Documentation

EVA3011U User Guide

Table of Content

1 Revision History

Version	Date	Changes	Page
Initial Version V1.0	/2010		

2 Overview

The EVA3011U is especially designed to evaluate the PE3011 HF RFID Integrated Circuit of Productivity Engineering GmbH featuring an integrated temperature sensor, external capacitive sensors and real time clock as well as an 8kBit EEPROM for data monitoring. Digital external sensors can also be incorporated with the help of a microcontroller through the SPI interface of the chip.
The EVA3011 serves as a demonstrator and evaluation kit with these features:

- PCB with IC, LED, resistors, sensors and battery
- passive RFID HF transponder with ISO 15693 protocol interface
- recording of time, sensor and temperature events in a defined time interval, either in normal mode or in outband mode
- recording of time and sensor events as interrupt function
- readout of all stored data via HF-RFID interface (ISO15693)
- detailed representation of data in MS Excel format for a graphical view in Windows OS
- default configuration of Evaluation Kit (BAT; C3=22p; C1=100n; PE3011)
- external sensors optional with single ended or differential hardware
- R4 and LED1 (indicator alarm port - optional)

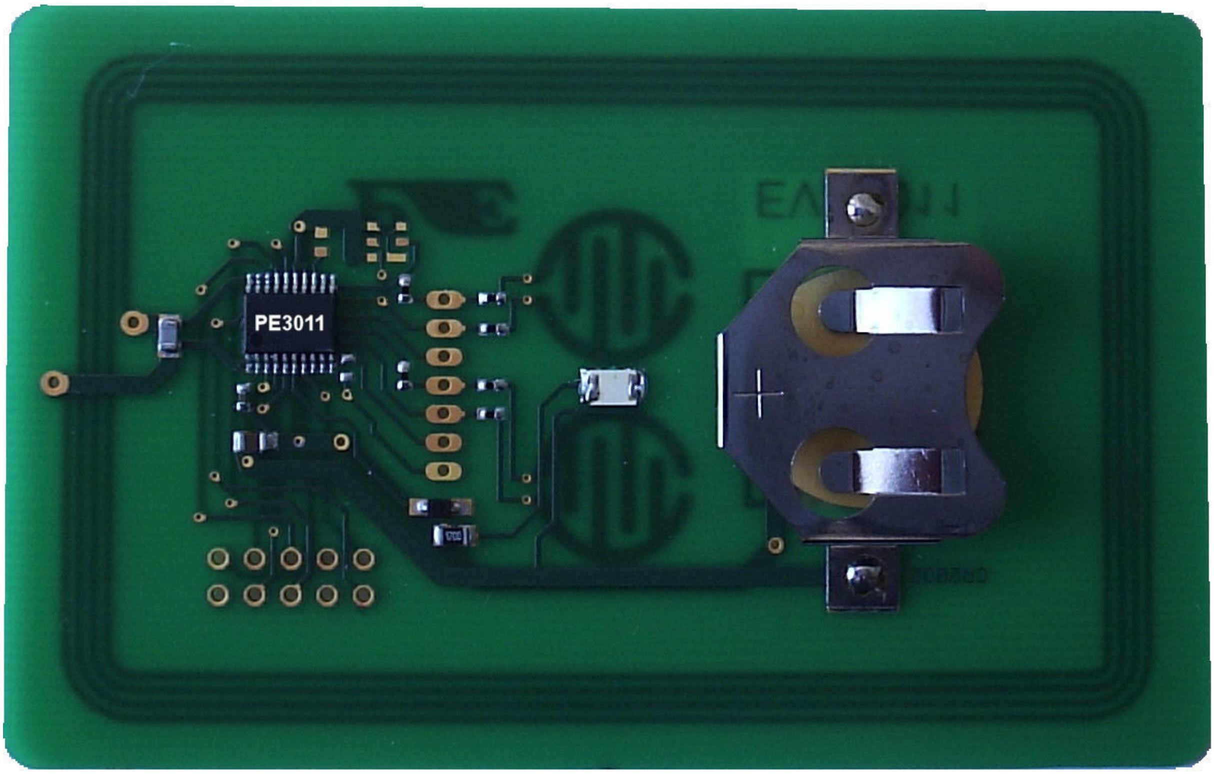


Figure1 - Evaluation Kit PE3011 v1.0

3 Controlling software „EVA3011U“

3.1 Installation

After downloading the software to manipulate the Data Monitor with a SCEMTEC Reader it needs to be installed by executing the file **“Install_EVA3011_1.0.exe”**.

The EVA3011 comes with a graphical user interface for WindowsXP ™ platforms. It is recommended to start with this simple software interface to learn about the basic functionality of the chip on the board.

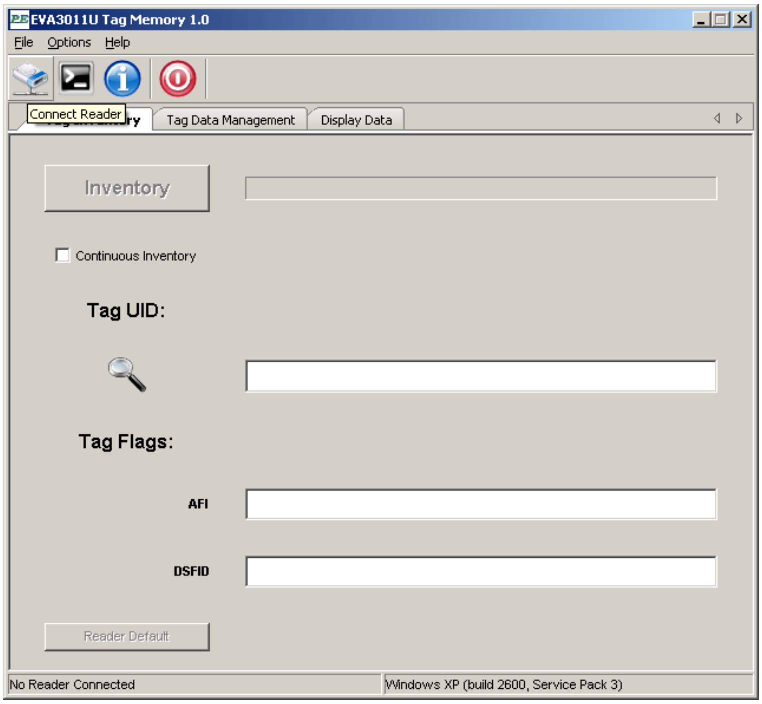
All necessary files will be copied into the target directory and a group will be created in the “Start” Menu. To communicate with SCEMTEC Reader the internal USB to COM driver with DLL will be used. Readers from other vendors may now work properly. Typically each reader requires its own USB driver interface. Driver installation can be done through standard software installation or through the “Start” Menu.

