Application Note TSYS01

PRODUCT HIGHLIGHTS

- **High Accuracy up to ±0.1°C**
- **Very Small Size**
- **Ready for SMT Assembly**
- **Multiple Interfaces I2C, SPI**
- **Adjustment of High Accuracy Temperature Range on Request**
- **Low Current Consumption**
- **Low Self Heating**
- **Additional Input for External Temperature Sensor Component**

DESCRIPTION

The TSYS01 is a single chip, versatile, new technology temperature sensor. The TSYS01 provides factory calibrated temperature information. It includes a temperature sensing chip and a 24 bit ΔΣ-ADC. The essence of the digital 24 bit temperature value and the internal factory set calibration values lead to highly accurate temperature information accompanied by high measurement resolution.

The TSYS01 can be interfaced to any microcontroller by an I^2C or SPI interface. This microcontroller has to calculate the temperature result based on the ADC values and the calibration parameters.

The basic working principle is:

- Converting temperature into digital 16/24 bit ADC value
- Providing calibration coefficients
- Providing ADC value and calibration coefficients by SPI or I^2C interface.

SPECIFICATION OVERVIEW

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Supply Voltage	V_{DD}	stabilized	2.2		3.6	V
High Accuracy Supply Voltage	V_{DD}	To achieve Acc1	3.2		3.4	V
Supply Current	I_{DD}	1 sample per second			12.5	µA
Standby current	IS	No conversion, VDD = 3V T = 25°C T = 85°C		0.02 0.70	0.14 1.40	µA µA
Peak Supply Current	I_{DD}	During conversion		1.4		mA
Conversion time	T_{CONV}		7.40	8.22	9.04	ms
Serial Data Clock SPI	F_{SCLK}				20	MHz
Serial Data Clock I^2C	F_{SCL}				400	kHz
VDD Capacitor		Place close to the chip	100nF			

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Supply Voltage	V_{DD}	stabilized	2.2		3.6	V
High Accuracy Supply Voltage	V_{DD}	To achieve Acc1	3.2		3.4	V
Supply Current	I_{DD}	1 sample per second			12.5	µA
Standby current	IS	No conversion, VDD = 3V T = 25°C T = 85°C		0.02 0.70	0.14 1.40	µA µA
Peak Supply Current	I_{DD}	During conversion		1.4		mA
Conversion time	T_{CONV}		7.40	8.22	9.04	ms
Serial Data Clock SPI	F_{SCLK}				20	MHz
Serial Data Clock I^2C	F_{SCL}				400	kHz
VDD Capacitor		Place close to the chip	100nF			

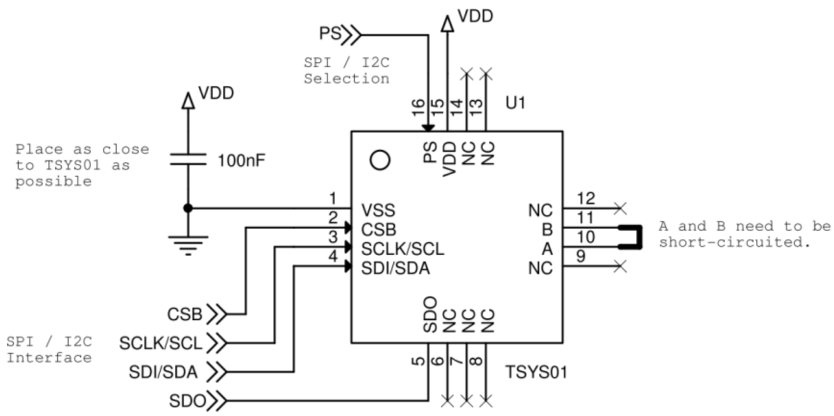
DIGITAL INPUTS (SCLK, SDI, CSB, PS)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High Voltage	V_{IH}	VDD = 2.2…3.6V	0.7 V_{DD}		V_{DD}	V
Input Low Voltage	V_{IL}	VDD = 2.2…3.6V	0.0 V_{DD}		0.3 V_{DD}	V
CS low to first SCLK rising	t_{CSL}		21			ns
CS high to first SCLK rising	t_{CSH}		21			ns
SDI setup to first SCLK rising	T_{DSO}		6			ns
SDI hold to first SCLK rising	T_{DO}		6			ns

