# Serial Digital Interface of HYGROSENS ASIC

## Protocol of

## I$^2$C Interface
## SPI Interface

## Index

## Manufacturer Informations

**Manufacturer Informations**

HYGROSENS INSTRUMENTS **GmbH**

Maybach Straße 2
79843 Löffingen

Postfach 1054
79840 Löffingen

| | |
|---|---|
| Phone: | 0049 7654 808969-0 |
| Fax: | 0049 7654 808969-9 |
| Internet: | www.hygrosens.com |
| Email: | info@hygrosens.com |

**Service and Orders**

Service and Orders

| | |
|---|---|
| Phone: | 0049 7654 808969-0 |
| Fax: | 0049 7654 808969-9 |
| Internet: | www.hygrosens.com |
| Email: | sales@hygrosens.com |

Technical Support and Calibration Service

| | |
|---|---|
| Phone: | 0049 7654 808969-0 |
| Fax: | 0049 7654 808969-9 |
| Internet: | www.hygrosens.com |
| Email: | support@hygrosens.com |

## 1 General Description

The HYGROSENS ASIC includes a serial digital interface, which is able to communicate using two different communication protocols – I$^2$C and SPI –communication. The serial digital interface allows the programming of the EEPROM to configure the application mode of the ASIC and to calibrate the conditioning equation. Furthermore it makes possible the read out of the conditioning result of measurand and both temperatures as digital 15 Bit values. The ASIC works always as slave. The used communication protocol has to be chosen by programming EEPROM. There are also commands to change the valid communication mode. Only one communication protocol is valid at one time. The implemented commands are available in all communication modes but are divided in two sets with different validity. Commands, which change the configuration of the device, are suppressed in “Normal Operation Mode” (NOM) and are available only after changing to “Command Mode” (CM). A non-configured device, identified by a CRC Error regarding EEPROM contents, starts up in a special mode so that communication by any of the available protocols is possible. After power-on a 20ms-time-slot is opened to start Service-communication. If no Service-communication is detected, the serial digital interface changed in a mode in which it can receive commands in I$^2$C-mode as well as in SPI-mode. If it is necessary to read out data from a non-configured device (transmission by slave), the used communication protocol has to be explicitly defined by sending a certain command. Note that the HYGROSENS ASIC also do not send acknowledges in non-configured mode if I$^2$C-communication is used. If the HYGROSENS ASIC receives a valid command at the serial digital interface the measurement cycle is interrupted because the internal micro-controller has to execute the requested command routine. An exception is a read-request. The HYGROSENS ASIC answers without interrupting the measurement cycle. This makes possible the read-out of the digital conditioning results during measurement cycle. A command consists of an address byte and a command byte. Additionally the commands for writing the EEPROM or its mirror in the RAM enclose two data bytes. This is independent from used communication protocol. To read data from the HYGROSENS ASIC (e.g. EEPROM contents) usually a certain command has to be sent to transfer this data into the output register of the serial interface. Thereafter the READ command consisting of the address byte with the read bit set is used to get this data. The data are transmitted continuously repeated as long as the master send the clock and do not abort by generating a stop condition. Again this is independent from used communication protocol. During the measurement cycle the HYGROSENS ASIC transfers the conditioning results into the output registers of the serial digital interface. There are three registers for humidity and both temperatures. The activation of these registers and consequently of the transmitted data have to be set by EEPROM programming. The results are sent in the sequence humidity, temperature 1 and temperature 2 according to the register activation if the master sends a read-request.

