Eilersen Electric A/S
Kokkedal Industripark 4
DK-2980 Kokkedal
DENMARK
Tel: **+45 49 18 01 00**
Fax: **+45 49 18 02 00**

MCE2040 SERIEL COMMUNICATION MODULE

Transfer of status and weight for digital loadcells using MODBUS protocol

Applies for:

CYGNAL no.:	MCE2040A.MODBUS.031110.1
Document no.:	1110M1U1.DOC
Date:	2004-11-19
Rev.:	1

1) CONTENTS

2) INTRODUCTION

2.1 Introduction

This document describes the use of the Eilersen Electric MCE2040 serial communication module, when equipped with program listed on the front page. With this program the MCE2040 communication module is capable of transmitting weight data of up to 6 load-cells using its MODBUS protocol. Each loadcell is connected to the communication module through a MCE9610/MCE2010 loadcell interface module. The MCE2040 module can be connected to a MODBUS master using RS232, RS485 or RS422.

With the implemented protocol either status and weight for each loadcell is transferred, or an 'OR'ed status and summed weight for all loadcells is transferred. Functions as zeroing and calibration must be implemented on the MODBUS master.

Exchange of data is performed as described in the following.

3) DATA EXCHANGE

3.1 Modbus specifications

The serial Modbus communication is made using the following specifications:

Physical layer:	RS232, RS485 or RS422
Modbus Mode:	ASCII or RTU depending on DIP-switch
Modbus Address:	0-31 depending on DIP-switches
Modbus Data Format:	SI32 or FP32 depending on DIP-switch
Modbus Baudrate:	38400 bps
Modbus Parity:	Odd
Modbus Stopbits:	1 or 2 depending on Modbus parity
Modbus Databits:	7 or 8 depending on Modbus mode

For more specific information regarding Modbus please refer to the Modbus specification.

3.1.1 Physical layer

Depending on if the communication is to run on RS232, RS485 or RS422 the corresponding connections in the Modbus communication connector (9 pole sub-D connector) is used. This is explained in the hardware description later. If communication is to run on RS485 or RS422 it may be necessary to change the positions of jumper S1 and S2. The correct jumper setting is explained in the hardware description.

3.1.2 Modbus mode

Depending on whether the communication should run in ASCII or RTU mode this is selected using Sw1.6 as explained in the hardware description.

3.1.3 Modbus address

The desired Modbus communication address (0-31) for the MCE2040 communication module is set using Sw1.1-Sw1.5 as explained in the hardware description.

3.1.4 Modbus data format

It is possible to select which data format the weight indications should be represented in. As explained in the hardware description it is possible using Sw1.7 to select between 32 bit signed integer (SI32) or 32 bit IEEE754 floating point (FP32) format. Please refer to other literature for a description of the two data formats.

3.1.5 Modbus serial communication parameters

Baudrate and parity at which the Modbus communication is run is hard coded to 38400 bps and Odd parity respectively. Number of stopbits is automatically set to 1 because of the used parity. Number of databits is set to 7 or 8 depending on if Modbus mode is set to ASCII or RTU using Sw1.6.

3.2 Modbus telegrams

Using the Modbus communication it is possible to access the following on the MCE2040 communication module:

- loadcell register indicating found loadcells during power-up.
- status for each individual loadcell.
- weight signal for each individual loadcell in grams, or with corresponding loadcell exponent (if a better resolution is desired).
- total status (OR'ed) for all connected loadcells.
- total system weight signal (summed) for all connected loadcells in grams, or with corresponding system exponent (if a better resolution is desired).

This is done using 'Read Holding Registers' telegrams (see Modbus specification), that is described in the following along with the corresponding responses.

As the transferred status and weight values (no matter if its individual loadcell signals or system/summed signal) are directly from the individual loadcells without zeroing etc., status handling, calculation of system weight(s), zeroing- and calibration functions **must** be implemented on the Modbus master.

Calculation of system status(es) is made by logical OR of the status registers for the loadcells belonging together. Calculation of system weight(s) is made by adding weight registers for the loadcells belonging together. Note that the result is only valid if all status registers for these loadcells does **not** indicate error. It is also up to the Modbus master to ensure, that consistent loadcell data are used when calculating system weight(s); the used data must come from the same telegram.

