ifm electronic

CE

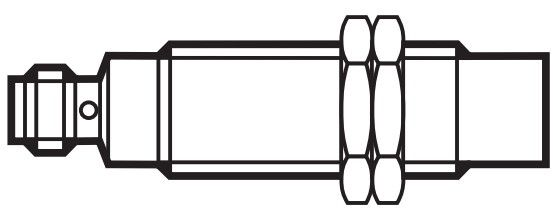
Operating instructions RFID read/write head with CANopen interface

UK

efector190

DTM425

706394 / 00 02 / 2015



Content

1 Preliminary note

This document applies to devices of the type "RFID read/write head with CANopen interface" (art. no.: DTM425). These instructions are part of the device.

This document is intended for specialists. These specialists are people who are qualified by their appropriate training and their experience to see risks and to avoid possible hazards that may be caused during operation or maintenance of the device. The document contains information about the correct handling of the device.

Read this document before use to familiarise yourself with operating conditions, installation and operation. Keep this document during the entire duration of use of the device.

Adhere to the safety instructions.

1.1 Symbols used

► Instructions
> Reaction, result
[…] Designation of keys, buttons or indications
→ Cross-reference


Important note
Non-compliance may result in malfunction or interference


Information
Supplementary note

2 Safety instructions

2.1 General

These instructions are an integral part of the device. They contain texts and figures concerning the correct handling of the device and must be read before installation or use.

Observe the operating instructions. Non-observance of the instructions, operation which is not in accordance with use as prescribed below, wrong installation or incorrect handling can seriously affect the safety of operators and machinery.

2.2 Target group

These instructions are intended for authorised persons according to the EMC and low-voltage directives. The device must only be installed, connected and put into operation by a qualified electrician.

2.3 Electrical connection

Disconnect the unit externally before handling it.

The connection pins may only be supplied with the signals indicated in the technical data and/or on the device label and only the approved accessories of ifm may be connected.

2.4 Tampering with the device

In case of malfunctions or uncertainties please contact the manufacturer. Any tampering with the device can seriously affect the safety of operators and machinery. This is not permitted and leads to the exclusion of any liability and warranty claims.

3 General information

3.1 CANopen technology

The CANopen communication profile is based on the CAN Application Layer (CAL) specification of the CiA organisation. CANopen is considered as a robust fieldbus with highly flexible configuration options. It is used in many various applications which are based on different application profiles. CANopen comprises a concept to configure and communicate real-time data using synchronous and asynchronous messages. Four message types (objects) are distinguished:

1. Administration messages (layer management, network management and identifier distribution)
2. Service Data Objects (SDO)
3. Process Data Objects (PDO)
4. Predefined Objects (emergency)

For further information please refer to the CiA-CAN specification (CiA 301 - CANopen).

3.2 References

http://www.can-cia.org

CAN Application Layer, DS 201 …207	CiA
LSS profile	DS305 CiA
CAL-based communication profile	DS 301 CiA
CAN specification version 2.0 A	Robert Bosch GmbH
CANary CAN controller	Atmel

4 Functions and features

The DTM425 RFID read/write head is used for reading and describing RFID tags. The read/write head is configured and data is exchanged via the integrated CANopen interface.

Typical applications are for example the identification of interchangeable tools and attachments on mobile machines.

5 Installation

5.1 General installation instructions

Observe the separate mounting instructions of DTM425.

When mounting several read/write heads adhere to the minimum distances between the systems.

The immediate vicinity of powerful HF emission sources such as welding transformers or converters can affect operation of the read/write heads.

5.2 Notes on the tag installation

Installation of the tags in or on metal reduces the read and write distances.

The orientation of the read/write head antenna axis must correspond with the axis of the tag coil.

5.3 Avoiding interference

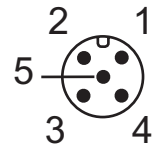
The device generates a modulated electrical field with a frequency of 13.56 MHz. To avoid interference of the data communication no other devices generating interference emission in this frequency band must be operated in its vicinity. Such devices are for example frequency converters and switched-mode power supplies.

5.4 Fixing

► Fix the device using the supplied nuts (M18).

6 Electrical connection

The RFID read/write head has a 5-pole round M12 connector (A-coded). The pin connection corresponds to CiA DR-303-1.

	1: CAN_SHLD 2: CAN V+ 3: CAN GND 4: CAN_H 5: CAN_L	n.c. (not connected) Supply voltage 24 V DC GND H bus cable L bus cable
M12 connector CAN		

UK

The device does not have an internal CAN terminating resistor. A connection cable without terminating resistor can cause interference on the CAN bus.

Use 120 Ω terminating resistors or a connection cable with integrated terminating resistor, e.g. article EVC492.

7 Indicators

Operating status	LED red	LED green	LED yellow
Preoperational	Off	Lights permanently	Off
Preoperational and tag detected	Off	Flashes alternately with yellow LED (every 1.6 s)	Flashes alternately with green LED (every 1.6 s)
Operational	Off	Flashes (every 0.4 s)	Off
Operational and tag detected	Off	Off	Lights permanently
Configuration error	Flashes (every 0.4 s)	LED reacts according to the current operating status	
Error in the CAN network	Flashes (every 1.2 s)		
CAN: bus OFF	Lights permanently	Off	Off
LSS service active	Flashing	Off	Off
Hardware error detected in the device	Off	Off	Flashing

8 CANopen interface

The RFID read/write head has a standardised CANopen interface according to CiA DS-301. All measured values and parameters can be accessed via the object directory (OD). The individual configuration can be saved in the internal permanent memory.