DIGITAL OUTPUTS (SDA, SDO)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Output High Voltage	V_{OH}	I_{Source} = 1mA	0.8 V_{DD}		V_{DD}	V
Output Low Voltage	V_{OL}	I_{Sink} = 1mA	0.0 V_{DD}		0.2 V_{DD}	V
SDO setup to first SCLK rising	t_{QS}		10			ns
SDO hold to first SCLK rising	t_{QH}		0			ns

CONNECTION DIAGRAM



PIN FUNCTION TABLE

Pin	Name	Type	Function
1	VSS	G	Ground
2	CSB	DI	SPI: Chip Select (active low) I^2C: Address Selection
3	SCLK/SCL	DI	SPI: Serial Data Clock I_2C: Serial Data Clock
4	SDI/SDA	DIO	SPI: Serial Data Input I_2C: Data Input / Output
5	SDO	DO	SPI: Serial Data Output
6 – 9	NC	---	Not connected / Do not connect
10	A	I	Connect Pin10 with Pin11
11	B	I	Connect Pin11 with Pin10
12 – 14	NC	---	Not connected / Do not connect
15	VDD	P	Supply Voltage
16	PS	DI	Communication protocol select (0=SPI, 1=I^2C)

SOLDER RECOMMENDATION

Solder reflow process according to IPC/JEDEC J-STD-020D (Pb-Free Process) is recommended.

MEASUREMENT GUIDELINES

GENERAL

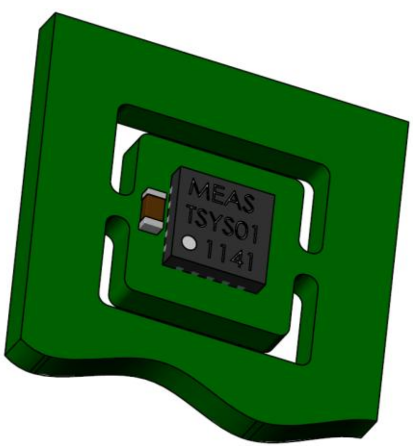
In order to achieve the most accurate temperature measurement results, please notice these advices

- Use a stabilized and noise free supply voltage
- Place a ceramic capacitor close to the supply pins
- Keep supply lines as short as possible
- Separate TSYS01 from any heat source which is not meant to be measured.
- Avoid air streams if the PCB temperature is meant to be measured.

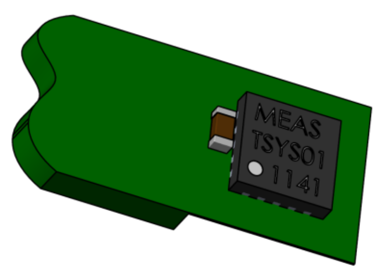
MEASUREMENT OF AIR TEMPERATURE

- Separate TSYS01 from the remaining electronics by PCB layout.

Milled thermal relief

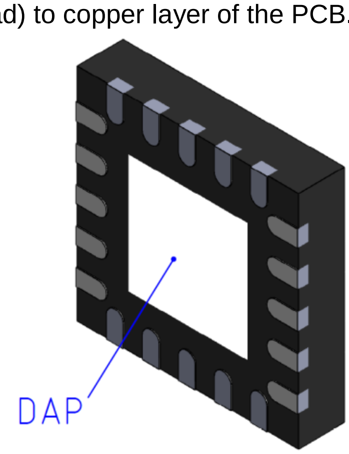


Flex PCB



MEASUREMENT PCB TEMPERATURE

- Connect DAP (die attach pad) to copper layer of the PCB.



INTERFACE DESCRIPTION

PROTOCOL SELECTION

PS pin input level has to be defined in dependence to protocol selection.

- PS = 0 activates SPI.
- PS = 1 activates I^2C.

I^2C INTERFACE

A I^2C communication message starts with a start condition and it is ended by a stop condition.
Each command consists of two bytes: the address byte and command byte.

I^2C ADDRESS SELECTION

The I^2C address can be selected by CSB pin.

- CSB=1 then the address is 1110110x.
- CSB=0 the address is 1110111x.

Therefore, two TSYS01 can be interfaced on the same I^2C bus.