SCEMTEC

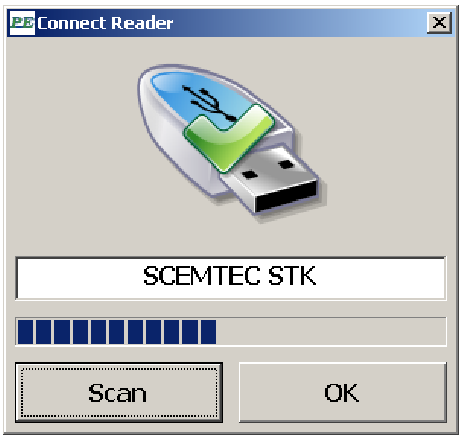
The driver interface for USB must be installed with the reader software from reader vendor. For the operation with SCEMTEC a DLL and special internal C++ functions are designed. With these functions a SCEMTEC reader SIR 2710 can communicate through the USB to COM conversion. The software automatically checks the devices connected to the ports and connects with the reader.

3.2 Software description – Connect Reader

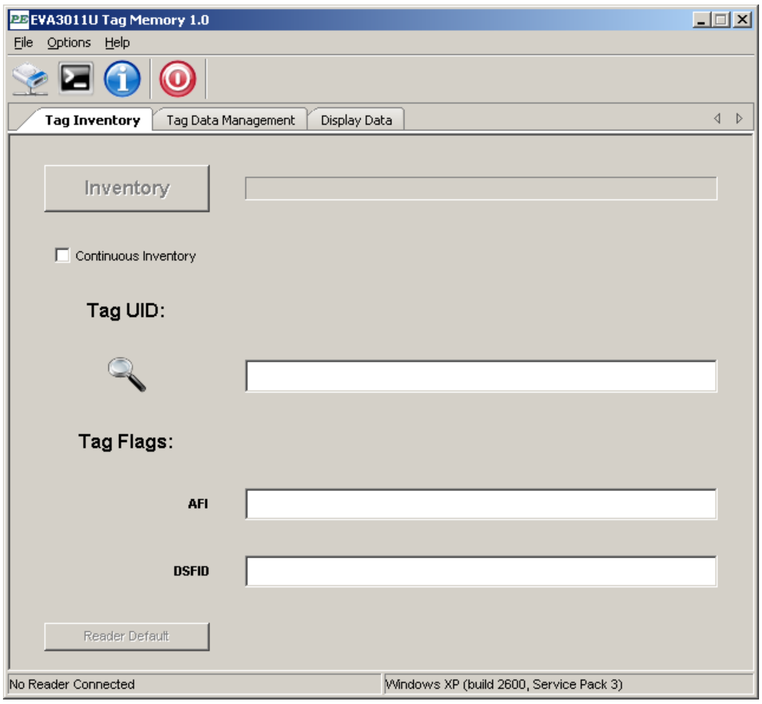
After starting the software (over “Start” Menu) and the access of an HF-Reader the connection can be established. For this action the menu **"File"** and the point **"Connect Reader"** or the icon should be used.



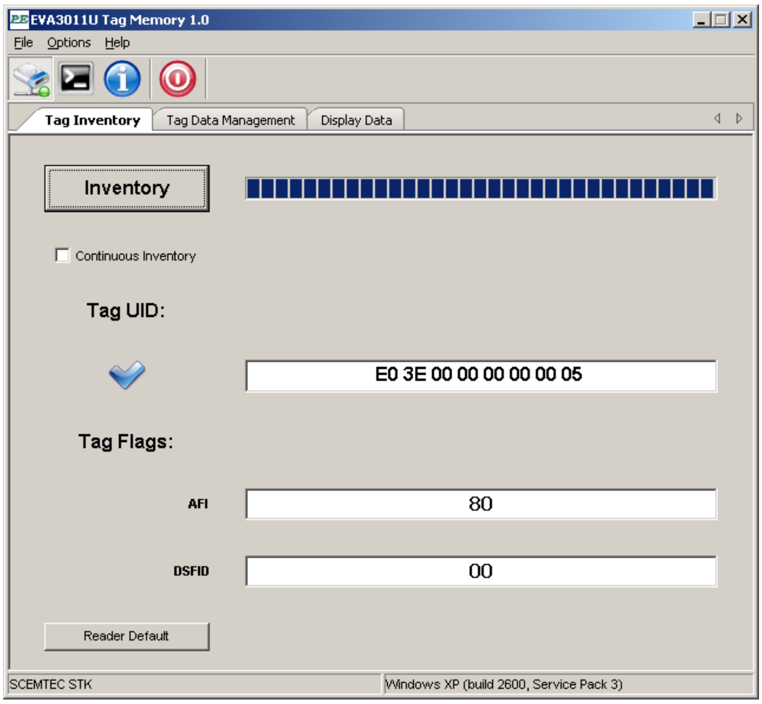
After pressing Scan it will now scan the ports for a connected reader. If it finds a valid reader the connection can be established by confirming with **"OK"**.



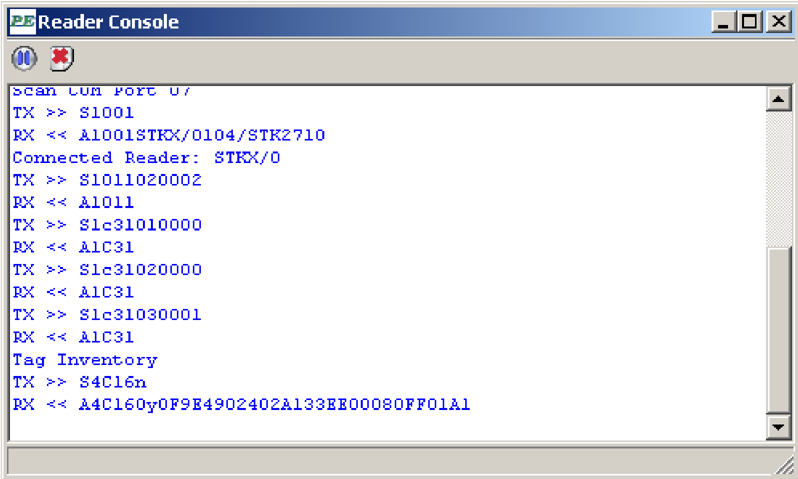
If no reader is found the Inventory button is not usable and the software is working without any tag information or activities.