8.1 CANopen functions

The following CANopen functions are available:

- 64 transmit and receive process data objects (TPDO1..64, RPDO1..64) in two possible operating modes:
  - individual check via a remote transmit-request telegram (RTR)
  - event-controlled transmission
- Error messages per emergency object (EMCY) with support of the:
  - general error register
  - manufacturer-specific register
  - error list (pre-defined error field)
- Monitoring mechanism heartbeat
- Status and error indication via LED
- In addition to the CiA DS-301 functionality there are more manufacturer and profile-specific characteristics:
  - setting of the node ID and the bit rate via object directory entry (SDO)
  - configuration and reading/writing of operational data via service data objects (SDO)
- Support of the layer settings service (LSS)

8.2 Change the node ID and bit rate

The device supports several options how to change the node ID and the bit rate.

The device is delivered with the node ID 32 and a bit rate of 125 Kbits/s.

Each node ID must only be assigned once in the CANopen network. If a node ID is assigned several times, malfunction in the CANopen network will result.

8.2.1 Change the node ID and bit rate in the object directory

The node ID is entered in the object directory in the objects 0x20F0 and 0x20F1. If the two values are identical, the setting is stored and is active after a software reset of the device. Values between 1 and 127 may be used as node ID.

The bit rate is entered in the objects 0x20F2 and 0x20F3. If the two values are identical, the setting is stored and is active after a software reset of the device. The following values may be used as bit rate:

Value	Bit rate
0	1000 Kbits/s
1	800 Kbits/s
2	500 Kbits/s
3	250 Kbits/s
4	125 Kbits/s
5	100 Kbits/s
6	50 Kbits/s
7	20 Kbits/s

If a master is used in the CANopen network for central storage of parameters, the changed values for node ID (0x20F0 and 0x20F1) and bit rate (0x20F2 and 0x20F3) must be additionally entered in the master.

Otherwise the values will be reset during each start of the CANopen network.

8.2.2 Set the note ID and bit rate via LSS

Using the layer setting service (LSS) an LSS master can change the node ID and bit rate of the device (LSS slave) via the CAN bus. The LSS master sets all LSS slaves to a configuration mode. Each LSS slave can be unambiguously identified via the device data (vendor ID, product code, revision number and serial number).

To change the bit rate the LSS master transfers the new bit rate in the configuration mode with the service "Configure timing bit". The LSS slave replies to the LSS master if the new bit rate is supported. Then the LSS master transmits the time "Switch delay" via the service "Activate bit timing" after which the new bit rate should be activated. After activation the LSS master switches the LSS slave again to the operating mode.

To change the node ID the LSS master transfers the new node ID in the configuration mode. The LSS slave replies to the master if the new node ID is valid. After changing the node ID the LSS master switches the LSS slave again to the operating mode.

The new bit rate and node ID become active after a software reset of the LSS slave.

8.3 Set-up

The CANopen standard CiA301 defines three possible operating states:

Pre-Operational
In the pre-operational state no PDO messages (process data) can be transmitted. The pre-operational state is used to set the sensor parameters or as standby mode.

During booting in the pre-operational mode on the CAN bus the device reports with the bootUP message "0x700+Node-ID".

Operational
In the operational state all communication services are carried out. The operational state is used to exchange the process data while in operation.

Stopped
In the stopped state only NMT messages (network management) are possible. This allows almost complete separation of redundant or faulty sensors from the bus.

The master or network manager can request the sensor via NMT messages to change the state accordingly.

8.4 Communication types of the process data object (PDO)

The TPDO can be checked at any time by transmitting a remote transmit-request telegram (RTR). Otherwise the TPDOs are sent automatically as soon as their value changes (event-driven).

A total of 64 TPDOs and 64 RPDOs is available; on delivery only the first 4 of each are active. If the configuration of the CANopen network allows it, the remaining process data objects can also be activated.

The process data is assigned to the linear address range in the standard settings of the RFID tag. The TPDO1 maps e.g. the first 8 bytes of the user data memory of the RFID tag.

Reading of the memory and transmission of the data via TPDO is effected automatically as soon as a new RFID tag is detected.

Writing of the data is effected in the same way by write access to the respective RPDO.

Data transfer per process data object is only possible in the "Operational" operating mode (→ 8.3 Set-up).