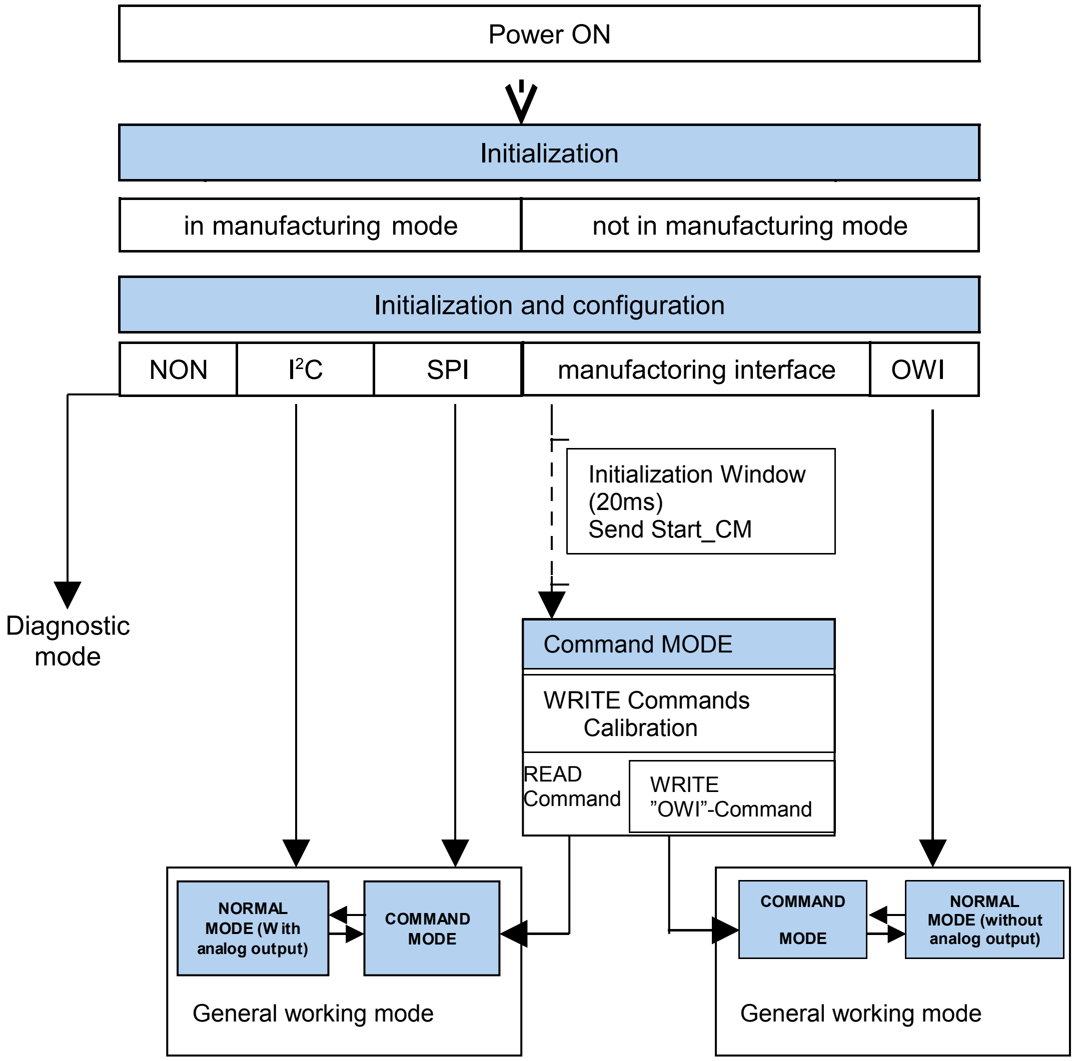

*Fig. 1: Possible Communication Configuration*

## 2 I²C Protocol

For I²C communication a data line (SDA) and a clock line (SCL) are required. The I²C protocol used is defined as follows:

### Idle period

During inactivity of the bus SDA and SCL are pulled-up to supply voltage VDDA.

### Start condition

A high to low transition on SDA while SCL is at high level indicates a start condition. Every command has to be initiated by a start condition sent by a master. A master can always generate a start condition.

### Stop condition

A low to high transition on SDA while SCL is at high level indicates a stop condition. A command has to be closed by a stop condition to start processing the command routine inside the IC.

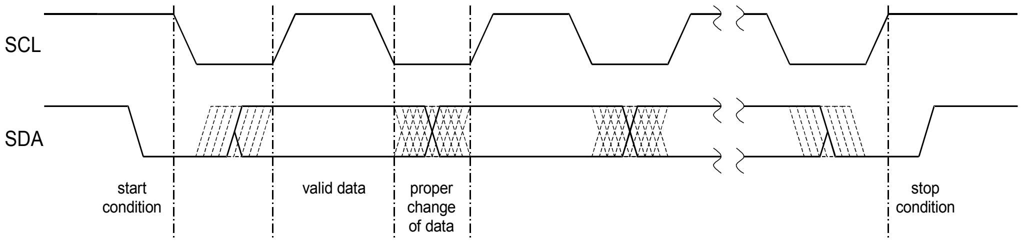

*Fig. 2:* ***Principles of I²C protocol***

### Valid data

Data is transmitted in Bytes (8 Bits) starting with the most significant bit (MSB). Each byte transmitted is followed by an acknowledge bit. Transmitted bits are valid if after a start condition SDA keeps at constant level during high period of SCL. The SDA level has to change only when clock signal at SCL is low.