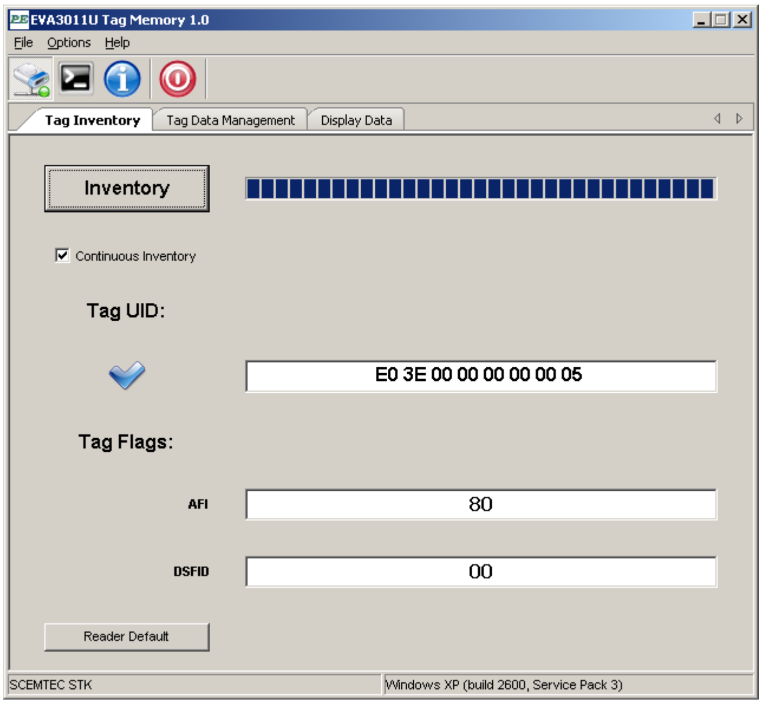
After a successful reader connection the software can **"Inventory"** a tag for UID, AFI and DSFID data.



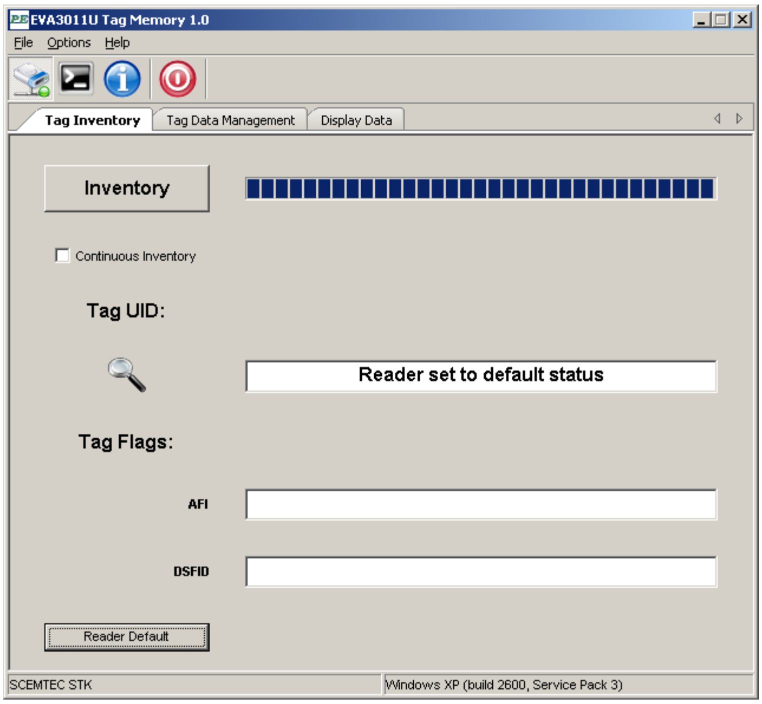
It is possible to use the Reader Console (menu **"OPTIONS"** -> **"Reader Console"** or the icon) to see the internal data transfer between software reader and tag.



The checkbox **"Continuous Inventory"** select a permanently read with the button **“Inventory”**. In this way it is possible to find an optimal position between tag and reader. A stop of thes function is possible with deselecting checkbox **"Continuous Inventory"**.



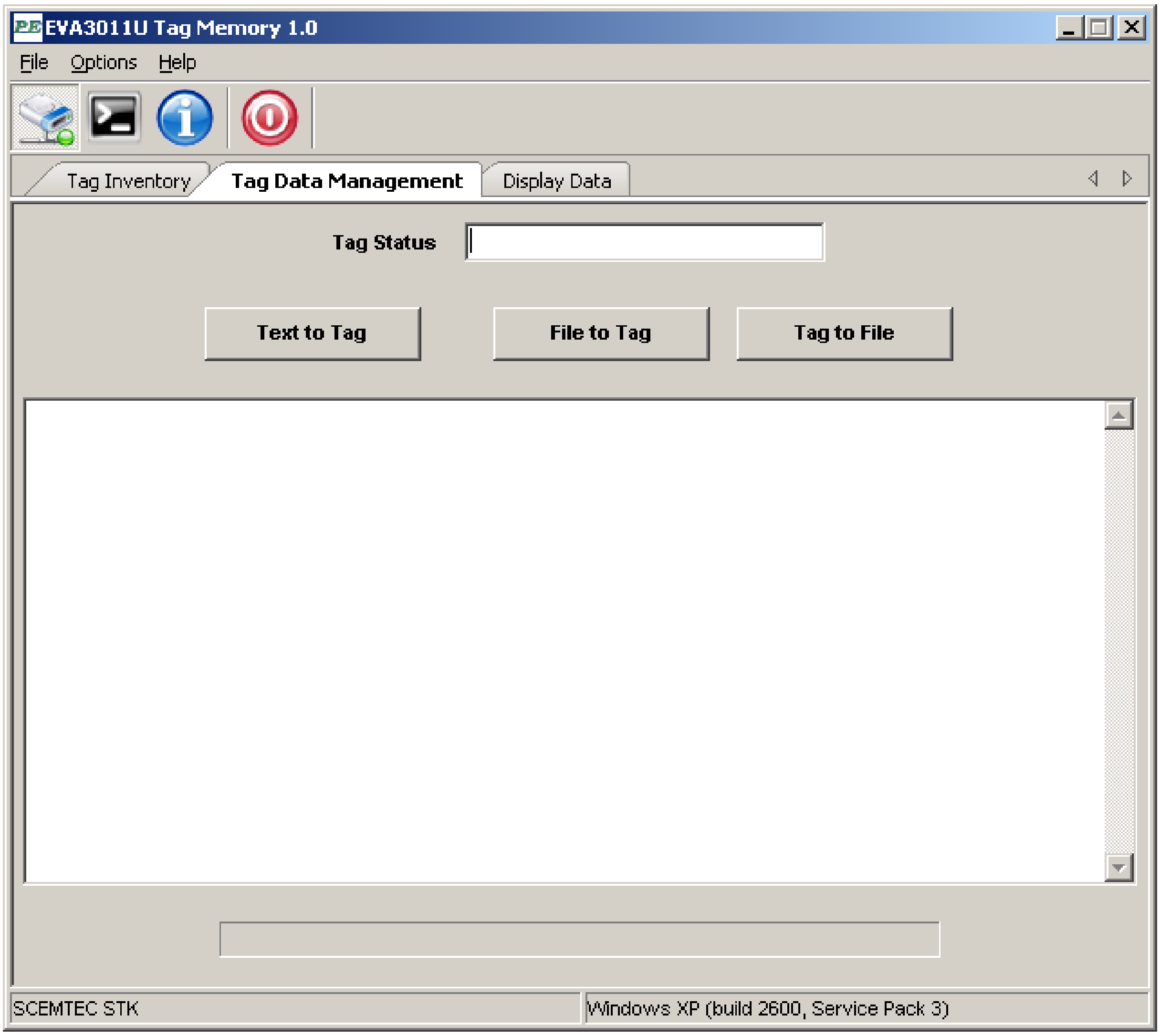
The button **"Reader Default"** sets all reader specific flags to default configuration. This is necessary if the reader is not set to ISO15693 conformity.