8.5 Object directory (OD)

Index	Subindex	Name (object)	Type	Access	Default value	PDO mapping capability	Save object value
CANopen communication (CiA 301)							
0x1000	0x00	Device type	u32	ro	0x00000000	-	-
0x1001	0x00	Error register	u8	ro	0x00	-	-
0x1003	0x01 0x02	Predefined error field	u32	ro	0x00000000		
0x1008	0x00	Manufacturer device name	vSTR	ro	DTM425	-	-
0x1009	0x00	Manufacturer hardware version	vSTR	ro	Current hardware version	-	-
0x100A	0x00	Manufacturer software version	vSTR	ro	Current software version	-	-
0x1010	0x01	Save parameter (save device parameter in non-volatile memory)	u32	rw	0x00000000	-	-
0x1011	0x01	Load default communication parameter	u32	rw	0x00000000	-	-
0x1014	0x00	COB ID EMCY (COB ID emergency message)	u32	rw	Node ID+ 0x80	-	
0x1015	0x00	Inhibit time EMCY (inhibit time for EMCY messages)	u16	rw	0x0000	-	Yes
0x1017	0x00	Producer heartbeat time (time difference between heartbeats sent in ms)	t16	ro	0x0000	-	Yes
0x1018	0x01	Vendor ID	u32	ro	0x0069666D	-	-
	0x02	Product code	u32	ro	0x20002000	-	-
	0x03	Revision number	u32	ro	Major revision + current software version	-	-
	0x04	Serial number	u32	ro	Serial number of the device	-	-
0x1200	0x01	COB ID client to server	u32	ro	Node ID+ 0x600	-	-
	0x02	COB ID client to server	u32	ro	Node ID+ 0x580	-	-

UK

Index	Subindex	Name (object)	Type	Access	Default value	PDO mapping capability	Save object value
0x1400-0x143F	0x01	RPDO parameter: COB ID	u32	rw	(→ 8.9.2)	-	Yes
	0x02	RPDO parameter: transmission type	u8	ro	0xFF	-	Yes
0x1600-0x163F	0x01-0x08	RPDO mapping	u32	rw	(→ 8.9.2)	-	Yes
0x1800-0x183F	0x01	TPDO parameter: COB ID	u32	rw	(→ 8.9.1)	-	Yes
	0x02	TPDO parameter: transmission type	u8	ro	0xFF	-	Yes
	0x03	TPDO parameter: inhibit time	u16	rw	0x00	-	Yes
0x1A00-0x1A3F	0x01-0x08	TPDO mapping	u32	rw	(→ 8.9.1)	-	Yes
Bus configuration							
0x20F0	0x00	Node ID switch A (node ID for CANopen communication)	u8	rw	32	-	Auto save
0x20F1	0x00	Node ID switch B (node ID for CANopen communication)	u8	rw	32	-	Auto save
0x20F2	0x00	Bit rate switch A (CAN bus bit rate)	u8	rw	4	-	Auto save
0x20F3	0x00	Bit rate switch B (CAN bus bit rate)	u8	rw	4	-	Auto save
Reader status and control							
0x2150	0x00	Device status (device status flags)	u32	ro		Yes	-
0x2151	0x00	Antenna active (enable HF front end of the device)	bool	rw	1	-	Yes
0x2160	0x01-0xFE	Tag type definition (name of supported tags)	dom	ro	(→ 8.12)	-	-
0x2161	0x00	Tag type selection (value selects tag type, which is defined in 0x2160)	u8	rw	2	-	Yes

Index	Subindex	Name (object)	Type	Access	Default value	PDO mapping capability	Save object value
Tag information							
0x2180	0x00	Current UID (UID of the tag that is in read range, PDO mappable)	u64	ro	0x0000000000000000	Yes	-
0x2181	0x00	Current DSFID (DSFID of the tag that is in read range, PDO mappable)	u8	ro	0x00	Yes	-
0x2182	0x01	Tag information: UID	u64	ro	0x0000000000000000	-	-
	0x02	Tag information: DSFID	u8	ro	0x00	-	-
	0x03	Tag information: AFI	u8	ro	0x00	-	-
	0x04	Tag information: memory size	u32	ro	0x00000000	-	-
	0x05	Tag information: IC reference	u8	ro	0x00	-	-
	0x06	Tag information: tag type (detected tag type, defined in 0x2160)	u8	ro	0x00	-	-
Read data mappable							
0x2200	0x01-0x40	Starting address read (start of the address range on the tag that should be read)	u16	rw	(→ 8.9.2)	-	Yes
0x2201	0x01-0x40	Length read (length of the memory range on the tag that should be read; max. 8 bytes)	u8	rw	(→ 8.9.2)	-	Yes
0x220A	0x01-0x40	Tag data (8 bytes of tag data, updated when new tag enters reading area)	u64	ro		Yes	-
Read data block							
0x2280	0x00	Starting address read (start of the address range on the tag that should be read)	u16	rw	0x0000	-	Yes
0x2281	0x00	Length read (length of the memory range on the tag that should be read)	u16	rw	0x0000	-	Yes