SPI INTERFACE

The serial interface is a 4-wire SPI bus, operating as a slave. CS (chip select), SCLK (serial clock), SDI (serial data in), and SDO (serial data out) are used to interact with the SPI master.
Communication with the chip starts when CS is pulled to low and ends when CS is pulled to high.
SCLK is controlled by the SPI master and idles low (SCLK low on CS transitions, mode 0).
A mode where the clock alternatively idles high is also supported (mode 3).

COMMANDS

The commands are the same for SPI and I^2C interface.
There are four commands:

- Reset
- Read PROM (calibration parameters)
- Start ADC Temperature conversion
- Read ADC Temperature result

Command	Hex Value
Reset	0x1E
Start ADC Temperature Conversion	0x48
Read ADC Temperature Result	0x00
PROM Read Address 0	0xA0
PROM Read Address 1	0xA2
PROM Read Address 2	0xA4
PROM Read Address 3	0xA6
PROM Read Address 4	0xA8
PROM Read Address 5	0xAA
PROM Read Address 6	0xAC
PROM Read Address 7	0xAE

INTERFACE CODE EXAMPLES

SPI INTERFACE

The code examples shown are meant to be understood as exemplary. The code has to be adjusted with respect to the used microcontroller in order to work correctly.

```
/******************************************************************
*      Function:    TSYS01_SPI_INIT                               *
*      Input:       ---                                           *
*      Return:      ---                                           *
*      Description: Initialization of SPI Port                    *
*******************************************************************/
void TSYS01_SPI_INIT(void)
{
       // Configure IOs
       SDI_DIR = IN;                              // SDI = Input
       SDO_DIR = OUT;                             // SDO = Output
       SCL_DIR = OUT;                             // SCL = Output
       CSB_DIR = OUT;                             // CSB = Output
}

/******************************************************************
*      Function:    TSYS01_SPI_TRANSFER                           *
*      Input:       char cTransmit      Byte to be send to TSYS01 *
*      Return:      char cReceive       Byte received from TSYS01 *
*      Description: Sends one byte to TSYS01 and read on byte     *
*                   from TSYS01 simultaneously                    *
*******************************************************************/
char TSYS01_SPI_TRANSFER(char cTransmit)
{
       char cReceive = 0;
       char cBit = 0;

       SDO = 0;      SCL = 0;                     // Reset SPI Lines

       for (cBit = 0; cBit < 8; cBit++)
       {
              cReceive = cReceive << 1;           // Shift Receive Register
              SCL = 0;                            // SCL = 0
              SDO = (cTransmit >> (7 - cBit));    // Outupt next Bit on SDO
              SCL = 1;                            // SCL = 1
              cReceive = cReceive | SDI;          // Input next Bit on SDI
       }

       RC3 = 0;      RC5 = 0;                     // Reset SPI Lines

       return cReceive;
}
```

```
/*******************************************************************
*      Function:    TSYS01_SPI_READ_ADC                           *
*      Input:       ---                                           *
*      Return:      cADC[4] via call by reference                 *
*      Description: Reads four bytes of ADC result (24bit)        *
*******************************************************************/
void TSYS01_SPI_READ_ADC(char *cADC)
{
       char cByte;

       CSB = 1;
       CSB = 0;                                    // Enable Chip Select

       cADC(0) = TSYS01_TRANSFER(0x48);            // Start Conversion
       while (SDI == 0);                           // Wait for Conversion done

       CSB = 1;
       CSB = 0;                                    // Enable Chip Select
       for (cByte = 0; cByte < 4; cByte++)
       {
              cADC[cByte] = TSYS01_TRANSFER(0x00);     // READ ADC
       }
       CSB = 1;
}

/*******************************************************************
*      Function:    TSYS01_SPI_READ_PROM_WORD                     *
*      Input:       char cAddress Address of Prom to be read      *
**     Return:      cPPROM[2] via call by reference               *
*      Description: Reads two byte (on word) of Prom memory       *
*******************************************************************/
void TSYS01_SPI_READ_PROM_WORD(char cAddress, char *cPROM)
{
       cAdress = 0xA0 | (cAddress << 1);

       CSB = 1;
       CSB = 0;                                    // Enable Chip Select
       cPPROM[0] = TSYS01_TRANSFER (cAdress);      // Command Read PROM

       cPPROM[0] = TSYS01_TRANSFER(0x00);          // Read high byte
       cPPROM[1] = TSYS01_TRANSFER(0x00);          // Read low byte
       CSB = 1;
}
```