<u>In the following these designations are used:</u>

LSBit = Least Significant Bit
MSBit = Most Significant Bit
LSB = Least Significant Byte
MSB = Most Significant Byte
LSW = Least Significant Word
MSW = Most Significant Word

3.2.1 Read LC-register, system status and system weight (grams)

Reading of the registers:

LC register:	is two bytes making a bit register to indicate the connected load-cells detected during power-up. Hence bit 0-5 will be ON, if the corresponding loadcell addresses were detected during power-up.
System status:	is two bytes making a register containing current system status achieved by logical OR of status for all detected loadcells. The meaning of each bit is explained later in the chapter about status codes.
System weight:	is four bytes making a register containing the current system weight in grams achieved by summation of weight signals for all detected loadcells.

is done by sending a "Read Holding Register" request to the MCE2040 module as follows:

	HEX	ASCII (chars)		RTU (binary)
Header:	xx	:		none
Slave Adr.:	xx	x	x	xxxxxxxx
Function:	03	0	3	00000011
Start Adr.(H):	00	0	0	00000000
Start Adr. (L):	01	0	1	00000001
# of points (H):	00	0	0	00000000
# of points (L):	04	0	4	00000100
Error Check:	xx	LRC(2 bytes)		CRC(2 bytes)
Trailer:	xx	CR	LF	none

To this the MCE2040 module responds as follows:

	HEX	ASCII (chars)		RTU (binary)
Header:	xx	:		none
Slave Adr.:	xx	x	x	xxxxxxxx
Function:	03	0	3	00000011
Byte Count:	08	0	8	00001000
Dat0(H)/LcRegister(MSB)	xx	x	x	xxxxxxxx
Dat0(L)/LcRegister(LSB)	xx	x	x	xxxxxxxx
Dat1(H)/SystemStatus(MSB)	xx	x	x	xxxxxxxx
Dat1(L)/SystemStatus(LSB)	xx	x	x	xxxxxxxx
Dat2(H)/SystemWeight(LSW)	xx	x	x	xxxxxxxx
Dat2(L)/SystemWeight(LSW)	xx	x	x	xxxxxxxx
Dat3(H)/SystemWeight(MSW)	xx	x	x	xxxxxxxx
Dat3(L)/SystemWeight(MSW)	xx	x	x	xxxxxxxx
Error Check:	xx	LRC(2 bytes)		CRC(2 bytes)
Trailer:	xx	CR	LF	none

3.2.2 Read LC-register, system status, system weight and system exponent

Reading of the registers:

LC register:	is two bytes making a bit register to indicate the connected load-cells detected during power-up. Hence bit 0-5 will be ON, if the corresponding loadcell addresses were detected during power-up.
System status:	is two bytes making a register containing current system status achieved by logical OR of status for all detected loadcells. The meaning of each bit is explained later in the chapter about status codes.
System weight:	is four bytes making a register containing the current system weight achieved by summation of weight signals for all detected loadcells.
System exponent:	is two bytes making a register containing the system exponent, that indicates the resolution of the corresponding system weight as described later.

is done by sending a "Read Holding Register" request to the MCE2040 module as follows:

	HEX	ASCII (chars)		RTU (binary)
Header:	xx	:		none
Slave Adr.:	xx	x	x	xxxxxxxx
Function:	03	0	3	00000011
Start Adr.(H):	00	0	0	00000000
Start Adr. (L):	65	6	5	01100101
# of points (H):	00	0	0	00000000
# of points (L):	05	0	5	00000101
Error Check:	xx	LRC(2 bytes)		CRC(2 bytes)
Trailer:	xx	CR	LF	none

To this the MCE2040 module responds as follows:

	HEX	ASCII (chars)		RTU (binary)
Header:	xx	:		none
Slave Adr.:	xx	x	x	xxxxxxxx
Function:	03	0	3	00000011
Byte Count:	0A	0	A	00001010
Dat0(H)/LcRegister(MSB)	xx	x	x	xxxxxxxx
Dat0(L)/LcRegister(LSB)	xx	x	x	xxxxxxxx
Dat1(H)/SystemStatus(MSB)	xx	x	x	xxxxxxxx
Dat1(L)/SystemStatus(LSB)	xx	x	x	xxxxxxxx
Dat2(H)/SystemWeight(LSW)	xx	x	x	xxxxxxxx
Dat2(L)/SystemWeight(LSW)	xx	x	x	xxxxxxxx
Dat3(H)/SystemWeight(MSW)	xx	x	x	xxxxxxxx
Dat3(L)/SystemWeight(MSW)	xx	x	x	xxxxxxxx
Dat4(H)/SystemExponent(MSB)	xx	x	x	xxxxxxxx
Dat4(L)/SystemExponent(LSB)	xx	x	x	xxxxxxxx
Error Check:	xx	LRC(2 bytes)		CRC(2 bytes)
Trailer:	xx	CR	LF	none