Index	Subindex	Name (object)	Type	Access	Default value	PDO mapping capability	Save object value
0x2282	0x00	Tag data (requested data from tag, configured in objects 0x2280 and 0x2281)	dom	ro		-	-
Write data mappable							
0x2300	0x01-0x40	Starting address write (start of the address range on the tag that should be written)	u16	rw	(→ 8.9.1)	-	Yes
0x2301	0x01-0x40	Length write (length of the memory range on the tag that should be written; max. 8 bytes)	u8	rw	(→ 8.9.1)	-	Yes
0x2302	0x01-0x40	Auto write (enable automatic write access, if new tag detected)	bool	rw	0	-	Yes
0x230A	0x01-0x40	Tag data (8 bytes of tag data)	u64	rww		Yes	-
0x230F	0x00	Write trigger	u64	rww	0x00000000 00000000	Yes	
Write data block							
0x2380	0x00	Starting address write (start of the address range on the tag that should be written)	u16	rw	0x0000	-	Yes
0x2381	0x00	Length write (length of the memory range on the tag that should be written)	u16	rw	0x0000	-	Yes
0x2382	0x00	Tag data (data that should be written to the tag, configured in objects 0x2380 and 0x2381)	dom	wo		-	-
Lock data block							
0x2480	0x00	Starting address lock (start of the address range on the tag that should be locked must be aligned to tag blocks)	u16	rw	0x0000	-	Yes

Index	Subindex	Name (object)	Type	Access	Default value	PDO mapping capability	Save object value
0x2481	0x00	Length lock (length of the memory range on the tag that should be locked must be aligned to tag blocks)	u16	rw	0x0000	-	Yes
0x2482	0x00	Tag lock trigger (trigger for locking data on the tag, configured in objects 0x2480 and 0x2481)	bool	wo		-	-

8.6 Error messages

The device supports a number of emergency messages that are sent in the event of a communication, hardware or RFID error. If one of these errors occurs, the error register (OV index 0x1001) and the pre-defined error field (OV index 0x1003) are updated.

The COB ID of the emergency message can be changed in the object "COB ID EMCY" (OV index 0x1014). By setting bit 31 in this object the emergency messages are deactivated.

The disable time between two emergency messages can be defined via the object 0x1015. The indication is made in steps of 100 µs.

The COB ID of the emergency messages is preset to 0x80 + node ID.

Emergency error code	Error register (0x1001)	Manufacturer error code	Manufacturer error name	Emergency description
0x8210	0x11			Protocol - PDO not processed due to length error
0x8130	0x01			Monitoring - node guarding or heartbeat error
0x8100	0x11			Monitoring - general communication error, sent in case of a busoff
0x5000	0x81	0x01		Device hardware error (antenna error)
0x4200	0x09	0x02		Device temperature too high
0xFF00	0x81	0x01	RX: ISO_ COMMAND_ ERROR_NO_ RESPONSE	Tag did not answer; maybe tag is not in the field any more?

Emergency error code	Error register (0x1001)	Manufacturer error code	Manufacturer error name	Emergency description
0xFF00	0x81	0x02	RX: ISO_ COMMAND_ ERROR_RX_ ERROR	Error while receiving the answer from tag (CRC error, framing error, collision, etc.)
0xFF01	0x81	0x01	TX: ISO_ COMMAND_ ERROR_NO_ RESPONSE	Tag did not answer; maybe tag is not in the field any more?
0xFF01	0x81	0x02	TX: ISO_ COMMAND_ ERROR_RX_ ERROR	Error while receiving the answer from tag (CRC error, framing error, collision, etc.)
0xFF02	0x81	0x01	ISO_TAG_ ERROR_ COMMAND_NOT_ SPECIFIED	The specific command is not supported. Example: command code error
0xFF02	0x81	0x02	ISO_TAG_ ERROR_ COMMAND_ SYNTAX	Cannot recognise the command. The number of blocks is above the limit. Example: format error
0xFF02	0x81	0x03	ISO_TAG_ ERROR_ OPTION_NOT_ SUPPORTED	Specific options are not supported.
0xFF02	0x81	0x0F	ISO_TAG_ ERROR_OTHER	Other errors
0xFF02	0x81	0x10	ISO_TAG_ ERROR_BLOCK_ NOT_USABLE	The specified block cannot be used (or was not found).
0xFF02	0x81	0x11	ISO_TAG_ ERROR_BLOCK_ ALREADY_ BLOCKED	The specified block has already been locked and cannot be locked again.
0xFF02	0x81	0x12	ISO_TAG_ ERROR_ BLOCK_NOT_ UPDATEABLE	The specified block has already been locked and its contents cannot be updated.
0xFF02	0x81	0x13	ISO_TAG_ ERROR_BLOCK_ WRITE_VERIFY	The specified block could not be programmed normally (a write verify error occurred).
0xFF02	0x81	0x14	ISO_TAG_ ERROR_BLOCK_ LOCK_VERIFY	The specified block could not be locked normally (a lock verify error occurred) .
0xFF03	0x81	0x00	STATUS_ BUFFER_OVERFL	Internal buffer overflow

8.7 Monitoring activity via heartbeat

By means of the heartbeat function the activity of a device in the CANopen network can be monitored by the master. The device regularly sends a heartbeat message containing the device status.

The heartbeat function is activated by entering a value greater than "0" into the heartbeat interval time object (OV index 0x1017). The value indicates the time between two heartbeat signals in milliseconds. The heartbeat function is deactivated with the value "0".

8.8 Change objects

Changes of the objects in the object directory are applied at once. The changes will get lost by a reset. To prevent this the objects have to be saved in the internal permanent memory (flash). All objects marked in the object directory as "Save object value: yes" are permanently stored in the device flash. By writing the command "Save" (65766173h) to save the objects (OV index 1010h/01h) all current objects of the object directory are transferred to the flash memory.