I²C INTERFACE

The code examples shown are meant to be understood as exemplary. The code has to be adjusted with respect to the used microcontroller in order to work.

```
/****************************************************************
*      Function:    TSYS01_I2C_INIT                             *
*      Input:       ---                                         *
*      Return:      ---                                         *
*      Description: Initialization of I2C Port                  *
****************************************************************/
void TSYS01_I2C_INIT(void)
{
      I2C_SCK_DIR = OUT;                          // SCK = Output
      I2C_SDA_DIR = OUT;                          // SDA = Output
}

/****************************************************************
*      Function:    TSYS01_I2C_READ_PROM_WORD                   *
*      Input:       char cAddress Address of Prom to be read    *
**     Return:      cPPROM[2] via call by reference             *
*      Description: Reads two byte (on word) of Prom memory     *
****************************************************************/
void TSYS01_I2C_READ_PROM_WORD(char cAddress, char *cPROM)
{
      cAdress = 0xA0 | (cAddress << 1);

      TSYS01_I2C_START();                         // Send Start Condition
      TSYS01_I2C_TRANSMIT_BYTE(I2C_ADRESS | I2C_W); // Send I2C-Address, Write
                                                  // Mode
      TSYS01_I2C_GET_ACK();                       // Get ACK
      TSYS01_I2C_SEND_BYTE(cAdress);              // Send Read PROM command
                                                  // including address to read
      TSYS01_I2C_GET_ACK();                       // Get ACK
      TSYS01_I2C_STOP();                          // Send Stop Condition

      TSYS01_I2C_START();                         // Send Start Condition
      TSYS01_I2C_TRANSMIT_BYTE(I2C_ADRESS | I2C_R); // Send I2C-Address, Read Mode
      TSYS01_I2C_GET_ACK();                       // Get ACK
      cPPROM[0] = TSYS01_I2C_RECEIVE_BYTE(void)   // Read high byte
      I2C_SET_ACK(TRUE);                          // Set ACK
      cPPROM[1] = TSYS01_I2C_RECEIVE_BYTE(void)   // Read low byte
      I2C_SET_ACK(FALSE);                         // Set NACK
      TSYS01_I2C_STOP();                          // Send Stop Condition
}
```

```
/*****************************************************************
*      Function:    TSYS01_I2C_READ_ADC                          *
*      Input:       ---                                          *
*      Return:      cADC[4] via call by reference                *
*      Description: Reads four bytes of ADC result (24bit)       *
*****************************************************************/
void TSYS01_I2C_READ_ADC(char *cADC)
{
       char cByte;

       // Send command to start ADC conversion
       TSYS01_I2C_START();                              // Send Start Condition
       TSYS01_I2C_TRANSMIT_BYTE(I2C_ADRESS | I2C_W);   // Send I2C-Address, Write
                                                        // Mode
       TSYS01_I2C_GET_ACK();                            // Get ACK
       TSYS01_I2C_SEND_BYTE(0x48);                      // Start Conversion
       TSYS01_I2C_GET_ACK();                            // Get ACK
       TSYS01_I2C_STOP();                               // Send Stop Condition

       // Repeat this block until Acknowledge is true
       // or wait 10ms for conversion to be done
       TSYS01_I2C_START();                              // Send Start Condition
       TSYS01_I2C_TRANSMIT_BYTE(I2C_ADRESS | I2C_W);   // Send I2C-Address, Write Mode
       TSYS01_I2C_GET_ACK();                            // Get ACK
       TSYS01_I2C_STOP();                               // Send Stop Condition

       TSYS01_I2C_START();                              // Send Start Condition
       TSYS01_I2C_TRANSMIT_BYTE(I2C_ADRESS | I2C_W);   // Send I2C-Address, Write Mode
       TSYS01_I2C_GET_ACK();                            // Get ACK
       TSYS01_I2C_SEND_BYTE(0x00);                      // Send Read ADC command
       TSYS01_I2C_GET_ACK();                            // Get ACK
       TSYS01_I2C_STOP();                               // Send Stop Condition

       TSYS01_I2C_START();                              // Send Start Condition
       TSYS01_I2C_TRANSMIT_BYTE(I2C_ADRESS | I2C_R);   // Send I2C-Address, Read Mode
       TSYS01_I2C_GET_ACK();                            // Get ACK
       cADC[0] = TSYS01_I2C_RECEIVE_BYTE(void)          // Read first byte
       I2C_SET_ACK(TRUE);                               // Set ACK
       cADC[1] = TSYS01_I2C_RECEIVE_BYTE(void)          // Read next byte
       I2C_SET_ACK(TRUE);                               // Set ACK
       cADC[2] = TSYS01_I2C_RECEIVE_BYTE(void)          // Read next byte
       I2C_SET_ACK(TRUE);                               // Set ACK
       cADC[3] = TSYS01_I2C_RECEIVE_BYTE(void)          // Read last byte
       I2C_SET_ACK(FALSE);                              // Set NACK
       TSYS01_I2C_STOP();                               // Send Stop Condition
}
```

```
/*****************************************************************
*      Function:     TSYS01_I2C_START                            *
*      Input:        ---                                         *
*      Return:       ---                                         *
*      Description:  Send I2C Start Condition                    *
*****************************************************************/
void TSYS01_I2C_START(void)
{
       I2C_SCK_DIR = OUT;                            // SCK = Output
       I2C_SDA_DIR = OUT;                            // SDA = Output

       I2C_SCK = 1;
       I2C_SDA = 1;
       I2C_SDA = 0;
}

/*****************************************************************
*      Function:     TSYS01_I2C_STOP                             *
*      Input:        ---                                         *
*      Return:       ---                                         *
*      Description:  Send I2C Stop Condition                     *
*****************************************************************/
void TSYS01_I2C_STOP(void)
{
       I2C_SCK_DIR = OUT;                            // SCK is Output
       I2C_SDA_DIR = OUT;                            // SDA is Output

       I2C_SCK = 1;
       I2C_SDA = 0;
       I2C_SDA = 1;
}
```

```
/*****************************************************************
*      Function:    TSYS01_I2C_TRANSMIT_BYTE                     *
*      Input:       char cTransmit      Byte to be send to TSYS01 *
*      Return:      ---                                           *
*      Description: Sends one byte to TSYS01                      *
******************************************************************/
void TSYS01_I2C_TRANSMIT_BYTE(char cTransmit)
{
       char cBit, cMask;
       cMask = 0x80;

       I2C_SCK_DIR = OUT;                          // SCK is Output
       I2C_SDA_DIR = OUT;                          // SDA is Output

       I2C_SCK = 0;
       for (cBit = 0; cBit < 8; cBit ++)
       {
              I2C_SDA = 0;
              if ((cTransmit & cMask) != 0)    I2C_SDA = 1;
              I2C_SCK = 1;
              I2C_SCK = 0;
              cMask = cMask >> 1;
       }
}

/*****************************************************************
       Function:    TSYS01_I2C_RECEIVE_BYTE
       Input:       ---
       Return:      char cReceive Byte received from TSYS01
       Description: Reads one byte from TSYS01
******************************************************************/
char TSYS01_I2C_RECEIVE_BYTE(void)
{
       char cReceive, cBit;

       I2C_SCK_DIR = IN;                           // SCK is Input
       I2C_SDA_DIR = IN;                           // SDA is Input

       while (I2C_SCK == 0);                       // Wait for SCL release

       I2C_SCK_DIR = OUT;                          // SCK is Output

       I2C_SCK = 0;
       I2C_SCK = 1;
       for (cBit = 0; cBit < 8; cBit++)
       {
              cReceive = cReceive << 1;
              I2C_SCK = 1;
              if (I2C_SDA == 1)   cReceive = cReceive + 1;
              I2C_SCK = 0;
       }
       return cReceive;
}
```