3.2.3 Read LC-register, LC-status and LC-weight (grams)

Reading of the following registers for **i** number of loadcells (1-6):

LC register:	is two bytes making a bit register to indicate the connected load-cells detected during power-up. Hence bit 0-5 will be ON, if the corresponding loadcell addresses were detected during power-up.
LC status[i]:	is two bytes making a register containing current loadcell status for loadcell **i**. The meaning of each bit is explained later in the chapter about status codes.
LC weight[i]:	is four bytes making a register containing the current weight for loadcell **i** in grams.

is done by sending a "Read Holding Register" request to the MCE2040 module as follows (here for 1 loadcell):

	HEX	ASCII (chars)		RTU (binary)
Header:	xx	:		none
Slave Adr.:	xx	x	x	xxxxxxxx
Function:	03	0	3	00000011
Start Adr.(H):	00	0	0	00000000
Start Adr. (L):	0A	0	A	00001010
# of points (H):	00	0	0	00000000
# of points (L):	04	0	4	00000100
Error Check:	xx	LRC(2 bytes)		CRC(2 bytes)
Trailer:	xx	CR	LF	none

To this the MCE2040 module responds as follows:

	HEX	ASCII (chars)		RTU (binary)
Header:	xx	:		none
Slave Adr.:	xx	x	x	xxxxxxxx
Function:	03	0	3	00000011
Byte Count:	08	0	8	00001000
Dat0(H)/LcRegister(MSB)	xx	x	x	xxxxxxxx
Dat0(L)/LcRegister(LSB)	xx	x	x	xxxxxxxx
Dat1(H)/LcStatus[0](MSB)	xx	x	x	xxxxxxxx
Dat1(L)/LcStatus[0](LSB)	xx	x	x	xxxxxxxx
Dat2(H)/LcWeight[0](LSW)	xx	x	x	xxxxxxxx
Dat2(L)/LcWeight[0](LSW)	xx	x	x	xxxxxxxx
Dat3(H)/LcWeight[0](MSW)	xx	x	x	xxxxxxxx
Dat3(L)/LcWeight[0](MSW)	xx	x	x	xxxxxxxx
Error Check:	xx	LRC(2 bytes)		CRC(2 bytes)
Trailer:	xx	CR	LF	none

Depending on how many loadcell is to be read the following coherence applies between number of loadcells, *# of points* in the request telegram and *byte count* (length) in the response telegram:

Desired number of loadcells:	1	2	3	4	5	6
***# of points* in request:**	0x04	0x07	0x0A	0x0D	0x10	0x13
***Byte Count* in response:**	0x08	0x0E	0x14	0x1A	0x20	0x26

For example the request and response telegram will look as follows during request of data for 2 loadcells:

```
REQUEST                          HEX              ASCII (chars)            RTU (binary)
Header:                          xx               :                        none
Slave Adr.:                      xx               x          x             xxxxxxxx
Function:                        03               0          3             00000011
Start Adr.(H):                   00               0          0             00000000
Start Adr. (L):                  0A               0          A             00001010
# of points (H):                 00               0          0             00000000
# of points (L):                 07               0          7             00000111
Error Check:                     xx               LRC(2 bytes)             CRC(2 bytes)
Trailer:                         xx               CR         LF            none

RESPONSE                         HEX              ASCII (chars)            RTU (binary)
Header:                          xx               :                        none
Slave Adr.:                      xx               x          x             xxxxxxxx
Function:                        03               0          3             00000011
Byte Count:                      0E               0          E             00001110
Dat0(H)/LcRegister(MSB)          xx               x          x             xxxxxxxx
Dat0(L)/LcRegister(LSB)          xx               x          x             xxxxxxxx
Dat1(H)/LcStatus[0](MSB)         xx               x          x             xxxxxxxx
Dat1(L)/LcStatus[0](LSB)         xx               x          x             xxxxxxxx
Dat2(H)/LcWeight[0](LSW)         xx               x          x             xxxxxxxx
Dat2(L)/LcWeight[0](LSW)         xx               x          x             xxxxxxxx
Dat3(H)/LcWeight[0](MSW)         xx               x          x             xxxxxxxx
Dat3(L)/LcWeight[0](MSW)         xx               x          x             xxxxxxxx
Dat4(H)/LcStatus[1](MSB)         xx               x          x             xxxxxxxx
Dat4(L)/LcStatus[1](LSB)         xx               x          x             xxxxxxxx
Dat5(H)/LcWeight[1](LSW)         xx               x          x             xxxxxxxx
Dat5(L)/LcWeight[1](LSW)         xx               x          x             xxxxxxxx
Dat6(H)/LcWeight[1](MSW)         xx               x          x             xxxxxxxx
Dat6(L)/LcWeight[1](MSW)         xx               x          x             xxxxxxxx
Error Check:                     xx               LRC(2 bytes)             CRC(2 bytes)
Trailer:                         xx               CR         LF            none
```

