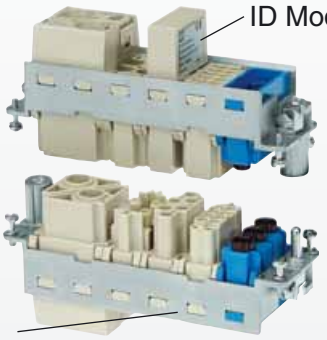
Electronic Components for Connectors

Han® ID Module

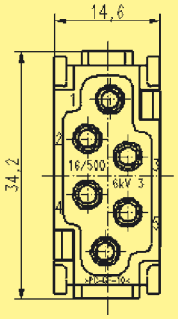
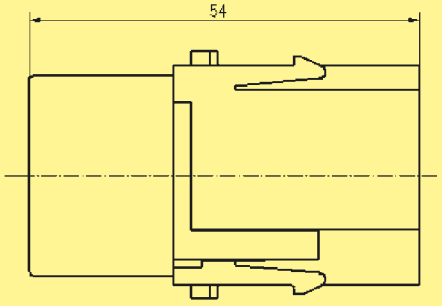


Control side

Electronic Identification module

Features

- Coding of tools possible (i.e. press tools) by means of an alpha numeric identification
- I²C Bus EEPROM as memory medium
- Communication with PLC via conventional digital I/Os
- Physical connection of PLC by means of well-proven Han® contacts
- Suitable for industrial applications
- Assembly of the module to the device by means of a Han® industrial connector

Identification	Part No.	Drawing	Dimensions in mm
ID Module	20 70 001 1001		

Technical characteristics

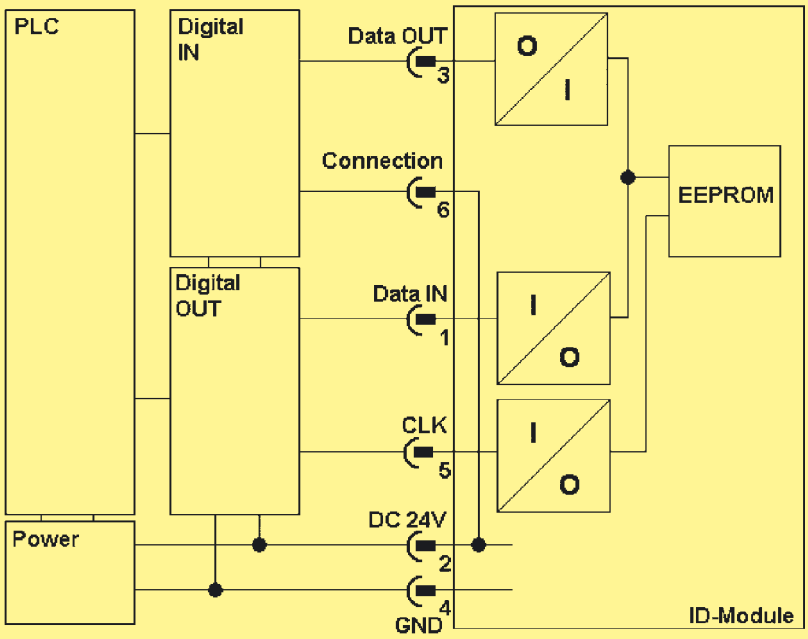
Code length	max. 128 Byte
Ambient temperature	0 … +70 °C in operation 0 … +85 °C in stocks
Power supply	24 V via digital I/O device
Electrical connector 24 V	Han E® Module
Degree of protection	IP 20 IP 65 / 68 within mated connector
Dimensions (W x H x D)	ca. 30 mm x 53 mm x 15 mm (incl. connector)
Max. length recommended between I/O device and ID module	100 m*

Pinning

Pin-No.	1	2	3	4	5	6
Function	Data IN	DC 24 V	Data OUT	GND	CLK	Connection

* the use of shielded cable is recommended for longer distances or EMC loaded environments

Block diagram / Wiring plan



Meaning of the connections

Pin-No.	Name	Meaning/function
1	Data IN	Input for data- and control signals from PLC
2	DC 24 V	Power connection ID Module
3	Data OUT	Output for data signals from ID Module to PLC
4	GND	Ground
5	CLK	System clock for synchronization
6	Connection	Output of the module for connection detection

General description

The HARTING connector identification module (ID Module) is suitable for storing of data and for coding of connectors. It is integrated in a Han-Modular® standard E Module.