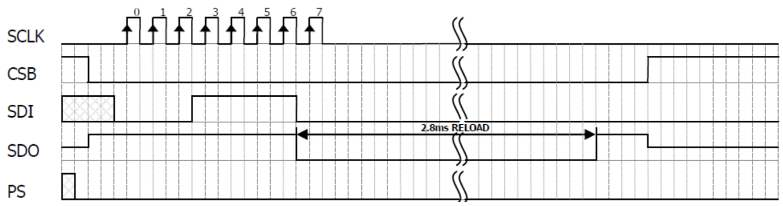
```
/*****************************************************************
*      Function:    TSYS01_I2C_GET_ACK                          *
*      Input:       ---                                         *
*      Return:      bit bACK     Bit represents ACK status      *
*      Description: Reads Acknowledge from TSYS01               *
*****************************************************************/
bit TSYS01_I2C_GET_ACK(void)
{
      bit bACK;

      I2C_SCK_DIR = OUT;                          // SCK is Output
      I2C_SDA_DIR = IN;                           // SDA is Input

      I2C_SCK = 0;
      I2C_SCK = 1;
      bACK = I2C_SDA;
      I2C_SCK = 0;
      return bACK;
}

/*****************************************************************
      Function:    TSYS01_I2C_Set_ACK
      Input:       bit bACK     Bit represents ACK status to be send
      Return:      ---
      Description: Reads Acknowledge from TSYS01
*****************************************************************/
void I2C_SET_ACK(bit bACK)
{
      I2C_SCK_DIR = OUT;                          // SCK is Output
      I2C_SDA_DIR = OUT;                          // SDA is Output

      I2C_SCK = 0;
      I2C_SDA = bACK;
      I2C_SCK = 1;
      I2C_SCK = 0;
}
```