When reading data for more loadcells the number of points in the request telegram is changed, after which the response telegram will be expanded in a similar way, as with the expansion from one to two loadcells (see above).

3.2.4 Read LC-register, LC-status, LC-weight and LC-exponent

Reading of the following registers for **i** number of loadcells (1-6):

LC register:	is two bytes making a bit register to indicate the connected load-cells detected during power-up. Hence bit 0-5 will be ON, if the corresponding loadcell addresses were detected during power-up.
LC status[i]:	is two bytes making a register containing current loadcell status for loadcell **i**. The meaning of each bit is explained later in the chapter about status codes.
LC weight[i]:	is four bytes making a register containing the current weight for loadcell **i**.
LC exponent[i]:	is two bytes making a register containing the loadcell exponent for loadcell **i**. This indicates the resolution of the corresponding loadcell weight as described later.

is done by sending a "Read Holding Register" request to the MCE2040 module as follows (here for 1 loadcell):

	HEX	ASCII (chars)		RTU (binary)
Header:	xx	:		none
Slave Adr.:	xx	x	x	xxxxxxxx
Function:	03	0	3	00000011
Start Adr.(H):	00	0	0	00000000
Start Adr. (L):	6E	6	E	01101110
# of points (H):	00	0	0	00000000
# of points (L):	05	0	5	00000101
Error Check:	xx	LRC(2 bytes)		CRC(2 bytes)
Trailer:	xx	CR	LF	none

To this the MCE2040 module responds as follows:

	HEX	ASCII (chars)		RTU (binary)
Header:	xx	:		none
Slave Adr.:	xx	x	x	xxxxxxxx
Function:	03	0	3	00000011
Byte Count:	0A	0	A	00001010
Dat0(H)/LcRegister(MSB)	xx	x	x	xxxxxxxx
Dat0(L)/LcRegister(LSB)	xx	x	x	xxxxxxxx
Dat1(H)/LcStatus[0](MSB)	xx	x	x	xxxxxxxx
Dat1(L)/LcStatus[0](LSB)	xx	x	x	xxxxxxxx
Dat2(H)/LcWeight[0](LSW)	xx	x	x	xxxxxxxx
Dat2(L)/LcWeight[0](LSW)	xx	x	x	xxxxxxxx
Dat3(H)/LcWeight[0](MSW)	xx	x	x	xxxxxxxx
Dat3(L)/LcWeight[0](MSW)	xx	x	x	xxxxxxxx
Dat4(H)/LcExponent[0](MSB)	xx	x	x	xxxxxxxx
Dat4(L)/LcExponent[0](LSB)	xx	x	x	xxxxxxxx
Error Check:	xx	LRC(2 bytes)		CRC(2 bytes)
Trailer:	xx	CR	LF	none

Depending on how many loadcell is to be read the following coherence applies between number of loadcells, *# of points* in the request telegram and *byte count* (length) in the response telegram:

Desired number of loadcells:	1	2	3	4	5	6
***# of points* in request:**	0x05	0x09	0x0D	0x11	0x15	0x19
***Byte Count* in response:**	0x0A	0x12	0x1A	0x22	0x2A	0x32