The objects can be reset to factory setting by writing the signature "Load" (64616F6Ch) to the OV index 1011h/01h. After a reset the changes are applied.

Depending on the architecture of the CANopen network the objects can also be stored centrally in a CANopen master. In this case the objects are transferred to the device when the system is started and the locally stored values are overwritten.

Special features of the object node ID (OV index 0x20F0 and 0x20F1) and the bit rate (OV index 0x20F2 and 0x20F3):

- Changes of the objects are only applied after a reset (→ 8.2 Change the node ID and bit rate).
- The objects cannot be transferred to the flash via the OV index 1010h/01h.
- The objects cannot be reset to the factory setting via the OV index 1011h/01h.

8.9 Process data objects

64 transmit and receive process data objects each are available. On delivery 4 process data objects are active.

8.9.1 Transmit process data objects (TPDO)

The table below contains the transmit process data objects (TPDO) on delivery.

	PDO mapping settings	Object directory			Tag memory	
TPDO	**COB**	**Mapped object index**	**Mapped object subindex**	**Mapped object length**	**Starting address read**	**Length read**
1	Node ID + 0x0180	0x2150	0x00	0x20	Device status	
2	Node ID + 0x0280	0x220A	0x01	0x40	0x00000000	0x08
3	Node ID + 0x0380	0x220A	0x02	0x40	0x00000008	0x08
4	Node ID + 0x0480	0x220A	0x03	0x40	0x00000010	0x08
5	0 (disabled)	0x220A	0x04	0x40	0x00000018	0x08
64	0 (disabled)	0x220A	0x40	0x04	0x000001F0	0x08

8.9.2 Receive process data objects (RPDO)

The table below contains the receive process data objects (RPDO) on delivery.

	PDO mapping settings	Object directory			Tag memory	
RPDO	**COB**	**Mapped object index**	**Mapped object subindex**	**Mapped object length**	**Starting address write**	**Length write**
1	Node ID + 0x0200	0x230F	0x01	0x40	Write trigger	
2	Node ID + 0x0300	0x230A	0x02	0x40	0x00000000	0x08
3	Node ID + 0x0400	0x230A	0x03	0x40	0x00000008	0x08
4	Node ID + 0x0500	0x230A	0x04	0x40	0x00000010	0x08
5	0 (disabled)	0x230A	0x05	0x40	0x00000018	0x08
64	0 (disabled)	0x230A	0x40	0x04	0x000001F8	0x08

8.10 Device status

The current device status is represented in the object "Device status" (OV index 0x2150, subindex 0x00). On delivery the object is assigned to TPDO1.

Bit	31	30	29	28	27	26	25	24
Status	tag_err							
Default value	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Status	write_err							
Default value	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Status	read_err							
Default value	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Status	r	r	buf_ovfl	fr_err	busy	present	ant	pow
Default value	0	0	0	0	0	0	1	1

Status	**Value**	**Description**	**EMCY Msg.**
pow	1	Power enabled (always 1)	
ant	0	Antenna disabled	
	1	Antenna enabled	
present	0	No tag present	
	1	Tag present	
busy	0	Idle	
	1	Read or write access active	
fr_err	0	Frontend ok	
	1	Frontend error detected (hardware problem)	Yes
buf_ovfl	0	Buffer ok	
	1	Buffer overflow detected	Yes
read_err		Error of last read operation	Yes
write_err		Error of last write operation	Yes
tag_err		Error message from tag for last operation	Yes

Read error codes (updated after each read access of the tag)		
0x00	ISO_COMMAND_ERROR_NO_ERROR	No error, command successfully executed

Read error codes (updated after each read access of the tag)		
0x01	ISO_COMMAND_ERROR_NO_ RESPONSE	Tag did not answer; maybe tag is not in the field any more
0x02	ISO_COMMAND_ERROR_RX_ERROR	Error while receiving the answer from tag (CRC error, framing error, collision, etc.)

Write error codes (updated after each write access of the tag)		
0x00	ISO_COMMAND_ERROR_NO_ERROR	No error, command successfully executed
0x01	ISO_COMMAND_ERROR_NO_ RESPONSE	Tag did not answer; maybe tag is not in the field any more?
0x02	ISO_COMMAND_ERROR_RX_ERROR	Error while receiving the answer from tag (CRC error, framing error, collision, etc.)

Tag error codes (updated after each read or write access of the tag)		
0x00	ISO_TAG_ERROR_NO_ERROR	No error from tag
0x01	ISO_TAG_ERROR_COMMAND_NOT_ SPECIFIED	The specific command is not supported. Example: command code error
0x02	ISO_TAG_ERROR_COMMAND_SYNTAX	Cannot recognise the command. The number of blocks is above the limit. Example: format error
0x03	ISO_TAG_ERROR_OPTION_NOT_ SUPPORTED	Specific options are not supported
0x0F	ISO_TAG_ERROR_OTHER	Other errors
0x10	ISO_TAG_ERROR_BLOCK_NOT_ USABLE	The specified block cannot be used (or was not found)
0x11	ISO_TAG_ERROR_BLOCK_ALREADY_ BLOCKED	The specified block has already been locked and cannot be locked again
0x12	ISO_TAG_ERROR_BLOCK_NOT_ UPDATEABLE	The specified block has already been locked and its contents cannot be updated
0x13	ISO_TAG_ERROR_BLOCK_WRITE_ VERIFY	The specified block could not be programmed normally (a write verify error occurred)
0x14	ISO_TAG_ERROR_BLOCK_LOCK_ VERIFY	The specified block could not be locked normally (a lock verify error occurred)

8.11 Deactivate antenna

The antenna in the device can be deactivated if the value 0 is written to the object "Antenna active" (OV index 0x2151). In this case no tag is detected any more since the magnetic field of the device is no longer active.