This way all reader and tag pre-definitions are completed.

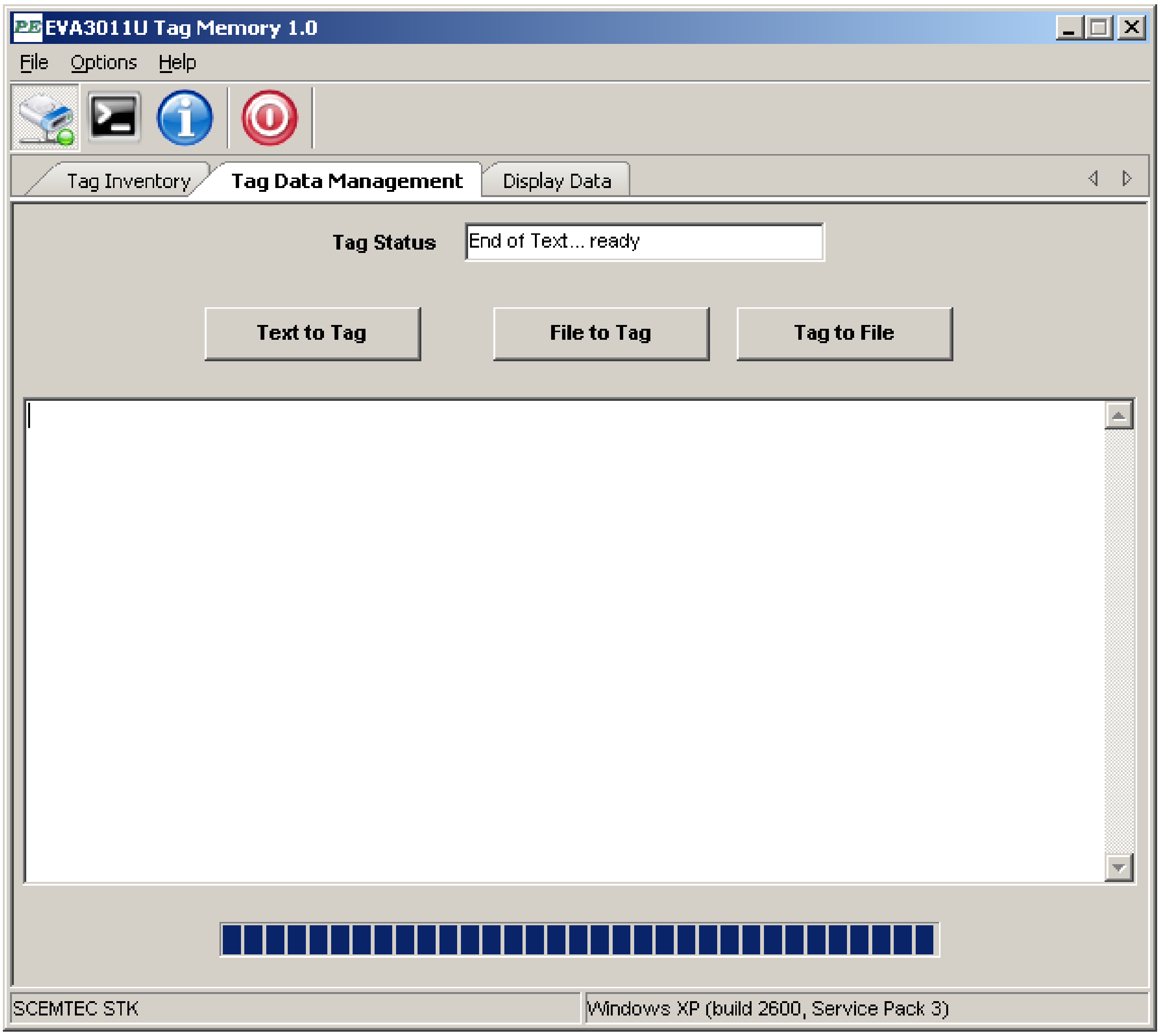
3.3 Software description – Tag Data Management

To read and manipulate the Data memory of the PE3011 tag the tab **"Tag Data Management"** can be used.