### Acknowledge

Acknowledge after transmitted byte is obligatory. The master must generate an acknowledge related clock pulse. The receiver (slave or master) pulls-down the SDA line during acknowledge clock pulse. If no acknowledge is generated by the receiver a transmitting slave will set inactive. A transmitting master can abort the transmission by generating a stop condition and may repeat the command.

A receiving master must signal the end of transfer to the transmitting slave by not generating an acknowledge related clock pulse at SCL.

The HYGROSENS ASIC as a slave changes to inactive interface mode during processing internal command routines started by a previously sent command.

### Addressing

Every slave connected to the I$^2$C-bus responds to a certain address. After generating the start condition the master sends the address byte containing a 7bit-address followed by a data direction bit (R/W). A ‘0’ indicates a transmission from master to slave (WRITE), a ’1’ indicates a data request (READ).

The addressed slave answers with an acknowledge, all other slaves connected with the I$^2$C-bus ignore this communication.

The general HYGROSENS ASIC slave address is 0x78 (7bit). As the address is completed with the R/W-Bit as LSB the address Byte (whole 8 bit) for reading data is 0xA1 for reading or 0xA0 for writing data to the ASIC.

By EEPROM programming it is possible to allocate and activate an additional arbitrary slave address to every single device. In this case the device recognizes communication on both addresses, on the general one and on the activated one.

### Write operation

During transmission from master to slave (WRITE) after the address byte follows a command byte and, depending on the transmitted command, optional 2 data bytes. The internal micro-controller evaluates the received command and processes the related routine. The available commands are described in chapter 4.5.

| | | | | | | optional | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| start condition | slave address | R/W 0 | A | command byte | A | 1$^{st}$ data byte | A | 2$^{nd}$ data byte | A | stop condition |
| send by: master | | | slave | master | slave | master | slave | master | slave | master |

*Fig. 3:* ***Write operation I$^2$C***

### Read operation

After a data request from master to slave by sending an address-byte including a set data direction bit the slave answers by sending data from the activated interface output registers. The master must generate the transmission clock on SCL, the acknowledges after each data byte (except after the last one) and finally the stop condition.

A data request is answered by the interface module itself and does consequently not interrupt the current process of the internal micro-controller. The data in the activated registers is sent continuously until a stop condition is detected, after transmitting all available data the slave starts repeating the data.

During running measurement cycle data is incessantly updated with conditioning results. To get other data from slave (e.g. EEPROM contents) usually a certain command has to be sent before the data request to initiate the transfer of this data to the interface output registers. This command does interrupt the current process of the internal microprocessor and consequently also a running measurement cycle.

| | | | | | | optional | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| start condition | slave address | R/W 1 | A | 1st data byte | A | 2nd data byte | A | … nth data byte | stop condition | |
| sent by: master | | | slave | slave | master | slave | master | slave | Master | |