INTERFACE TRANSMISSIONS

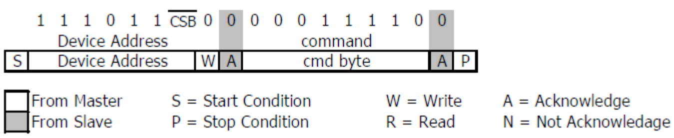
RESET SEQUENCE

The Reset sequence has to be sent once after power-on. It can be also used to reset the device ROM from an unknown condition.

SPI



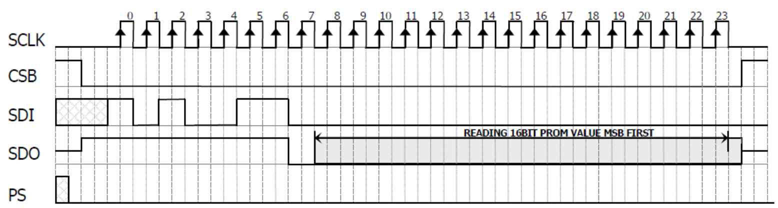
I²C



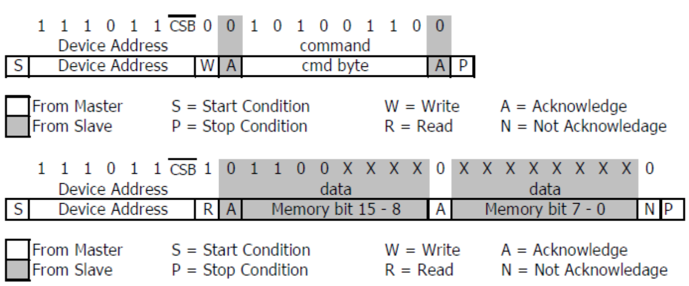
PROM READ SEQUENCE

The PROM Read command consists of two parts. First command sets up the system into PROM read mode. The second part gets the data from the system.
Below examples are sequences to read address 3 (command 0xA6).

SPI



I²C

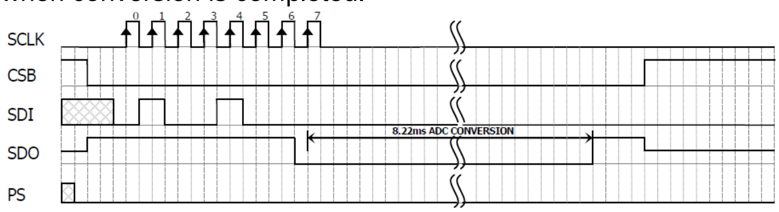


CONVERSION SEQUENCE

A conversion has to be started by sending this command. The sensor stays busy until conversion is done. When conversion is finished the data can be accessed by using ADC read command

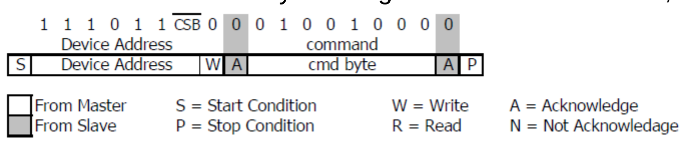
SPI

The last clock will start the conversion which TSYS01 indicates by pulling SDO low. SDO goes high when conversion is completed.