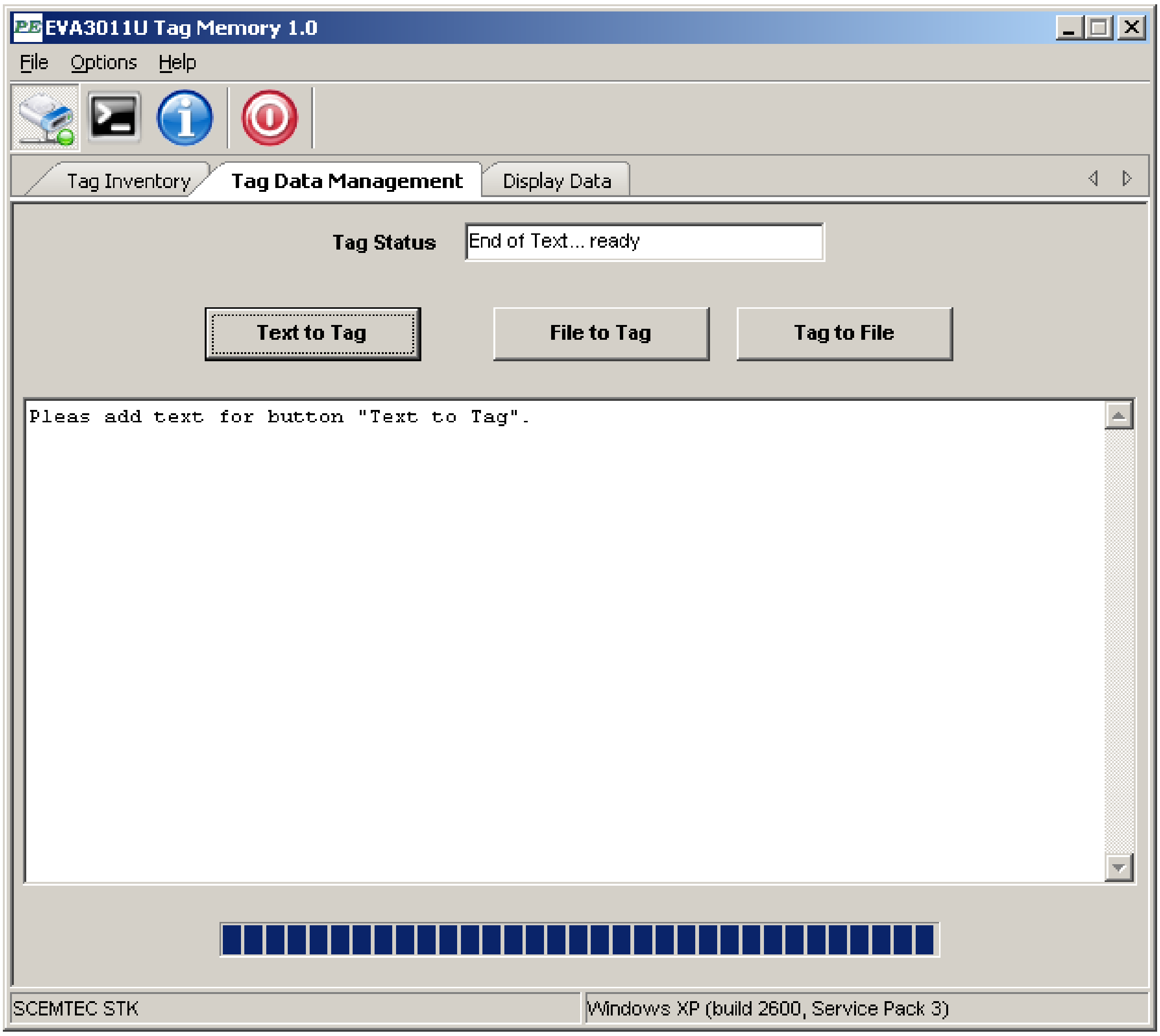
Tag Status

At first it is possible to see the field **"Tag Status".** This shows the current status of the tag in text field during any action on this tab.



Text to Tag

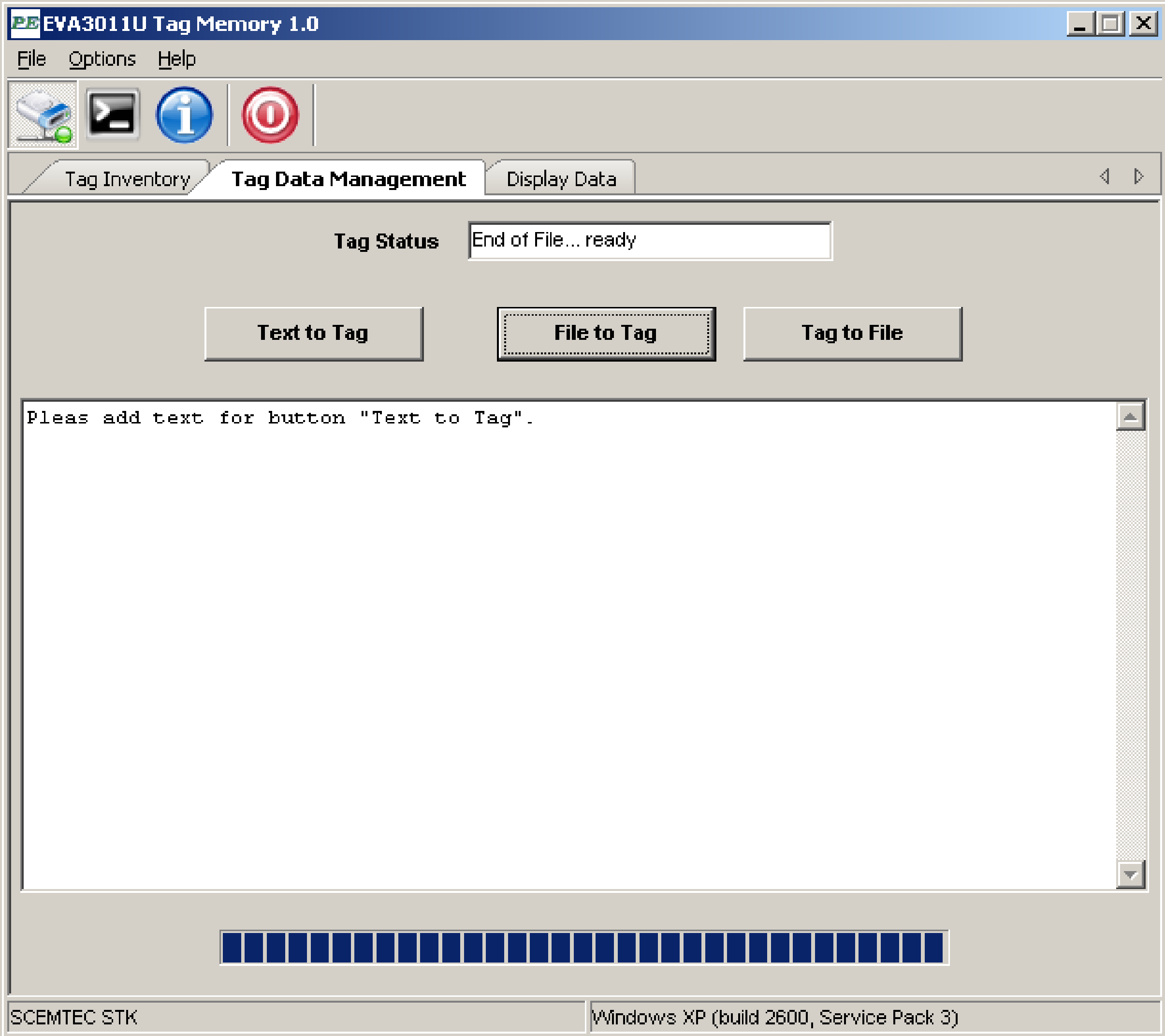
The button **"Text to Tag"** saves the text from the text field below in the tag. After this event the PE3011 tag contains the saving date, the length of data and the data itself as ASCII-Code in the USER-Bank of the tag.



Note: If the text is larger than the memory space the text will be truncated.

File to Tag

Push the button **"File to Tag"** to save data from text file in the tag memory.