The antenna is reactivated with the value 1. With the object "Antenna active" it is possible to prevent interference between two devices placed next to each other by alternately deactivating the antennas of the two devices.

8.12 Select tag type

The device is compatible with several tag types according to ISO15693. Depending on the size of the user data memory and manufacturer the tags differ in the access to data. Therefore the device must know which type of tag is used in the system.

In object 0x2161 the tag type used in the RFID system can be selected. The available tag types can be read in the object 0x2180, subindex 0x01-0xFE.

Tag type	Name	Block size [Byte]	Number of blocks
1	User defined	?	?
2	I code SLI	4	28
3	I code SLI-S	4	40
4	I code SLI-L	4	8
5	F-MEM 2k	8	250
6	F-MEM 232b	4	58
7	F-MEM 8k	32	256
8	TI_32b	4	8
9	TI_256b	4	64
10	ST_128b	4	32
11	ST_256b	4	64
12	ST_8k	4	2048

Via the object 0x2182 0x06 it is possible to poll the tag type read by the device. First of all the detected tag type must be read from the object 0x2182 subindex 0x06 and this value must be entered in the object 0x2161.

Of special importance is tag type 1: The parameters "Block size" and "Number of blocks" are automatically determined by the device. If the parameters do not match the known tag types, type 1 "User defined" is used.

Detection of tag types is not supported by all tags.

The set tag type is only permanently saved in the device if the object "Save parameter" is used
(→ 8.8 Change objects).

Tag type 2 is preset.

8.13 Read information of a tag

The information of a tag can be read via the objects 0x2180 to 0x2182. To do so, the tag has to be within the detection range of the device.

The objects 0x2180 and 0x2182 are only valid as long as the tag is detected. If there is no tag within the range, the values of the objects are reset to 0.

The value of the object 0x2182 can be read from the tag on request.

Reading of information is not supported by each tag type.

9 Data transfer with the tag

9.1 Read data from the tag via PDO transfer

The transfer of the PDO data from the tag may be event-controlled. That means that the configured TPDOs are automatically transmitted by the device when the data change. This is the case, for example, when a new tag is detected in the detection range of the device. The data is automatically read by the tag and transferred by means of the TPDOs via the CAN bus.

The data that was read by the tag and assigned to a TPDO are in the objects 0x220A with the subindexes 0x01-0x40.

Only that data is read by the tag that is assigned to a TPDO. Data objects that are not assigned are not updated automatically.

There are two objects for each data object that are used for configuration: 0x2200 (starting address read) and 0x2201 (length read) with subindexes matching the data object. The start address in the user data area of the tag and length of the files to be read are set in the objects.

Only 64-bit data (8 bytes) is always transmitted by a TPDO. If the configured data length is smaller than 64 bits, the remaining bits are filled with 0.

Max. 64 bits can be transmitted in one TPDO. If larger data volumes are to be transferred, more TPDOs have to be assigned and the respective data objects are to be configured.

9.1.1 Example 1

The data range 0x10 to 0x18 (8 bytes) is to be transferred with the 2nd TPDO.

	PDO mapping settings	Object directory		
TPDO	**COB**	**Object index**	**Object subindex**	**Object length**
2	Node ID + 0x0280	0x220A	0x01	0x40

Object directory			
Index	**Subindex**	**Name (object)**	**Value**
0x2200	0x01	Starting address read (start of the address range on the tag that should be read)	0x10
0x2201	0x01	Length read (length of the memory range on the tag that should be read; max. 8 bytes)	0x08

9.1.2 Example 2

The data range 0x44 to 0x48 (4 bytes) is to be transferred with the 6th TPDO.

	PDO mapping settings	**Object directory**		
TPDO	**COB**	**Object index**	**Object subindex**	**Object length**
6	Node ID + 0x0680	0x220A	0x05	0x40

Object directory			
Index	**Subindex**	**Name (object)**	**Value**
0x2200	0x05	Starting address read (start of the address range on the tag that should be read)	0x44
0x2201	0x05	Length read (length of the memory range on the tag that should be read; max. 8 bytes)	0x04

9.2 Write data to the tag via PDO transfer

To write data to a tag via PDO transfer an RPDO must be assigned to the object 0x230A with a subindex in the range from 0x01 to 0x40. The address of the tag user data range to which the data is to be written is defined in object 0x2300. The subindexes of these objects have to be identical.

The tag is written on after the data was written to the RPDO and the respective bit was changed in "Write trigger" object (OV index 0x230F, subindex 0x00).