*Fig. 4:* ***Read operation I$^2$C***

# SERIAL DIGITAL INTERFACE

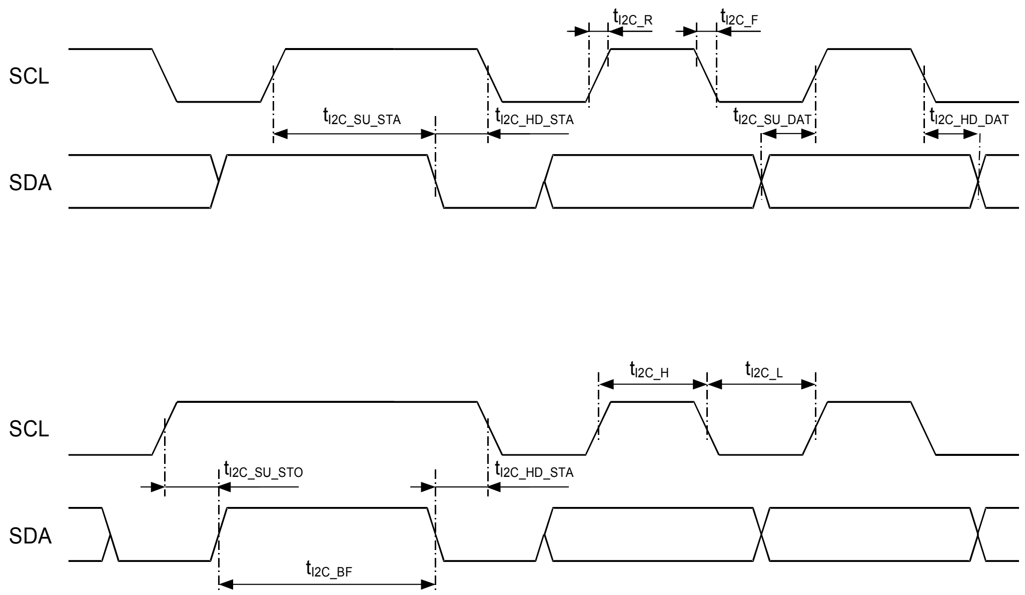

***Fig. 5:** Timing I$^2$C protocol*

| Nr. | Parameter | Symbol | min | typ | max | Unit |
|---|---|---|---|---|---|---|
| 1 | SCL Clock frequency | $f_{SCL}$ | | | 400 | kHz |
| 2 | Bus free time betw. start and stop condition | $t_{I2C\_BF}$ | 1.3 | | | µs |
| 3 | Hold time start condition | $t_{I2C\_HD\_STA}$ | 0.6 | | | µs |
| 4 | Setup time repeated start condition | $t_{I2C\_SU\_STA}$ | 0.6 | | | µs |
| 5 | Low period SCL/SDA | $t_{I2C\_L}$ | 1.3 | | | µs |
| 6 | High period SCL/SDA | $t_{I2C\_H}$ | 0.6 | | | µs |
| 7 | Data hold time | $t_{I2C\_HD\_DAT}$ | 0 | | | µs |
| 8 | Data setup time | $t_{I2C\_SU\_DAT}$ | 0.1 | | | µs |
| 9 | Rise time SCL/SDA | $t_{I2C\_R}$ | | | 0.3 | µs |
| 10 | Fall time SCL/SDA | $t_{I2C\_F}$ | | | 0.3 | µs |
| 11 | Setup time stop condition | $t_{I2C\_SU\_STO}$ | 0.6 | | | µs |
| 12 | Noise interception SDA, Spikes are suppressed | $t_{I2C\_NI}$ | | | 50 | ns |

***Table 1** Timing I$^2$C protocol*

### Example

Factory setting for configuation of I$^2$C interface:

The following example is valid if the HYGROSENS USB Interface in conjunction with a standard terminal software is used:

| | |
|---|---|
| t_050 | "set switch-off time to 50 ms" |
| t01200 | "select trigger channel KS5V with a trigger time of 200 ms" |
| iwt7800172 | "write command 0x72 to address 0x78 with trigger" |
| iw_7800152 | "configure SIF to communication mode I$^2$C" |
| iw_7800171 | "start Normal Operation Mode" |
| t00000 | "turn OFF the trigger channel ==> KS5V = OFF" |

The command 0x52 is configurating the serial interface for I$^2$C communication. After starting the normal operating mode the serial interface is updated continously with the actual conditioned output values. The analog value is still available.

To read out the values

| | |
|---|---|
| ir_78004 | “read four bytes in the order MSB humidity LSB humidity,<br>MSB temperature, LSB temperature” |

## 3 Synchronous serial peripheral interface (SPI)

During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows individual selection of a slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. HYGROSENS ASIC's SPI slave interface supports all combinations of clock phase (CPHA) and polarity (CPOL). Slave CPOL and CPHA has to be programmed to master adjustments before beginning of transmission. RAM/EEPROM register 0x17$_{hex}$ contains initialization of SPI interface:

| | |
|---|---|
| SFGSIF: SIFMD | = interface mode SPI or I2C |
| SFGSIF: SPICKP | = clock polarity CPOL |
| SFGSIF: SPICKE | = clock phase CPHA |

### SPI Clock Phase and Polarity Controls

Software can select any of four combinations of serial clock (SCK) phase and polarity using two bits in the SPI control register (SPCR). The clock polarity is specified by the CPOL control bit, which selects an active high or active low clock and has no significant effect on the transfer format. The clock phase (CPHA) control bit selects one of two fundamentally different transfer formats.

### CPHA Equals Zero Transfer Format

Fig. 12 shows a timing diagram of an SPI transfer where CPHA is zero. Two waveforms are shown for SCK: one for CPOL equals zero and another for CPOL equals one. The diagram may be interpreted as a master or slave timing diagram since the SCK, master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signals the output from the master. The /SS line is the slave select input to the slave.

