to 0 of sub-index 0x01 of PDO communication parameter.

8.4 PDO parameter objects

8.4.1 0x1400 *RPDO_parameter_1*
0x1401 *RPDO_parameter_2*
0x1402 *RPDO_parameter_3*
0x1403 *RPDO_parameter_4*

Index	Name	Object type
0x1400	RPDO_parameter_1	RECORD

Subindex	Name	Data type
1	COB_ID	UINT32
2	Transmission_type	UINT8
3	Inhibit_time	UINT16
5	Event_timer	UINT16

The PDO parameter describes the communication abilities of the PDO.

0x1400.1 ***COB_ID***

Data type	Acces	Default	Range	Unit
UINT32	RW	512	[0x00000080 - 0xFFFFFFFF]	-

Contains the CAN-ID (COB-ID) of RPDO

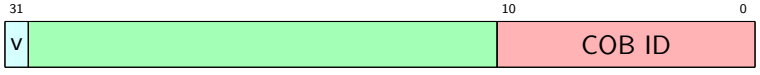


Table 2.15: COB ID

- **v**: The bit **valid** select which RPDOs are used in the NMT state Operational. PDOs can be fully configured but not used, and therefore set to "not valid" .

Value	Description
0x00	PDO exists / is valid
0x01	PDO does not exist / is not valid

Figure 2.25: Description of bit 31: v

- **COB ID CAN-ID** of RPDO

`0x1400.2` ***Transmission_type***

Data type	Acces	Default	Range	Unit
UINT8	RW	1	-	-

The transmission of PDO depends on the configuration of Transmission Types parameters which can be:

Value	Description
0x00 to 0xF0	Synchronous
0xFE	Event

Table 2.16: Description of RPDO transmission type

- **Synchronous**: the synchronous transmission type means that the data is transmitted immediately, but it is applied when SYNC is received. The SYNC service provides a data synchronization signal over the network.
- **Event**: the Event-driven transmission type means that the PDO may be received at any time and that the data is applied immediately after reception.

`0x1400.3` ***Inhibit_time***

Data type	Acces	Default	Range	Unit
UINT16	RW	0	-	0.1 ms

The value is defined as multiple of 100 ţs. The value of 0 shall disable the inhibit time. The value shall not be changed while the PDO exists.

`0x1400.5` ***Event_timer***

Data type	Acces	Default	Range	Unit
UINT16	RW	0	-	ms

The value is defined as multiple of 1 ms. The value of 0 shall disable the event-timer. The RPDO use the time for deadline monitoring. The deadline monitoring is activated within the next reception of an RPDO after configuring the event-timer. A timeout results in an indication to the local application.

8.4.2 `0x1600` ***RPDO_mapping_1*** `0x1601` ***RPDO_mapping_2*** `0x1602` ***RPDO_mapping_3*** `0x1603` ***RPDO_mapping_4***

Index	Name	Object type
0x1600	RPDO_mapping_1	RECORD

Subindex	Name	Data type
0	Number_of_entries	UINT8
1	PDO_mapping_entry_1	UINT32
2	PDO_mapping_entry_2	UINT32
3	PDO_mapping_entry_3	UINT32
4	PDO_mapping_entry_4	UINT32
5	PDO_mapping_entry_5	UINT32
6	PDO_mapping_entry_6	UINT32
7	PDO_mapping_entry_7	UINT32
8	PDO_mapping_entry_8	UINT32

The PDO mapping parameter contains information about the content of PDO.

`0x1600.0` ***Number_of_entries***

Data type	Acces	Default	Range	Unit
UINT8	RW	1	[0 - 8]	-

The number of valid object entries within the mapping record. If it is equal to 0, the mapping is disabled.

`0x1600.1` ***PDO_mapping_entry_1***

Data type	Acces	Default	Range	Unit
UINT32	RW	1614807056	-	-

The information of mapped application objects. The object describes content of PDO by index, sub-index and length of the mapped object.