The module can be connected to a 24 V digital I/O device of a PLC. Two digital inputs are used for detecting the module connection and the data input. Two digital outputs are used for the data output and system clock. Furthermore the module must be connected with 24 V and GND. Communication is carried out with voltage levels of 24 V according to the I²C bus standard. The total memory capacity is 128 byte, e.g. for storing part numbers to identify the module. It is also possible to stoire the start parameters or operating data for machine components.

A typical data structure is displayed in the following table:

Byte No.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Check-sum		Operating hours of tools					Start parameter					Part number			

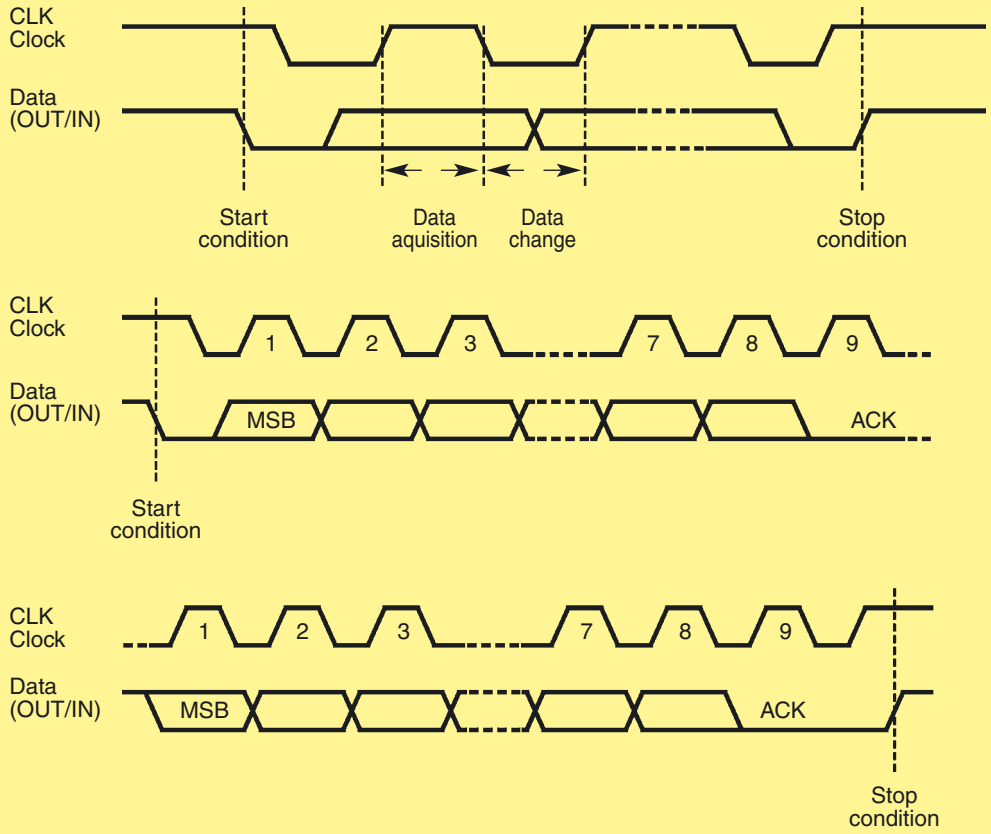
Application for the module can be found in modular machines and production lines. A great advantage of the module is the non volatile decentralized storing of e.g. operating data. When changing the location stored data can protect the machines from damages. In service cases of the equipment data can be analyzed to minimize service time.