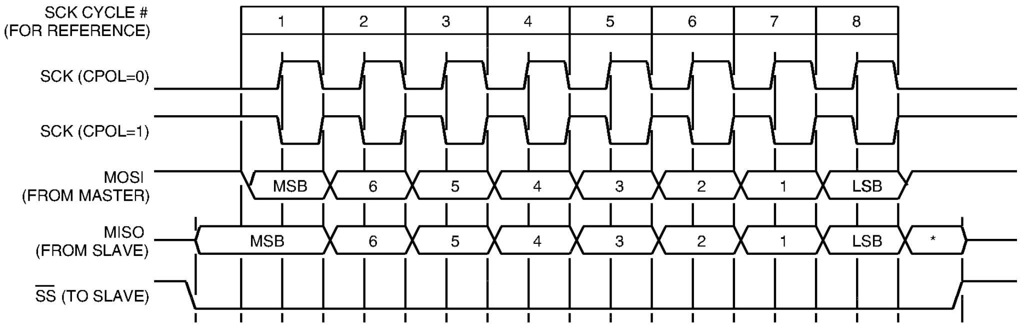

*Fig. 6: CPHA Equals Zero SPI Transfer Format – principle transfer illustration*

### CPHA Equals One Transfer Format

Fig. 13 shows a timing diagram of an SPI transfer where CPHA is one. Two waveforms are shown for SCK: one for CPOL equals zero and another for CPOL equals one. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO, and MOSI pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The /SS line is the slave select input to the slave.

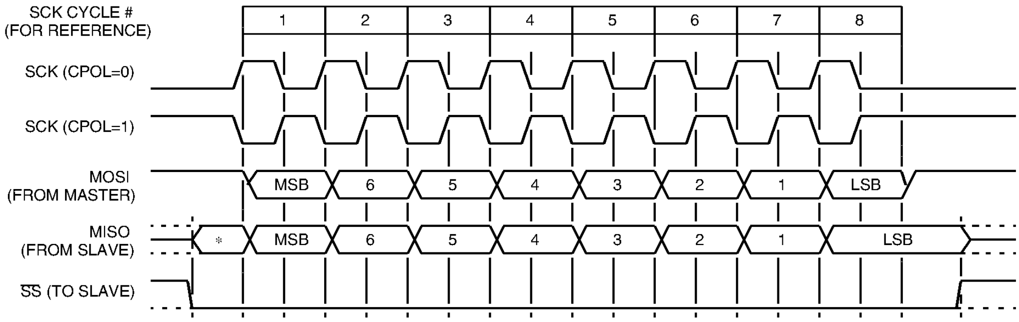

*Fig. 7: CPHA Equals One SPI Transfer Format – principle transfer illustration*

When CPHA equals zero, the /SS line must be negated and reasserted between each successive serial byte. Also, if the slave writes data to the SPI data register (SPDR) while /SS is active low, a write-collision error results.

When CPHA equals one, the /SS line may remain active low between successive transfers (can be tied low at all times). This format is sometimes preferred in systems having a single fixed master and a single slave driving the MISO data line.

### SPI Pin Signals

There are four I/O pin signals associated with SPI transfers: the SCK, the MISO data line, the MOSI data line, and the active low /SS pin. When the master initiates a transfer, eight clock cycles are automatically generated on the SCK pin. The SCK pin is an input and the clock signal from the master synchronizes the data transfer between the master and slave devices. Slave devices ignore the SCK signal unless the /SS pin is active low. In both the master and slave SPI devices, data is shifted on one edge of the SCK signal and is sampled on the opposite edge where data is stable. Edge polarity is determined by the SPI transfer protocol.

The MISO and MOSI data pins are used for transmitting and receiving serial data. At slave MOSI is the data input line, and MISO is the data output line. One by master selected slave device optionally drives data out its MISO pin to the MISO master pin. The automatic control of the direction of these pins makes reconfiguration through external logic unnecessary when a new device becomes the master.

The /SS pin behaves differently on master and slave devices. On a slave device, this pin is used to enable the SPI slave for a transfer. If the /SS pin of a slave is inactive (high), the device ignores SCK clocks and keeps the MISO output pin in the high-impedance state.

### Beginning and Ending SPI Transfers

A transfer includes the eight SCK cycles plus an initiation period at the beginning and ending period of the transfer. The details of the beginning and ending periods depend on the CPHA format selected and whether the SPI is configured as a master or a slave. The initiation delay period is also affected by the SPI clock rate selection when the SPI is configured as a master.

It may be useful to refer to the transfer format illustrated in Fig. 12 and Fig. 13 to understand how the beginning and ending details fit into a complete transfer operation.

### Transfer Beginning Period (Initiation Delay)

All SPI transfers are started and controlled by a master SPI device. As a slave the transfer to begin with the first SCK edge or the falling edge of /SS, depending on the CPHA format selected. When CPHA equals zero, the falling edge of /SS indicates the beginning of a transfer. When CPHA equals one, the first edge on the SCK indicates the start of the transfer. In either CPHA format, a transfer can be aborted by taking the /SS line high, which causes the SPI slave logic and bit counters to be reset. The SCK rate selected has no effect on slave operations since the clock from the master is controlling transfers.