	MSB								**LSB**
Bit	63	62	61	..	..	..	2	1	0
Trigger	tr64	tr63	tr62	..	..	..	tr3	tr2	tr1
Default value	0	0	0	0	0	0	0	0	0

Trigger	**Function**
tr64	Trigger for tag data 64 (0x230A/0x40)
tr63	Trigger for tag data 63 (0x230A/0x3F)
tr62	Trigger for tag data 62 (0x230A/0x3E)
tr61	Trigger for tag data 61 (0x230A/0x3D)

Trigger	Function
tr60	Trigger for tag data 60 (0x230A/0x3C)
tr59	Trigger for tag data 59 (0x230A/0x3B)
tr58	Trigger for tag data 58 (0x230A/0x3A)
…	…
tr6	Trigger for tag data 6 (0x230A/0x6)
tr5	Trigger for tag data 5 (0x230A/0x5)
tr4	Trigger for tag data 4 (0x230A/0x4)
tr3	Trigger for tag data 3 (0x230A/0x3)
tr2	Trigger for tag data 2 (0x230A/0x2)
tr1	Trigger for tag data 1 (0x230A/0x1)

The writing process is always made with the bit change of the respective bit (0->1 or 1->0). Ideally, the object "Write trigger" (OV index 0x230F, subindex 0x00) is assigned to an RDPO. On delivery the object "Write trigger" is assigned to the first RPDO.

Automatic writing of data can be activated with the object "Auto write" (OV index 0x2302). As soon as a tag is in the detection range, the last data is written to the tag.

Only data up to the configured data length is written to the tag. The subsequent data is ignored. If more than 8 bytes are transferred, more RPDOs have to be assigned and the respective data objects have to be configured.

9.2.1 Example 1

The data range 0x10 to 0x18 (8 bytes) is to be transferred with the 2nd RPDO.

	PDO mapping settings	Object directory		
RPDO	**COB**	**Object index**	**Object subindex**	**Object length**
2	Node ID + 0x0200	0x230A	0x01	0x40

Object directory			
Index	**Subindex**	**Name (object)**	**Value**
0x2300	0x01	Starting address read (start of the address range on the tag that should be read)	0x10
0x2301	0x01	Length read (length of the memory range on the tag that should be read; max. 8 bytes)	0x08
0x2302	0x01	Auto write	0x00

Transfer data via RPDO:

PDO transmission	PDO	Data
To the device	RPDO 2	0x12345678

Start write access:

PDO transmission	PDO	Data
To the device	RPDO 1	Switch bit 0

9.2.2 Example 2

The data range 0x44 to 0x48 (4 bytes) is to be transferred with the 6th RPDO. In addition this data is to be written to a tag each time it reaches the detection range of the device.

	PDO mapping settings	Object directory		
RPDO	**COB**	**Object index**	**Object subindex**	**Object length**
6	Node ID + 0x0600	0x230A	0x05	0x40

Object directory			
Index	**Subindex**	**Name (object)**	**Value**
0x2300	0x05	Starting address read (start of the address range on the tag that should be read)	0x44
0x2301	0x05	Length read (length of the memory range on the tag that should be read; max. 8 bytes)	0x04
0x2302	0x05	Auto write	0x01

Transfer data via RPDO:

PDO transmission	PDO	Data
To the device	RPDO 6	0x12340000

The data is written to the tag when it has reached the detection range.

64-bit data (8 bytes) always have to be sent to an RPDO. If the configured data length is smaller than 64 bits, the remaining bits are ignored.

9.3 Error handling for PDO transfer

If a read or write access to a tag is not possible, the device creates an emergency message on the CAN bus.

The error code can be read from the error register (OV index 0x1001, subindex 0x00) and the predefined error field (OV index 0x1003, subindex 0x01-0x02) (→ 8.6 Error messages).

9.4 Read data from the tag via SDO transfer

To read data from a tag via SDO transfer it is necessary to define the data address and length on the tag. The address must be indicated in object 0x2280 and the data length in object 0x2281.

Then read access can be started from the tag via a data transfer to object 0x2282.

9.4.1 Example

The data range 0x50 to 0x70 is to be read from the tag.

Object directory			
Index	**Subindex**	**Name (object)**	**Value**
0x2280	0x00	Starting address read (start of the address range on the tag that should be read)	0x50
0x2281	0x00	Length read (length of the memory range on the tag that should be read; max. 8 bytes)	0x20

Transfer is started via reading the object 0x2282, subindex 0x00.

The data is transferred in one piece as domain data type. Up to a data length of 4 bytes transfer is effected as expedited transfer; longer data lengths as segmented transfer.

The recipient must be prepared for temporary storage and processing of the data.

9.5 Write data to the tag via SDO transfer

To write data to a tag via SDO transfer it is necessary to define the data address and length on the tag.

The address must be indicated in object 0x2380 and the data length in object 0x2381. Then the write access to the tag can be started via a data transfer to object 0x2382.

9.5.1 Example

The data range 0x34 to 0x37 is to be transferred to the tag.

Object directory			
Index	**Subindex**	**Name (object)**	**Value**
0x2380	0x00	Starting address write (start of the address range on the tag that should be written)	0x34
0x2381	0x00	Length write (length of the memory range on the tag that should be written)	0x03
0x2382	0x00	Tag data (data that should be written to the tag)	0x010203

The data is transferred in one piece as domain data type. Up to a data length of 4 bytes transfer is effected as expedited transfer; longer data lengths as segmented transfer.