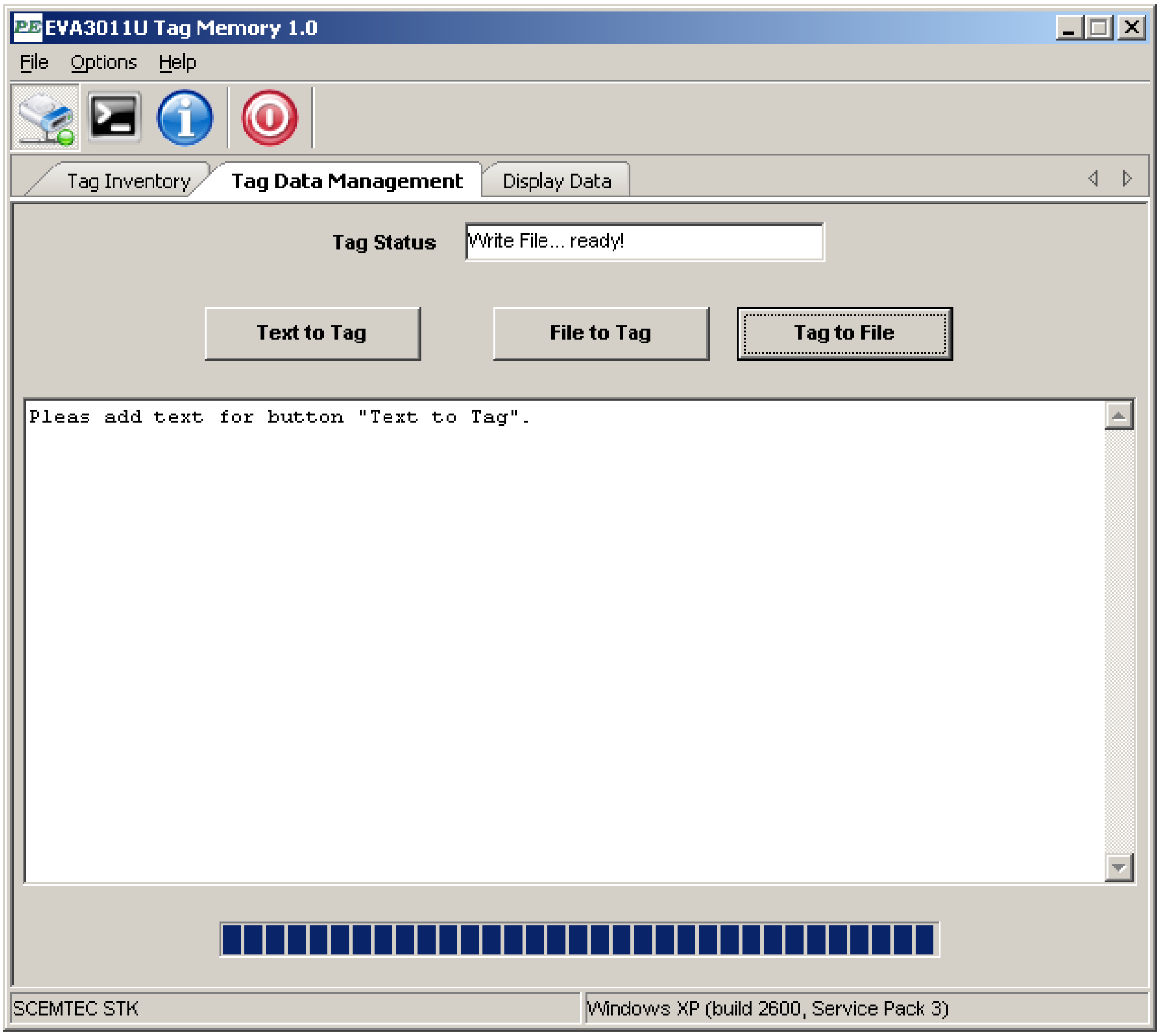


Note: If the text is larger than the memory space the text will be truncated.

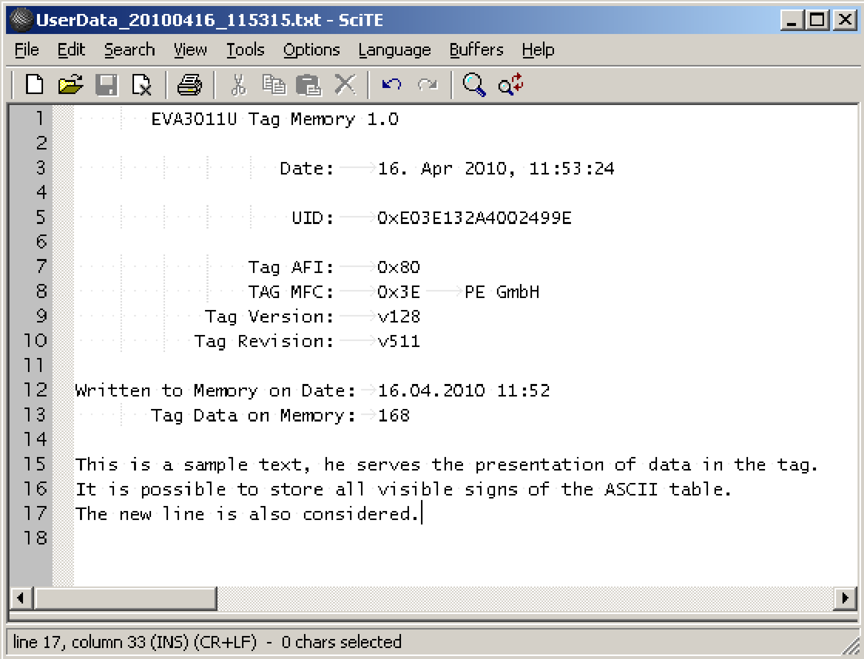
Tag to File

After successful saving of data it is possible to write detailed information to a text file in "UserData_YYYYMMDD_hhmmss.txt" format.

Writing data to and reading back data from the PE3011 can also be done without any battery support.



The current status of User Data is displayed below. The fields show the read values, the saving time and counted length of data. At the end of the file the saved ASCII data will be provided.