CPHA has no effect on the delay to the start of the transfer, but it does affect the initial state of the SCK signal. When CPHA equals zero, the SCK signal remains inactive for the first half of the first SCK cycle. When CPHA equals one, the first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The SPI clock rate (selected by SPR[1:0]) affects the delay from the write to SPDR and the start of the SPI transfer (see Fig. 14). The internal SPI clock in the master is a free-running derivative of the internal MCU clock. Since the SPI clock is free-running, there is an uncertainty about the write operation to SPDR will occur relative to the slower SCK. This uncertainty causes the variation in the initiation delay shown in fig. 8.

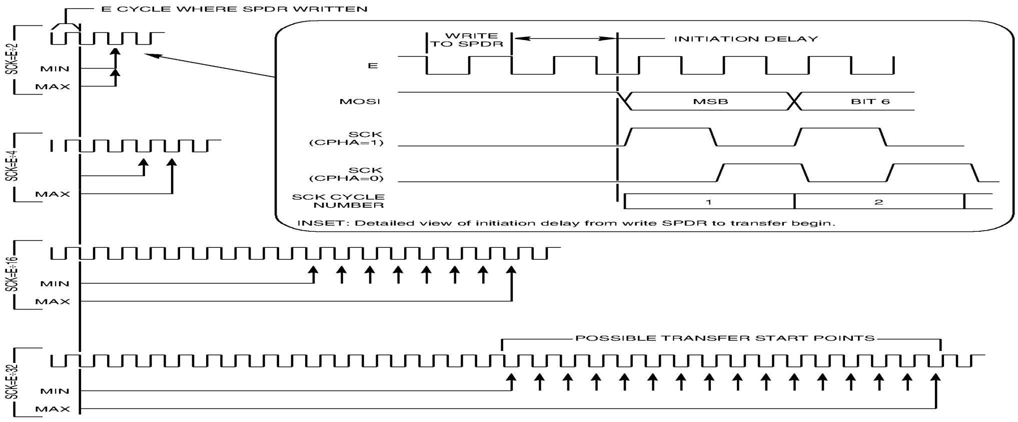

*Fig. 8: Delay from Write SPDR to Transfer Start (Master)*

### Transfer Ending Period

An SPI transfer is technically complete when the SPIF flag is set, but, depending on the configuration of the SPI system, there may be additional tasks. Because the SPI bit rate does not affect timing of the ending period, only the fastest rate will be considered in discussions of the ending period.

When the SPI is configured as a master, SPIF is set at the end of the eighth SCK cycle. When CPHA equals one, SCK is inactive for the last half of the eighth SCK cycle. Fig. 15 shows the transfer ending period for a master. The SCK waveforms in this figure show only the CPOL equals zero case, since clock polarity does not affect timing of the ending period.

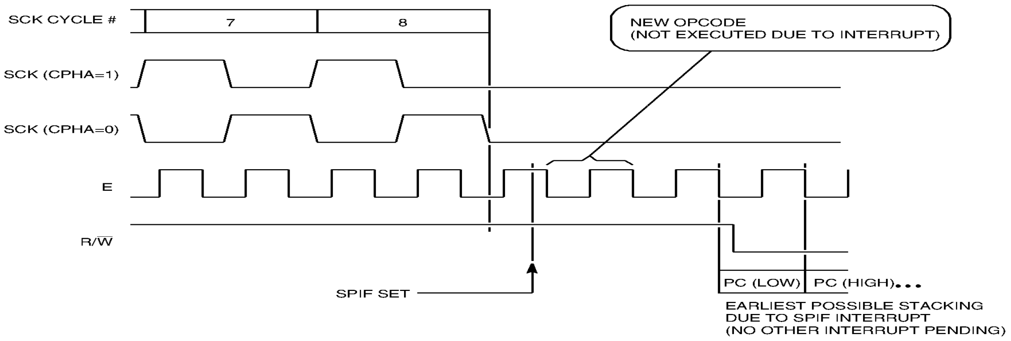

*Fig. 9: Transfer Ending for an SPI Master*

When the SPI is operating as a slave, the ending period is different because the SCK line can be asynchronous to the MCU clocks of the slave and because the slave does not have access to as much information about SCK cycles as the master. For example, when CPHA equals one, where the last SCK edge occurs in the middle of the eighth SCK cycle, the slave has no way of knowing when the end of the last SCK cycle is. For these reasons, the slave considers the transfer complete after the last bit of serial data has been sampled, which corresponds to the middle of the eighth SCK cycle. A synchronization delay is required so the setting of the SPIF flag is properly positioned relative to the internal clock of the slave. Fig. 16 shows the ending period for a slave. The SCK waveforms in this figure show only the CPOL equals zero case, since clock polarity does not affect timing of the ending period.