For example the request and response telegram will look as follows during request of data for 2 loadcells:

```
REQUEST                          HEX             ASCII (chars)            RTU (binary)
Header:                          xx              :                        none
Slave Adr.:                      xx              x         x              xxxxxxxx
Function:                        03              0         3              00000011
Start Adr.(H):                   00              0         0              00000000
Start Adr. (L):                  6E              6         E              01101110
# of points (H):                 00              0         0              00000000
# of points (L):                 09              0         9              00001001
Error Check:                     xx              LRC(2 bytes)             CRC(2 bytes)
Trailer:                         xx              CR        LF             none

RESPONSE                         HEX             ASCII (chars)            RTU (binary)
Header:                          xx              :                        none
Slave Adr.:                      xx              x         x              xxxxxxxx
Function:                        03              0         3              00000011
Byte Count:                      12              1         2              00010010
Dat0(H)/LcRegister(MSB)          xx              x         x              xxxxxxxx
Dat0(L)/LcRegister(LSB)          xx              x         x              xxxxxxxx
Dat1(H)/LcStatus[0](MSB)         xx              x         x              xxxxxxxx
Dat1(L)/LcStatus[0](LSB)         xx              x         x              xxxxxxxx
Dat2(H)/LcWeight[0](LSW)         xx              x         x              xxxxxxxx
Dat2(L)/LcWeight[0](LSW)         xx              x         x              xxxxxxxx
Dat3(H)/LcWeight[0](MSW)         xx              x         x              xxxxxxxx
Dat3(L)/LcWeight[0](MSW)         xx              x         x              xxxxxxxx
Dat4(H)/LcExponent[0](MSB)       xx              x         x              xxxxxxxx
Dat4(L)/LcExponent[0](LSB)       xx              x         x              xxxxxxxx
Dat5(H)/LcStatus[1](MSB)         xx              x         x              xxxxxxxx
Dat5(L)/LcStatus[1](LSB)         xx              x         x              xxxxxxxx
Dat6(H)/LcWeight[1](LSW)         xx              x         x              xxxxxxxx
Dat6(L)/LcWeight[1](LSW)         xx              x         x              xxxxxxxx
Dat7(H)/LcWeight[1](MSW)         xx              x         x              xxxxxxxx
Dat7(L)/LcWeight[1](MSW)         xx              x         x              xxxxxxxx
Dat8(H)/LcExponent[1](MSB)       xx              x         x              xxxxxxxx
Dat8(L)/LcExponent[1](LSB)       xx              x         x              xxxxxxxx
Error Check:                     xx              LRC(2 bytes)             CRC(2 bytes)
Trailer:                         xx              CR        LF             none
```

When reading data for more loadcells the number of points in the request telegram is changed, after which the response telegram will be expanded in a similar way, as with the expansion from one to two loadcells (see above).

3.3 Modbus Test Mode

It is possible to place the MCE2040 module in Modbus test mode. This is done as explained in the hardware description using v.h.a. Sw1.8. In Modbus test mode all transferred weight indications are set to 123456 independent of the current load (hex with MSB first):

	32 bit signed integer	32 bit floating point
Weight indication:	00 01 E2 40	47 F1 20 00

3.4 Exponent

It is possible using two of the Modbus telegrams to read weight values using the loadcell resolution. This resolution can be different from grams. If these telegrams are used the weight value must be used together with its corresponding exponent. In certain cases (where the exponent is less than 0) this allows to read the weight value using a resolution that is higher than grams. The following coherence applies between the exponent and the weight value resolution:

Exponent [Decimal]	**Exponent [Hexadecimal]**	**Conversion factor to grams**	**SI unit**
-3	0xFFFD	$*10^{-3}$	mg
-2	0xFFFE	$*10^{-2}$	
-1	0xFFFF	$*10^{-1}$	
0	0x0000	$*10^{0}$	gram
1	0x0001	$*10^{1}$	
2	0x0002	$*10^{2}$	
3	0x0003	$*10^{3}$	Kg
4	0x0004	$*10^{4}$	
5	0x0005	$*10^{5}$	
6	0x0006	$*10^{6}$	ton

4) ZEROING AND CALIBRATION

4.1 Zeroing procedure

Zeroing of the system should be performed as follows:

1) The weighing arrangement should be empty and clean.
2) Read and store the actual weight values for the connected loadcells in corresponding zeroing registers.
3) After this the actual weight for loadcell **X** can be calculated as:

   LcGross(**X**) = LcWeight(**X**) – LcZero(**X**)

   and the system weight (uncalibrated) for the connected loadcells is calculated as:

   SystemWeight = LcGross(0) + LcGross(1) + ….

If only summed signals are used, the it is only necessary to store the actual summed weight in one zeroing register and thereafter calculate the system weight as actual summed weight minus zero register.