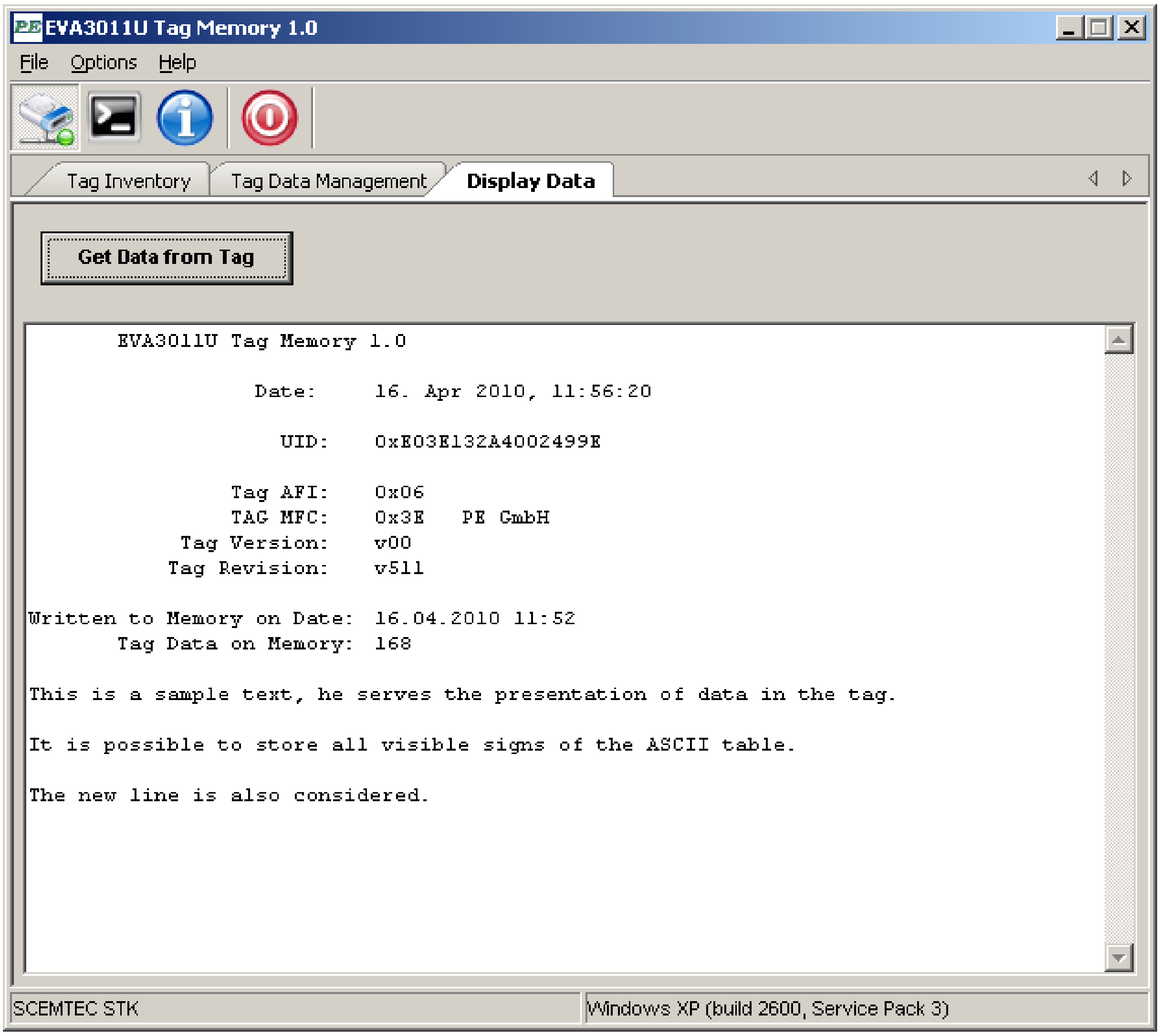

```
        EVA3011U Tag Memory 1.0

                    Date:   16. Apr 2010, 11:53:24

                     UID:   0xE03E132A4002499E

                 Tag AFI:   0x80
                 TAG MFC:   0x3E    PE GmbH
             Tag Version:   v128
            Tag Revision:   v511

Written to Memory on Date:  16.04.2010 11:52
       Tag Data on Memory:  168

This is a sample text, he serves the presentation of data in the tag.
It is possible to store all visible signs of the ASCII table.
The new line is also considered.
```

3.4 Software description – Display Data

This tab has been programmed to display the contents of the tag memory (UID data, saving date, ASCII length) and the stored ASCII data.

Get Data from Tag

It is possible to use the button **"Get Data from Tag",** this clears the old data and shows the current content of the RFID tag.



4 Memory definition

The memory for the Data Monitor is in the RFID IC and is organized as specified in the ISO15693 standard. The monitor configuration and the trimming values are stored in the MONREG bank.

Table 1 – EEPROM definition

Tag Bank	EEPROM Block	Data
UID	00h	VICC, DSFID, Flags ISO15693 conform
	01h	AFI ISO15693 conform
	02h	UID[63:32] ISO15693 conform
	03h	UID[31:0] ISO15693 conform
	04h-0Bh	LOCK ISO15693 conform
	0Ch-0Dh	RFU
	0Eh	TempScaling
	0Fh	MonCycle
MONREG	10h	MonCount
	11h	ExtSensFlags
	12h	Temp, RTC
	13h	TMS
	14h	Area
	15h	TimeDef
	16h	IntDataX
	17h	IntDataY
	18h	CoeffNormX
	19h	CoeffNormY
	1Ah	CPCN
	1Bh	CalibConst
MONDAT Read Only!	1Ch	IntDatExtSense
	…	IntDatExtSense
	1Ch+IntDataArea-1	IntDatExtSense
	1Ch+IntDataArea	MonitorData
	…	MonitorData
	MonDataEnd	MonitorData
USER Bank 11b	MonDataEnd+1	UserDat
	…	UserDat
	FFh	UserDat

Generally all measured sensor data will be stored in MONDAT-Bank. In this application this function is not used. For this application the USER-Bank will be used. The structure is stored with saving date, text length and ASCII data. The edge between USER and MONDAT-Bank can be changed by configuration byte MonDataEnd. In this application the MonDataEnd is static defined as.

222 addresses are reserved for recording text data in the memory. The total number of values to be stored is 888.

Note: PE GmbH delivers samples and production volume ICs in calibrated condition (RTC, TMS). The nature of the chip allows everybody to access AND change these values at any time as long as the MONREG-Bank is not locked. If the MONREG-Bank will be locked the data monitoring setup can not be changed any more. The values are trimmed in the test process during manufacturing and are guaranteed to be within specified limits.