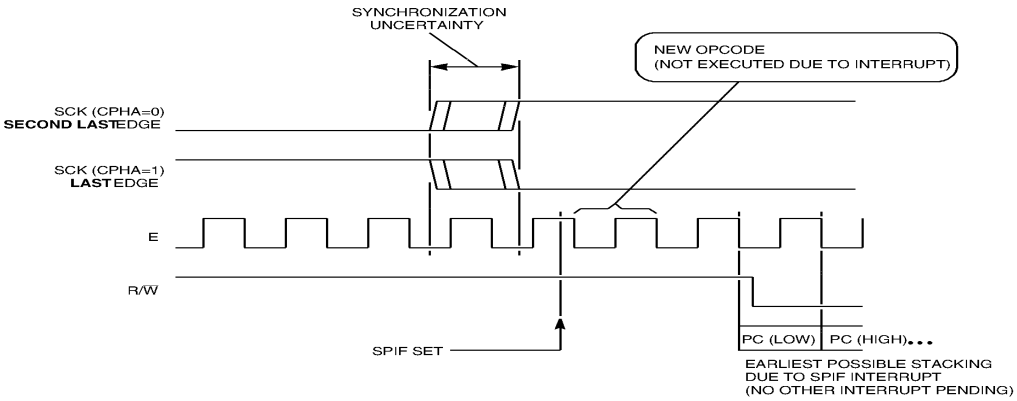

*Fig. 10: Transfer Ending for an SPI Slave*

When CPHA equals zero, there is a potential problem that can be avoided by proper software but is sometimes overlooked. The SPIF flag is set at the end of a transfer, but the slave is not permitted to write new data to the SPDR while the /SS line is still low. If the master device is busy, the /SS line to the slave can remain low longer than the slave expects.

## 4 Interface Commands

All implemented commands are available in all communication modes – I²C and SPI. A received valid command interrupts the internal micro-controller and initiates the processing of a command routine.

| Com-mand (HEX) | Data | Command | Remarks | Processing time System clock 2MHz |
|---|---|---|---|---|
| 01 | | START_CYC_EEP | Start measurement cycle including initialization from EEPROM | 350µs |
| 02 | | START_CYC_RAM | Start measurement cycle including initialization from RAM | 220µs |
| 10 .. 1F | | READ_RAM0 | Read data from RAM address 00 .. 0F<br>Writes data from RAM to SIF Output Registers<br>Usually followed by Read operation | 50µs |
| 20 .. 2F | | READ_RAM1 | Read data from RAM address 10 .. 1F<br>Writes data from RAM to SIF Output Register<br>Usually followed by Read operation | 50µs |
| 30 .. 3F | | READ_EEP0 | Read data from EEPROM address 00 .. 0F<br>Writes data from EEPROM to SIF Output Register<br>Usually followed by Read operation | 50µs |
| 40 .. 4F | | READ_EEP1 | Read data from EEPROM address 10 .. 1F<br>Writes data from EEPROM to SIF Output Register<br>Usually followed by Read operation | 50µs |
| **5*-commands do not change EEPROM or RAM configuration, used for communication to devices with unknown configuration** | | | | |
| 50 | | CFG_SIF_TO_OWI | Configure SIF to Communication Mode OWI | 50µs |
| 51<br>55<br>59<br>5D | | CFG_SIF_TO_SPI | Configure SIF to Communication Mode SPI<br>51 SET_SIF_2_SPI SPI (CKE = 0, CKP = 0)<br>55 SET_SIF_2_SPI SPI (CKE = 0, CKP = 1)<br>59 SET_SIF_2_SPI SPI (CKE = 1, CKP = 0)<br>5D SET_SIF_2_SPI SPI (CKE = 1, CKP = 1) | 50µs |
| 52 | | CFG_SIF_TO_I2C | Configure SIF to Communication Mode I²C | |
| 60 | 2 Byte | SET_DAC | Set Output DAC to value defined by data bytes [0x0 … 0x7FF] | 100µs |
| 70 | | START_OM | Start Open Mode<br>In Open Mode the command set is restricted!<br>Change to Command Mode is possible | 50µs |
| 71 | | START_NOM | Start Normal Operation Mode (NOM)<br>In Normal Operation Mode the command set is restricted!<br>Change to Command Mode or Open Mode is NOT possible | |
| 72 | | START_CM | Start Command Mode (CM)<br>Complete command set!<br>Change to Open Mode or Normal Operation Mode is possible | |