4.2 Calibration procedure

Fine calibration of the system should be performed as follows:

1) Check that the weighing arrangement is empty, and that the gross weight is zero. Zero if necessary.
2) Place a known load (calibration weight) on the weighing arrangement.
3) Calculate the calibration factor that should be multiplied on the system weight in order to achieve correct showing as:

   Calibration factor = (Calibration weight)/(Actual showing)

   After this the determined calibration factor is used to calculate the calibrated weight as follows:

   Sys.Weight(Calibrated) = Calibration factor * Sys.Weight(Uncalibrated)

   If the determined calibration factor falls outside the interval 0.9 to 1.1 it is very likely that there is something wrong with the mechanical part of the system.

5) STATUS CODES

Status codes are shown as a 4 digit hex number. If more than one error condition is present the error codes are OR'ed together.

CODE (Hex)	CAUSE
0001	**Invalid/missing 'sample' ID** Bad connection between communication module and loadcell module. Not all telegrams from communication module are received in loadcell module.
0002	**Loadcell timeout** Check that the loadcell is connected to the loadcell module.
0004	**Loadcell not synchronised** Bad connection between loadcell and loadcell module, or very powerful under- or overload.
0008	**Hardware synchronisation error** Loadcell samples are not synchronized. Cable between loadcell modules shorted or disconnected.
0010	**Power failure** Supply voltage to loadcells is to low.
0020	**Overflow in weight calculation** Internal error in loadcell module.
0040	**Invalid/missing 'latch' ID** Bad connection between communication module and loadcell module. Not all telegrams from communication module is received in loadcell module.
0080	**No answer from loadcell module** No data is received from this loadcell module. This can be caused by the removal of the loadcell module, no power to the module or that the connection between loadcell module and communication module is broken.
0100	*Reserved for future use*
0200	*Reserved for future use*
0400	*Reserved for future use*
0800	**No loadcell modules answer** Bad connection between communication module and loadcell module. Not all telegrams from communication module are received in loadcell module.
1000	*Reserved for future use*
2000	*Reserved for future use*
4000	*Reserved for future use*
8000	*Reserved for future use*

6) INSTALATION OF SYSTEM

6.1 Checklist during installation

During installation of the system the following should be checked:

1) The loadcells are mounted mechanically and connected to the MCE2040 module using their corresponding loadcell interface module (MCE9610/MCE2010). The loadcell addresses are set using the DIP-switches (Sw1.5-Sw1.8) on the MCE9610/MCE2010 modules, so that they forth running from address 0 (0-5).

2) The MCE2040 module is connected to the Modbus network. If RS485 or RS422 communication is used the position of jumpers S1 and S2 should be checked. Possible termination should be made if necessary at the MCE2040 Modbus slave.

3) The Modbus communication address (0-31), the Modbus mode (ASCII/RTU), Modbus data format (SI32/FP32) and Modbus test mode (OFF/ON) should be set using the MCE2040 module switches; Sw1.1- Sw1.5, Sw1.6, Sw1.7 and Sw1.8 respectively. Power is applied.

4) Verify that the TXBB LED on the MCE2040 module is lit and that the TXBB LED's on the loadcell modules (MCE9610/MCE2010) are also lit (can flash slightly).

5) Verify that the TxExt LED on the MCE2040 module flashes each time it sends a Modbus telegram as a response to requests from the Modbus master.

6) Verify that the MCE2040 module has found the correct loadcells and that no error codes are indicated for these.

7) Verify that every loadcell gives a signal by placing a load directly above each loadcell one after the other (possibly with a known load).

The system is now installed and a possible zero and fine calibration is made as described earlier. Finally verify that the weighing system(s) returns a value corresponding to a known actual load.

Note that in the above checklist no consideration has been made on which functions and how they are implemented on the Modbus master.

7) MCE2040 HARDWARE DESCRIPTION

7.1 DIP-switch settings

The MCE2040 module is equipped with one DIP-switch block. DIP-switch block 1 has the following function:

SWITCH	FUNCTION
Sw1.1-Sw1.5	**Modbus communication address** Used to set the Modbus communication address (0-31) of the MCE2040 module. The address is set as the DIP-switches are binary coded, so that Sw1.1 is LSBit and Sw1.5 is MSBit. **Note** that these switches are only read at power-up.
Sw1.6	**Modbus mode** OFF: **ASCII** mode ON: **RTU** mode **Note** that this switch is only read at power-up.
Sw1.7	**Dataformat** The switch determines if the weight indications in the communication are in ”32 bit signed integer” or in ”IEEE754 floating point” format. OFF: **SI32** mode (32 bit signed integer) ON: **FP32** mode (IEEE754 floating point) **Note** that this switch is only read at power-up.
Sw1.8	**Modbus test mode** OFF: Normal operation ON: Modbus test mode **Note** that this switch is only read at power-up.