Communication I²C bus protocol

Process of the Communication:

While storing data into the module signals from outputs of the plc are returned instantly to the inputs of the plc because of the internal connection between DATA IN and DATA OUT. Whren reading data from the module the outputs of the plc have to be a HIGH level, because for signalizing bit strings the module pulls down the level to LOW potential.

Any device that sends data is defined to be a transmitter, and any device that reads data is defined to be a receiver. The device that controls the data transfer is known as the master, and the other as the slave. A data transfer can only be initiated by the master, which will also provide the serial clock (CLK) for synchronization. In the concrete application the master is the PLC and the ID Module the only slave at the bus. PLC and ID Module can both be transmitters and receivers.



Start condition:

A START condition must precede any data transfer command. START is defined by a high to low transition of the data OUT line while the clock, CLK, is stable in the high state. The memory device continuously monitors (except during a programming cycle) the data OUT and CLK lines for a START condition, and will not respond unless one is given.

Stop condition:

STOP is identified by a low to high transition of the data OUT line while the clock CLK is stable in the high state. A STOPcondition terminates communication between the memory device and the bus master. A STOP condition at the end of a write command triggers the internal EEPROM write cycle.

Data input:

During data input, the memory device samples the data OUT signal on the rising edge of the clock, CLK. For correct device operation, the data IN / OUT signal must be stable during the clock low-to-high transition, and the data must change only when the CLK line is low.

Acknowledge bit:

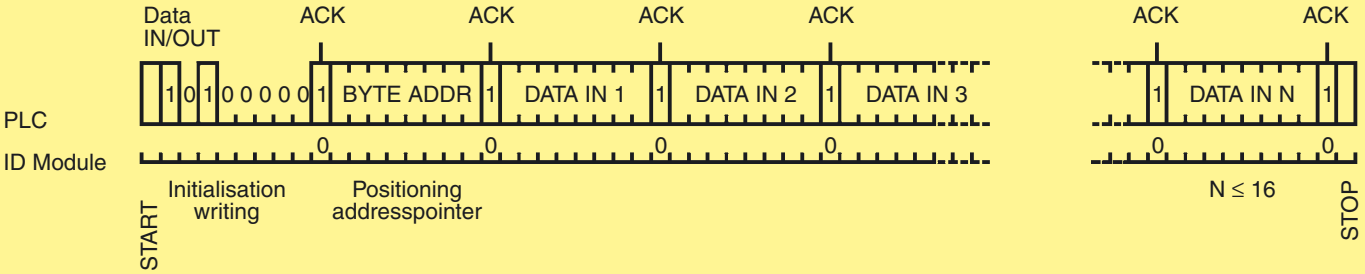
The data transfer works byte by byte starting with a MSB (Most Significant Bit). An acknowledge signal is used to indicate a successful byte transfer. The bus transmitter, whether it is master or slave, pulls the data OUT / data IN bus to HIGH signal after sending eight bits of data. During the 9th clock pulse period, the receiver pulls the data OUT / data IN bus low to acknowledge the receipt of the eight data bits.

Communication I²C bus protocol

Communication procedure for writing the module:

The communication starts with a Start-condition. After that an initial byte is sent on to the bus (4 bit device code 1010, 3 bit bus address fixed to 000 and write bit 0 1010 000 0), followed by an ACK-bit of the memory device (0). The next byte contains the start byte address. Writing data is shifted to the device byte by byte accknowledged by ACK-bit. Limited by the internal address counter max. 16 byte can be transferred without repositioning of the address pointer. Communication ends with a Stop-condition.