I²C

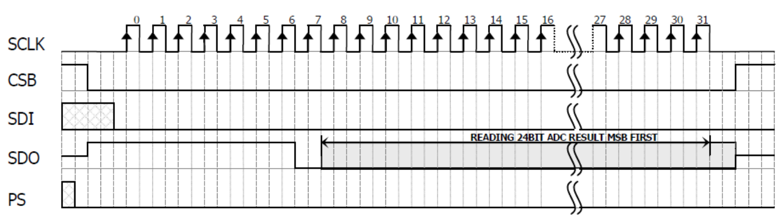
When the command is sent the TSYS01 stays busy until the conversion is done. All other commands except the reset command will not be executed during this time. When the conversion is finished the data can be accessed by sending a ADC read command, when an acknowledge appears from TSYS01.



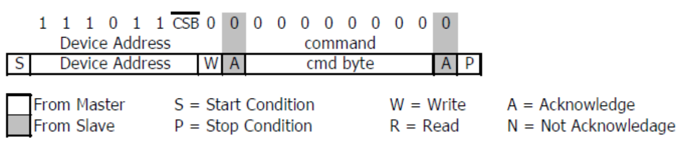
READ ADC RESULT

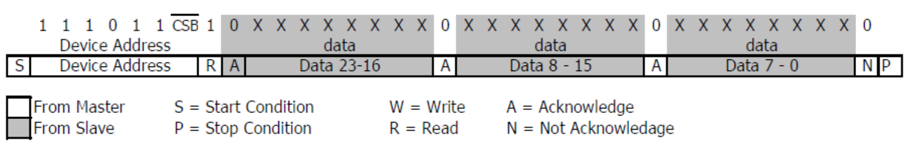
After the conversion command the ADC result is read using ADC read command. Repeated ADC read commands, or command executed without prior conversion will return all 0 as result.

SPI

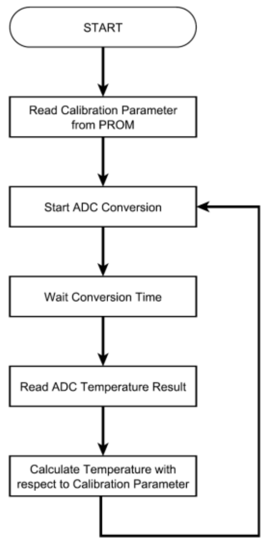


I²C





TEMPERATURE CALCULATION



CALIBRATION PARAMETER

Variable	Description	Command	Size / bit	Min	Max	Example
k_4	Coefficient k_4 of polynomial	0xA2	16	0	65535	28446
k_3	Coefficient k_3 of polynomial	0xA4	16	0	65535	24926
k_2	Coefficient k_2 of polynomial	0xA6	16	0	65535	36016
k_1	Coefficient k_1 of polynomial	0xA8	16	0	65535	32791
k_0	Coefficient k_0 of polynomial	0xAA	16	0	65535	40781

TEMPERATURE POLYNOMAL

ADC24: ADC value
ADC16: ADC24 / 256

$$
\begin{aligned}
T / {}^\circ C = \; & (-2) * k_4 * 10^{-21} * ADC16^4 + \\
& 4 * k_3 * 10^{-16} * ADC16^3 + \\
& (-2) * k_2 * 10^{-11} * ADC16^2 + \\
& 1 * k_1 * 10^{-6} * ADC16 + \\
& (-1.5) * k_0 * 10^{-2}
\end{aligned}
$$