The transmitter must be able to provide the indicated data length.

9.6 Lock data ranges on the tag via SDO transfer

The data ranges of the tag can be write-protected.

The write protection of a data range cannot be removed.

The start address of the data range to be protected is stored in the object "Starting address lock" (OV index 0x2480). In addition the data range length is stored in the object "Length lock" (OV index 0x2481).

The start address must be identical with the start address of a storage block on the tag. The length must be a multiple of the length of a storage block on the tag.

To activate the write protection 1 is written on the trigger (OV index 0x2482).

9.6.1 Example

The data range 0x04 to 0x0C is to be write-protected for a tag of block size 4 (2 blocks or 8 bytes).

Object directory			
Index	**Subindex**	**Name (object)**	**Value**
0x2480	0x00	Starting address lock (start of the address range on the tag that should be locked)	0x04
0x2481	0x00	Length write (length of the memory range on the tag that should be locked)	0x08
0x2482	0x00	Tag lock trigger	0x01

9.7 Error handling for SDO transfer

SDO transfers are acknowledged transfers. If there is an error during transfer or during actions caused by the transfer, an error is signalled after the SDO transfer.

SDO error code	Description	Possible cause
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 a 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	Subindex does not exist.	

SDO error code	Description	Possible cause
0x06090030	Invalid value for parameter (download only).	
0x06090031	Value of written parameter is too high (download only).	
0x06090032	Value of written parameter is too low (download only).	
0x06090036	Maximum value is lower than minimum value.	
0x060A0023	Resource not available: SDO connection	
0x08000000	General error	
0x08000020	Data cannot be transferred to or stored in the application.	Read or write access error from tag. Detailed information in device status object (0x2150)
0x08000021	Data cannot be transferred to or stored in the application because of local control.	
0x08000022	Data cannot be transferred to or stored in the application because of the present device state.	
0x08000023	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error).	
0x08000024	No data available	Data length = 0

10 EDS files

The EDS file serves as a template for different configurations of a device type. The EDS file is turned into a DCF file which contains device configurations, object values, node ID and bit rate.

CANopen configuration tools are available for the configuration of the CANopen network and the devices.

An EDS file can be downloaded from ifm's homepage:

www.ifm.com → Service → Download → Identification systems

Contents of the EDS file:

- Communication functions and objects (to CANopen profile DS-301)
- Manufacturer-specific objects

The installation of the EDS file depends on the configuration tool. Please contact the manufacturer of your controller, if necessary.

11 Maintenance, repair and disposal

► Do not open the housing as the device does not contain any components which can be maintained by the user. The device must only be repaired by the manufacturer.

► Dispose of the device in accordance with the national environmental regulations.

12 Glossary

Term	Description
0b …	Binary value (for bit coding), e.g. 0b0001 0000
0x …	Hexadecimal value, e.g. 0x64 (= 100 decimal)
AFI	Indication of the application range of the tag
CAN	Controller Area Network (bus system for the use in mobile vehicles)
CAN_H	CAN high; CAN connection/cable with a high voltage level
CAN_L	CAN low; CAN connection/cable with a low voltage level
CANopen	CAN-based network protocol on the application level with an open configuration interface (object directory)
CiA	CAN in Automation e.V. (user and manufacturer organisation in Germany/Erlangen, definition and control body for CAN and CAN-based network protocols)
COB	CANopen communication object (PDO, SDO, EMCY, ...)
COB ID	Communication object identification number for assignment of the data packages in the CANopen network

Term	Description
DSFID	Identification number for the assignment of the data structure on the tag
EDS	Electronic data sheet
EMCY object	Emergency object (alarm message; device signals an error)
Emergency messages	Messages on the CANopen bus to communicate errors
Error reg	Error register (entry with an error code)
Heartbeat	Configurable cyclic monitoring among network participants. In contrast to "node guarding" no superior NMT master is required.
ID	Identifier characterising a CAN message. The numerical value of the ID also contains a priority concerning the bus access (ID 0 = highest priority)
Identifier	See ID
ID tag	RFID tag
LSS	Layer Settings Service, procedure to set basic device settings
NMT	Network management
NODE-ID	Unambiguous number of a participant in the CANopen network
Object / OBJ	Term for data/messages which can be exchanged in the CANopen network
OV	Object directory
PDO	Process Data Object; in the CANopen network to transfer process data in real time such as motor speed. PDOs have a higher priority than SDOs; in contrast to the SDOs they are transferred without confirmation. PDOs consist of a CAN message with identifier and up to 8 bytes of user data.
PDO mapping	Describes the application data transferred with a PDO.
ro	Read only (unidirectional: reading only)
RPDO	Process data object, received by the device
rw	Read-write (bidirectional, read / write)
SDO	Service Data Object; with this object direct access to the object directory of a network participant is possible (read/write). An SDO can consist of several CAN messages. The transfer of the individual messages is confirmed by the addressed participant. With the SDOs, devices can be configured and parameters can be set.
TPDO	Process data object, sent by the device
UID	Unique identification number of a tag
wo	Write only