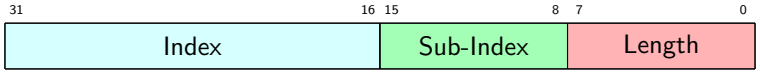


Figure 2.26: Structure of PDO mapping

Note: The MSB is first.

Note: The length is the length of object in bits.

8.4.3 `0x1800` ***TPDO_parameter_1*** `0x1801` ***TPDO_parameter_2*** `0x1802` ***TPDO_parameter_3*** `0x1803` ***TPDO_parameter_4***

Index	Name	Object type
0x1800	TPDO_parameter_1	RECORD

Subindex	Name	Data type
1	COB_ID	UINT32
2	Transmission_type	UINT8
3	Inhibit_time	UINT16
5	Event_timer	UINT16
6	SYNC_start_value	UINT8

The PDO parameter describes the communication abilities of the PDO.

`0x1800.1` ***COB_ID***

Data type	Acces	Default	Range	Unit
UINT32	RW	384	[0x00000080 - 0xFFFFFFFF]	-

Contains the CAN-ID (COB-ID) of the TPDO

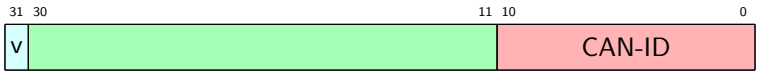


Table 2.17: COB ID

- **v**: The bit **valid** select which TPDOs are used in the NMT state Operational. PDOs can be fully configured but not used, and therefore set to "not valid" .

Value	Description
0x00	PDO exists / is valid
0x01	PDO does not exist / is not valid

Table 2.18: Description of bit 31: v

- **COB ID CAN-ID** of TPDO

`0x1800.2` ***Transmission_type***

Data type	Acces	Default	Range	Unit
UINT8	RW	1	-	-

The transmission of the PDO depends on the configuration of Transmission Types parameters which can be:

Value	Description
`0x00`	Synchronous (acyclic)
`0x01` to `0xF0`	Synchronous cyclic every N SYNC
`0xFC`	RTR-only (synchronous)
`0xFD`	RTR-only (event-driven)
`0xFE`	Event-driven

Table 2.19: Description of TPDO transmission type

- **Synchronous acyclic**: On internal event, the sampling will start and will transmit after the next SYNC.
- **Synchronous**: the TPDO is transmitted after each SYNC received. The sampling of the data will start and will transmit on reception of each the SYNC received.
- **RTR-only (synchronous)**: On RTR (Remote Transmission Request) received, the sampling will start and will transmit after the next SYNC.
- **RTR-only (event-driven)**: On RTR (Remote Transmission Request) received, the sampling start and transmit immediately.
- **Event-driven**: sampling may be transmitted at any time when sampling is different of previous.

`0x1800.3` ***Inhibit_time***

Data type	Acces	Default	Range	Unit
UINT16	RW	0	-	0.1 ms

It's a minimum interval time for PDO transmission. This parameter is available only for the transmission type `0xFE`. The value is defined as multiple of 100 μs. The value of 0 shall disable the inhibit time. The value shall not be changed while the PDO exists.

`0x1800.5` ***Event_timer***

Data type	Acces	Default	Range	Unit
UINT16	RW	0	-	ms

TPDO frame sending timer. Independent from SYNC. The value is defined as multiple of 1 ms. The value of 0 shall disable the event-timer.

`0x1800.6` ***SYNC_start_value***

Data type	Acces	Default	Range	Unit
UINT8	RW	0	-	-

The SYNC message of which the counter value equals the SYNC Start value is be regarded as the first received SYNC message. The value of 0 shall disable the SYNC Start value.