EXAMPLE

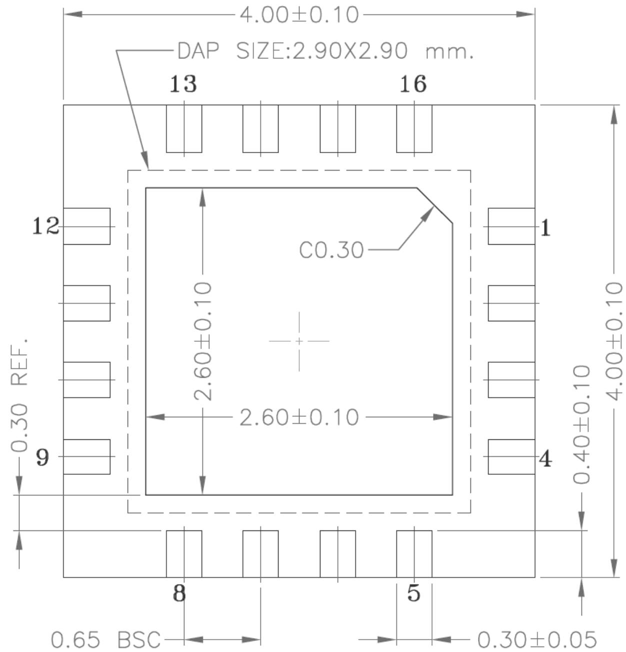
ADC24: 9378708
ADC16: 9378708 / 256 = $\underline{36636}$

$$
\begin{aligned}
T / {}^\circ C = \; & (-2) * 28446 * 10^{-21} * 36636^4 + \\
& 4 * 24926 * 10^{-16} * 36636^3 + \\
& (-2) * 36016 * 10^{-11} * 36636^2 + \\
& 1 * 32791 * 10^{-6} * 36636 + \\
& (-1.5) * 40781 * 10^{-2}
\end{aligned}
$$

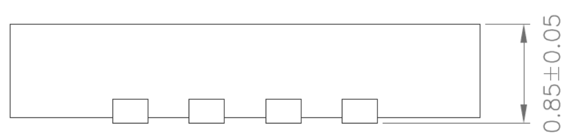
$$T / {}^\circ C = \underline{10.59}$$

DIMENSIONS

BOTTOM VIEW

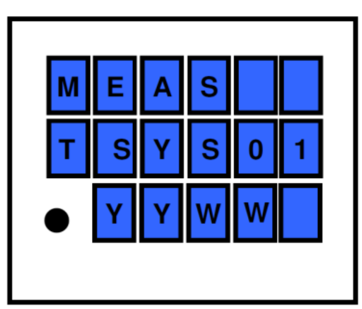


SIDE VIEW



MARKING

Line	Description	Example
1	Manufacturer	MEAS
2	Product Name	TSYS01
3	Pin 1 Dot, Date Code YYWW	1141



Application Note TSYS01

ORDER INFORMATION

Please order this product using following:

Part Number	Part Description
G-NICO-018	TSYS01 Digital Temperature Sensor

EMC

Due to the use of these modules for OEM application no CE declaration is done.

Especially line coupled disturbances like surge, burst, HF etc. cannot be removed by the module due to the small board area and low price feature. There is no protection circuit against reverse polarity or over voltage implemented.

The module will be designed using capacitors for blocking and ground plane areas in order to prevent wireless coupled disturbances as good as possible.

DEFINITIONS AND DISCLAIMERS

- Application information – Applications that are described herein for any of these products are for illustrative purpose only. MEAS Deutschland GmbH makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.
- Life support applications – These products are not designed for use in life support appliances, devices, or systems where malfunctions of these products can reasonably be expected to result in personal injury. MEAS Deutschland GmbH customers using or selling this product for use in such applications do so at their own risk and agree to fully indemnify MEAS Deutschland GmbH for any damages resulting from such improper use or sale.

TECHNICAL CONTACT INFORMATION

NORTH AMERICA

Measurement Specialties, Inc.
910 Turnpike Road
Shrewsbury, MA 01545
Tel: 1-508-842-0516
Fax: 1-508-842-0342

Sales email:
temperature.sales.amer@meas-spec.com

EUROPE

MEAS Deutschland GmbH
Hauert 13
44277 Dortmund
Tel: +49 (0) 231/9740-0
Fax: +49 (0) 231/9740-20

Sales email:
info.de@meas-spec.com

ASIA

Measurement Specialties (China) Ltd.
No. 26 Langshan Road
Shenzhen High-Tech Park (North)
Nanshan District, Shenzhen 518057, China
Tel: +86 (0) 755 33305088
Fax: +86 (0) 755 33305099

Sales email:
temperature.sales.asia@meas-spec.com