*Table 2: Restricted command set for serial digital interface – commands are always valid*

During processing time the digital serial interface (SIF) is disabled, transmitted commands are ignored. The processing time depends from the internal system clock frequency, which is usually 2MHz but is changeable by EEPROM programming. The commands are divided in two sets with different validity. Commands, which change the configuration of the device, are ignored in NOM and are available only after changing to CM.

## SERIAL DIGITAL INTERFACE

| Com-mand (HEX) | Data | Command | Remarks | Processing time System clock 2MHz / EEPROM-Progr. Steps |
|---|---|---|---|---|
| 80 .. 8F | 2 Byte | WRITE_RAM0 | Write data to RAM address 00 .. 0F | 50µs |
| 90 .. 9F | 2 Byte | WRITE_RAM1 | Write data to RAM address 10 .. 1F | 50µs |
| A0 .. AF | 2 Byte | WRITE_EEP0 | Write data to EEPROM address 00 .. 0F | 12.5ms / 1 |
| B0 .. BF | 2 Byte | WRITE_EEP1 | Write data to EEPROM address 10 .. 1F | 12.5ms / 1 |
| C0 | | COPY_EEP2RAM | Copy Contents of EEPROM to RAM<br>Restores EEPROM Configuration in RAM | 130µs |
| C3 | | COPY_RAM2EEP | Copy Contents of RAM to EEPROM<br>Stores RAM Configuration to EEPROM<br>(signature is copied without check!!!) | 400ms / 32 |
| C8 | | GET_EEP_SIGN | Calculates EEPROM Signature and output to SIF Output Register | 150µs |
| C9 | | GEN_EEP_SIGN | GET_EEP_SIGN + write the Signature to EEPROM address 0x1D | 12.6ms / 1 |
| CA | | GET_RAM_SIGN | Calculates RAM Signature and output to SIF Output Register | 150µs |
| CB | | GEN_RAM_SIGN | GET_RAM_SIGN + write the Signature to RAM address 0x1D | 150µs |
| CC | | CLEAR_EEP | Clear EEPROM<br>Sets complete EEPROM to 0x0000 | 12.5ms / 1 |
| CF | | ROM_VERSION | Read ROM version and write it to SIF Output Register<br>Usually followed by Read operation | 50µs |
| All "D*"-Commands: used for calibration process, write raw conversion result to SIF Output Registers, do not effect to analog output | | | | |
| D0 | | START_AD_P | Start cyclic A/D-conversion at channel Humidity P | 50µs<br>+<br>AD-conversion time |
| D1 | | START_AD_T1 | Start cyclic A/D-conversion at channel Temperature T1 | |
| D2 | | START_AD_T2 | Start cyclic A/D-conversion at channel Temperature T2 | |
| D4 | | START_AD_PAZ | Start cyclic A/D-conversion at channel Auto-Zero Humidity PAZ | |
| D5 | | START_AD_TAZ1 | Start cyclic A/D-conversion at channel Auto-Zero Temperature TAZ1 | |
| D6 | | START_AD_TAZ2 | Start cyclic A/D-conversion at channel Auto-Zero Temperature TAZ2 | |
| D8 | | START_AD_P_AZ | Start cyclic A/D-conversion at channel Humidity P incl. Auto-Zero-Correction | |
| D9 | | START_AD_T1_AZ | Start cyclic A/D-conversion at channel Temperature T1 incl. Auto-Zero-Correction | |
| DA | | START_AD_T2_AZ | Start cyclic A/D-conversion at channel Temperature T2 incl. Auto-Zero-Correction | |
| DB | | START_AD_CMV_AZ | Start cyclic A/D-conversion at channel Common Mode Voltage CMV<br>incl. Auto-Zero-Correction | |

***Table 3: Additional command set for serial digital interface***

Note: These commands are only available in CM.

CM is set by command START_CM (0x72$_{hex}$) as first command after power on.

The technical information in this document has been checked with adequate care at our end and is intended to inform about the product and its applications. The descriptions are not to be understood as assurance of the defined characteristics of the product and should be checked by the user for the intended application. Any possible industrial third party patent rights are to be considered.

Issued August 2008 - This documentation supersedes all previous editions.