8.4.4 `0x1A00` *TPDO_mapping_1*
`0x1A01` *TPDO_mapping_2*
`0x1A02` *TPDO_mapping_3*
`0x1A03` *TPDO_mapping_4*

Index	Name	Object type
`0x1A00`	TPDO_mapping_1	RECORD

Subindex	Name	Data type
0	Number_of_entries	UINT8
1	PDO_mapping_entry_1	UINT32
2	PDO_mapping_entry_2	UINT32
3	PDO_mapping_entry_3	UINT32
4	PDO_mapping_entry_4	UINT32
5	PDO_mapping_entry_5	UINT32
6	PDO_mapping_entry_6	UINT32
7	PDO_mapping_entry_7	UINT32
8	PDO_mapping_entry_8	UINT32

The PDO mapping parameter contains information about the content of PDO.

`0x1A00.0` ***Number_of_entries***

Data type	Acces	Default	Range	Unit
UINT8	RW	1	[0 - 8]	-

The number of valid object entries within the mapping record. If it is equal to 0, the mapping is disabled

`0x1A00.1` ***PDO_mapping_entry_1***

Data type	Acces	Default	Range	Unit
UINT32	RW	1614872592	-	-

The information of mapped application objects. The object describes content of PDO by the index, sub-index and length of the mapped object.

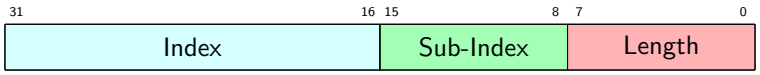


Figure 2.27: Structure of PDO mapping

Note: The MSB is first.

Note: The length is the length of object in bits.

9 Object description

9.1 Communication Profile Area object

9.1.1 `0x1000` Device_type

Data type	Acces	Default	Range	Unit
UINT32	RO	402	-	-

This object provides information about the device type. It is composed of two fields, the device profile and additional information, both on 16 bits.

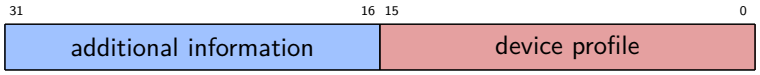


Table 2.20: Frame of Device Type

Value	Device profile	Description
0x191	CiA 401 profile	Inputs / outputs
0x192	CiA 402 profile	Motion control

Table 2.21: Description of device type

9.1.2 0x1008 Manufacturer_device_name

Data type	Acces	Default	Range	Unit
VSTRING	RO	UMC1BDS32 motion controller	-	-

Provides the name of device.

9.1.3 0x1009 Manufacturer_hardware_version

Data type	Acces	Default	Range	Unit
USTRING	RO	vXX.XX.XXX	-	-

Provides the manufacturer hardware version description.

9.1.4 0x100A Manufacturer_software_version

Data type	Acces	Default	Range	Unit
VSTRING	RO	-	-	-

Provides the manufacturer software version description.

9.1.5 0x100C Guard_time

Data type	Acces	Default	Range	Unit
UINT16	RW	0	[0 - 65535]	ms

The life time factor multiplied with the guard time gives the life time for the life guarding protocol. The value of 0x00 disable the life guarding.

9.1.6 0x100D Life_time_factor

Data type	Acces	Default	Range	Unit
UINT8	RW	0	[0 - 255]	-

The life time factor multiplied with the guard time gives the life time for the life guarding protocol. The value of 0x00 disable the life guarding.

9.1.7 0x1017 Producer_heartbeat_time

Data type	Acces	Default	Range	Unit
UINT16	RW	0	[0 - 65535]	ms

The producer heartbeat time indicate the configured cycle time of heartbeat.

9.1.8 0x1010 Store_parameters

Index	Name		Object type
0x1010	Store_parameters		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT32	RW	-	-

Subindex	Name
1	Save_all_parameters
2	Save_communication_parameters
3	Save_standardized_parameters
4	Save_manufacturer_parameters

This object stores the value of the parameters according to the communication profile, manufacturer profile and standardized profile or all profiles. The profile areas are defined to Object Dictionary (OD).