7.2 Light emitting diodes

The MCE2040 module is equipped with 2 light emitting diodes (LED's). These LED's have the following function:

LED	FUNCTION
TxBB (Green LED)	**Communication with loadcells** The MCE2040 communicates with the loadcells
TxExt (Green LED)	**Extern bus TxD (Transmit Data)** The MCE2040 module transmits data across the MODBUS.

7.3 Jumpers

The MCE2040 module is equipped with 2 jumpers. These jumpers have the following function:

JUMPER	FUNCTION
S1	**Selection of RS422 or RS485** This jumper is placed depending on whether the module should use RS422 or RS485 communication on the external bus: RS422 (4 wire): **OFF** RS485 (2 wire): **ON (shorts R- to T-)**
S2	**Selection of RS422 or RS485** This jumper is placed depending on whether the module should use RS422 or RS485 communication on the external bus: RS422 (4 wire): **OFF** RS485 (2 wire): **ON (shorts R+ to T+)**

7.4 EE-bus connector

The MCE2040 module is equipped with a 10 pole connector for connection to the Eilersen Electric EE-bus. Hereby connection to the individual MCE9610/MCE2010 loadcell modules as well as to the power supply for the MCE2040 module is achieved. The connection is made using a ribbon cable with mounted connectors for the individual modules. The 10 pole connector (J1) has the following connections:

J1 Connector	Function
J1.1-J1.2	RS485-B (**negative** line)
J1.3-J1.4	RS485-A (**positive** line)
J1.5-J1.6	0VDC (Gnd1)
J1.7-J1.8	+24VDC (Vin1)
J1.9-J1.10	Not used

7.5 MCE9601 connection module

The MCE2040 module is normally connected to loadcells and power supply through a ribbon cable and a MCE9601 connection module. The MCE9601 module has the following connections:

MCE9601	Supply	Terminal
24V	+24VDC	(+24VDC)
A		A (R+/T+)
B		B (R-/T-)
Gnd	0VDC	Gnd

7.6 RS232/RS422/RS485 external communication connector

The MCE2040 module is equipped with a 9 pole sub-D connector (female) for connection to the external communication network. The connector (J2) has the following connections:

Pin	Designation	RS232 Connection	RS485 Connection	RS422 Connection
J2.1	RS422.R- (RS485.B)	-	Modbus.RS485.B (-)	Modbus.RS422.T-
J2.2	RS232.RX	Modbus.RS232.TX	-	-
J2.3	RS232.TX	Modbus.RS232.RX	-	-
J2.4	RS422.T+ (RS485.A)	-	Modbus.RS485.A (+)	Modbus.RS422.R+
J2.5	GND	Modbus.RS232.GND	Modbus.RS485.GND	Modbus.RS422.GND
J2.6	RS422.R+ (RS485.A)	-	Modbus.RS485.A (+)	Modbus.RS422.T+
J2.7	-	-	-	-
J2.8	-	-	-	-
J2.9	RS422.T- (RS485.B)	-	Modbus.RS485.B (-)	Modbus.RS422.R-

7.7 Digital I/O connector

The MCE2040 module is equipped with a digital I/O connector (J3) for possible connection of 2 digital inputs and 2 digital outputs. With the installed program the following functions are implemented on the I/O signals:

Pin	Designation	Connection
J3.1	IN1	Not used
J3.2	GND	
J3.3	IN2	Not used
J3.4	24VDC OUT	
J3.5	OUT1	Not used
J3.6	GND	
J3.7	OUT2	Not used

Note that J3.1 is placed next to the **TxExt** light emitting diode.

7.8 Internal JTAG connector

The MCE2040 module is equipped with an internal JTAG connector. The connector (J5) is used exclusively by Eilersen Electric A/S during download of software to the Cygnal processor.

7.9 MCE2040 Update time

All loadcells are sampled over a period of 200 mS. The hereby found loadcell signals are used in data communication until new signals are achieved when the next sample period expires. The transmission time depends on the selected baudrate, the selected mode, and the length of the individual MODBUS telegrams.