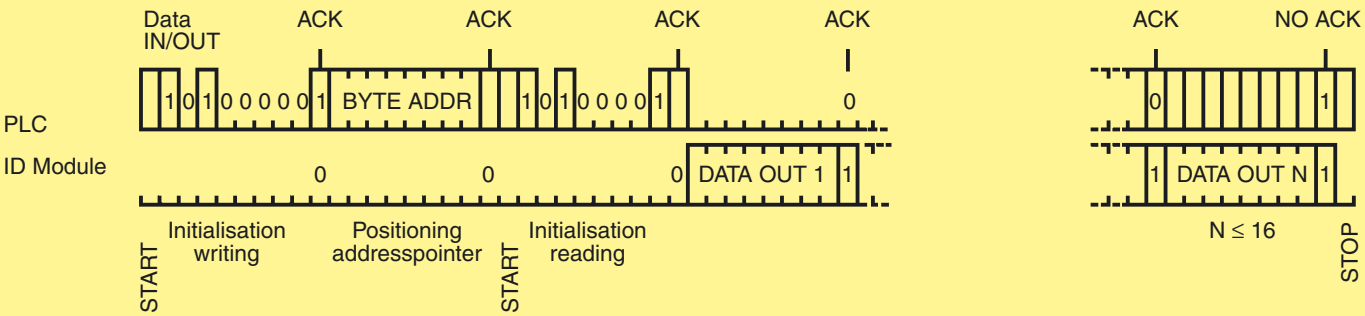
Writing data



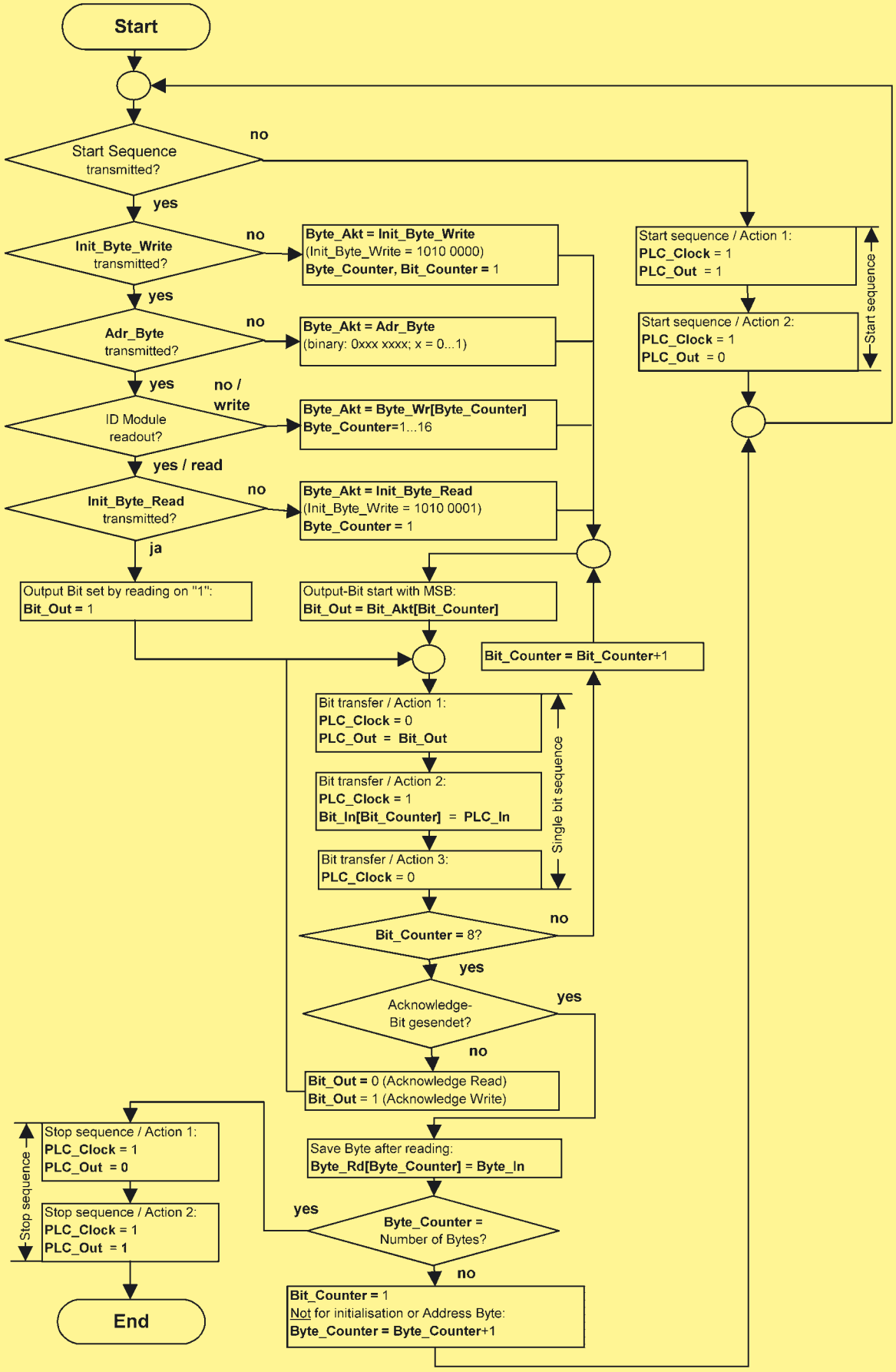
Communication procedure for reading the module:

The communication starts with a Start-condition. After that an initial byte is sent on to the bus (4 bit device code 1010, 3 bit bus address fixed to 000 and write bit 0 1010 000 0), followed by an ACK-bit of the memory device (0). The next byte contains the start byte address. After positioning the address pointer the device mode has to be changed to read mode. Therefore a Start-condition is sent and the last bit of the following initial byte has to be 1 (4 bit device code 1010, 3 bit bus address fixed to 000 and read bit 1 1010 000 1). After that the memory device starts sending data bytes. The PLC has to quit correctly in transferred bytes with an ACK-bit on LOW potential. Limited by the internal address counter max. 16 byte can be transferred without repositioning of the address pointer. To quit communication the PLC sends an ACK-bit on HIGH potential. Communication ends with a Stop-condition.

Reading data



Communication I²C bus protocol



General

1. The flow chart shows one single write- or read access; because of the restriction placed by internal address counter, a maximum of 16 bytes can be read or written in a single access procedure.

2. To simplify the chart, possibly necessary flag bits, usually required for branch lines, are not depicted. Neither is the reaction of the controller depicted if the ID Module does not transmit an acknowledgement.

3. In the case of assignments (=) the result of the operation is shown on the left.

4. The names for the variables used are to be regarded as suggestions – please refer to legend below. Please attend: **_Out** denotes the direction from the controller to the ID Module, and **_In** denotes the direction from the ID Module to the controller.

Legend

PLC_Out: Digital Output of the controller -> Pin No. 1 / Data IN of the ID Module

PLC_In: Digital Input of the controller -> Pin No. 3 / Data OUT of the ID Module

PLC_Clock: Digital Output of the controller -> Pin No. 5 / CLK of the ID Module

Init_Byte_Write: Initialisation byte, write (Content: 1010 0000)

Init_Byte_Read: Initialisation byte, read (Content: 1010 0001);

Adr_Byte: Address pointer byte (Content: 0xxx xxxx, here x = 0...1)

Bit_Out: Bit variable, Output internal

Byte_Akt: Byte variable, current internal output byte, consisting of: **Bit_Akt[Bit_Counter]** – Bit variable, current internal output bit

Byte_In: Byte variable, current internal input byte, consisting of: **Bit_In[Bit_Counter]** – Bit variable, current internal input bit

Bit_Counter: Counter, indicates the current bit number, allows values from 1 to 8;

Byte_Wr[Byte_Counter]: the byte variables for writing user data

Byte_Rd[Byte_Counter]: read byte variables

Byte_Counter: Counter, indicates the number of the current user data byte (but not the initialising or address byte), allows values from 1 to maximum 16