This functionality can only work in state PreOp if this is not the case, the device respond with the SDO abort transfer service (SDO abort code: 0x08000021).

Signature To start backup of parameters, a specific signature is required to avoiding wrong manipulation. Specific signature is written in appropriate subindex.

The signature is "save":

3	2	1	0
e	v	a	s
0x65	0x76	0x61	0x73

Figure 2.28: Signature of store

Upon receipt of the correct signature in the appropriate subindex, the device restores the default settings and then confirms the SDO transmission (SDO download initiation response).

If an erroneous signature is written, the device refuses to store the defaults and responds with the SDO abort transfer service (SDO abort code: 0x08000020).

9.1.9 0x1011 Restore_default_parameters

Index	Name		Object type
0x1011	Restore_default_parameters		ARRAY
Data type	**Acces**	**Range**	**Unit**
UINT32	RW	-	-

Subindex	Name
1	Restore_all_factory_parameters
2	Restore_factory_communication_parameters
3	Restore_factory_standardized_parameters
4	Restore_factory_manufacturer_parameters
5	Restore_all_saved_parameters
6	Restore_saved_communication_parameters
7	Restore_saved_standardized_parameters
8	Restore_saved_manufacturer_parameters

This object restores the factory or saved values of the parameters according to the communication profile, manufacturer profile and standardized profile or all profiles. Profile areas are defined to Object Dictionary (OD).

This functionality can only work in State PreOp if this is not the case, the device respond with the SDO abort transfer service (SDO abort code: 0x08000021).

Signature To start the restoration of parameters, a specific signature is required to avoid wrong manipulation. Specific signature is written in appropriate subindex.

The signature is "load":

3	2	1	0
d	a	o	l
0x64	0x61	0x6F	0x6C

Figure 2.29: Signature of restore

Upon receipt of the correct signature in the appropriate subindex, the device restore the default settings and then confirm the SDO transmission (SDO download initiation response).

If an erroneous signature is written, the device refuse to restore the defaults and respond with the SDO abort transfer service (SDO abort code: 0x08000020).

Automatic restore This feature determines an automatic restore after a NMT service reset node, NMT service reset communication or power cycled. This functionality is configured with a command (table below) written in the appropriate subindex.

Value	Description
0x00000000	Device does not restore settings automatically
0x00000001	Device restores settings automatically

Table 2.22: Automatic restore values

9.1.10 0x1018 Identity_object

Index	Name	Object type
0x1018	Identity_object	RECORD

Subindex	Name	Data type
1	Vendor_ID	UINT32
2	Product_code	UINT32
3	Revision_number	UINT32
4	Serial_number	UINT32

Provide general identification information of the CANOpen device.

0x1018.1 *Vendor_ID*

Data type	Acces	Default	Range	Unit
UINT32	RO	1186	-	-

Unique vendor value of a CANOpen device.

0x1018.2 *Product_code*

Data type	Acces	Default	Range	Unit
UINT32	RO	4097	-	-

The unique value that identifies a specific type of CANOpen devices.

0x1018.3 *Revision_number*

Data type	Acces	Default	Range	Unit
UINT32	RO	16777219	-	-

The major revision number and the minor revision number of the revision of the CANOpen device.

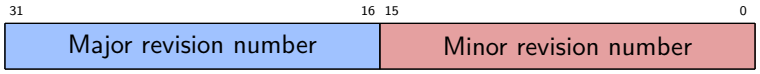


Table 2.23: Revision number

0x1018.4 *Serial_number*

Data type	Acces	Default	Range	Unit
UINT32	RO	0	-	-

The serial number identify a CANOpen device within a product group and a specific revision.

Appendix A

Firmware version history

Version	Date	Change
1.0.0	2022/01/20	Initial public version
1.0.1	2023/08/01	Fixed effects and speed controller

Appendix B

Datasheet revision history

Revision	Date	Change
A	2023/07/11	Initial public revision