4.1 Header for PE3011 Tag definition in C++

The complete source code is available from Productivity Engineering GmbH.

```
/****************************************************************
 * Name:     tagdefs.h
 * Purpose: tag defines
 * Created: 02-2010
 ***************************************************************/

#ifndef __tagdefs__
#define __tagdefs__

// Sensor definitions
#define FSENSSLOPEDIV   1
#define FSENSSLOPESINGLE   2
#define FSENSOFFSET  127
#define FSENSFEMTO2PICO  0.001
#define FYSENSSLOPE   0.1
#define FYSENSOFFSET  0
#define BANK_USER     0x03
#define BANK_RESERVED 0x00
#define ICMFG_ADR     0xA0
#define ICMFG_DATA    0x3E


// memory size
#define MONREG_INTDATAAREA 0x00
#define MONREG_MONDATAEND  0x1d
#define USER_DATA          0x04

// memory bank size
#define MIN_UID       0x00
#define MAX_UID       0x0F
#define MIN_MONREG    0x10
#define MAX_MONREG    0x1B
#define DELTA_MONREG MAX_MONREG - MIN_MONREG
#define MAX_USER      0xFF // user bank size (USER_SIZE = MAX_USER - MONDATAEND)
#define MIN_MONDAT    0x1C // reserved bank size (RESERVED_SIZE = MIN_MONDAT + MONDATAEND)
#define MAX_MONDAT    0xFF

// memory addresses epc bank
#define UID_FIRST       0x00
#define UID_TMPSCALING 0x0E
#define UID_UID2        0x02
#define UID_UID1        0x03
#define UID_AFI         0x01

// memory addresses tid bank
#define MONREG_FIRST      0x10
#define MONREG_MONSTATUS  0x10 - MONREG_FIRST
#define MONREG_SENSFLAGS  0x11 - MONREG_FIRST
#define MONREG_TEMPLIMIT  0x12 - MONREG_FIRST
#define MONREG_AREA       0x14 - MONREG_FIRST
#define MONREG_TIMER      0x15 - MONREG_FIRST
#define MONREG_XSENSELIM  0x16 - MONREG_FIRST
#define MONREG_YSENSELIM  0x17 - MONREG_FIRST
#define MONREG_NORMX      0x18 - MONREG_FIRST
#define MONREG_NORMY      0x19 - MONREG_FIRST
#define MONREG_CPCN       0x1A - MONREG_FIRST
#define MONREG_CALIB      0x1B - MONREG_FIRST
#define MONREG_MONSTATUSADR  0x10
#define MONREG_SENSFLAGSADR  0x11
#define MONREG_TEMPLIMITADR  0x12
#define MONREG_AREAADR       0x14
#define MONREG_TIMERADR      0x15
#define MONREG_XSENSELIMADR  0x16
#define MONREG_YSENSELIMADR  0x17
#define MONREG_NORMXADR      0x18
#define MONREG_NORMYADR      0x19
#define MONREG_CPCNADR       0x1A
```

```
// memory addresses user bank
#define USER_STARTDATE 0xFF
#define USER_STARTDATEADR 0xFF
#define USER_DATALENGTHADR 0xFE
#define USER_DATALENGTH USER_DATALENGTHADR - MONREG_MONDATAEND - 2

// memory addresses reserved bank
#define MONDAT_FIRST   0x1C

// MONREG block status flag register
#define GET_TEMPSLOPE(reg)      ((reg & 0x0000FF00) >> 8)
#define GET_TEMPOFFSET(reg)     ((reg & 0x000000FF) >> 0)
#define GET_MONDATAEND(reg)     ((reg & 0x000000FF) >> 0)
#define GET_MONCOUNT(reg)       ((reg & 0x000000FF) >> 0)
#define GET_INTADDRCNT(reg)     ((reg & 0x00FF0000) >> 16)
#define GET_TAGAFI(reg)         ((reg & 0x000000FF) >> 0)
#define GET_FIRMA(reg)          ((reg & 0x00FF0000) >> 16)
#define GET_VERSION(reg)        ((reg & 0x00FF0000) >> 16)
#define GET_MSI(reg)            ((reg & 0x0000FFFF) >> 0)
#define GET_MONSTATUS(reg)      ((reg & 0x0000FF00) >> 8)
#define GET_STARTTIME(reg)      ((reg & 0xFFFF0000) >> 16)
#define GET_INTERVALLTIME(reg)  ((reg & 0x000000FF) >> 0)
#define GET_TEMPLIMITHI(reg)    ((reg & 0xFF000000) >> 24)
#define GET_TEMPLIMITLO(reg)    ((reg & 0x00FF0000) >> 16)
#define GET_MWSIZE(reg)         ((reg & 0x00000F00) >> 8)
#define GET_ALARMMONCNT(reg)    ((reg & 0x00FF0000) >> 16)
#define GET_SENSFLAGS(reg)      ((reg & 0x0000000F) >> 0)
#define GET_DIFFFLAGS(reg)      ((reg & 0x00000300) >> 8)
#define GET_INTDATAREA(reg)     ((reg & 0xFF000000) >> 24)
#define GET_INTSAMPLE(reg)      ((reg & 0x00FF0000) >> 16)
#define GET_INTMWSENSE(reg)     ((reg & 0x00000030) >> 4)
#define GET_INTDATAHI(reg)      ((reg & 0x0FFF0000) >> 16)
#define GET_INTDATALO(reg)      ((reg & 0x00000FFF) >> 0)
#define GET_CPX(reg)            ((reg & 0x00003F00) >> 8)
#define GET_CPY(reg)            ((reg & 0x3F000000) >> 24)
#define GET_CNX(reg)            ((reg & 0x0000003F) >> 0)
#define GET_CNY(reg)            ((reg & 0x003F0000) >> 16)
#define GET_GAIN(reg)           ((reg & 0x0FFF0000) >> 16)
#define GET_OFFS(reg)           ((reg & 0x00000FFF) >> 0)
#define GET_CALIBX(reg)         ((reg & 0x00000FFF) >> 0)
#define GET_CALIBY(reg)         ((reg & 0x0FFF0000) >> 16)
#define GET_RTC(reg)            ((reg & 0x0000FFFF) >> 0)
#define GET_CLK1M(reg)          ((reg & 0x0F000000) >> 24)
#define GET_ACTDATE(reg)        ((reg & 0xFFFFFFFF) >> 0)
#define GET_LOGWORD(reg)        ((reg & 0x00FF3FFF) >> 0)

// reserved block data (logsize = 0x1AE)
#define TEMP_START  0x1C // 384 blocks

// status values
#define MONSTATUS_STARTLOG    0x80
#define MONSTATUS_MEMOVERFLOW 0x20
#define MONSTATUS_ACTIVE      0x40
#define MONSTATUS_ALARM       0x04
#define MONSTATUS_BATT        0x08
#define MONSTATUS_OUTBAND     0x02
#define MONSTATUS_INTEXT      0x01
#define SENSFLAG_XSENSOR      0x02
#define SENSFLAG_YSENSOR      0x04
#define SENSFLAG_TEMP         0x01
#define SENSFLAG_INTMODE      0x08
#define DIFFFLAG_DIFFENX      0x01
#define DIFFFLAG_DIFFENY      0x02
#define CPN_CPENX             0x00004000
#define CPN_CPENY             0x40000000
#define CPN_CNENX             0x00000040
#define CPN_CNENY             0x00400000

// calibration values
#define CAL_TEMPSLOPEDIV  100.0 //
#define CAL_TEMPOFFSETDIV  1.0 //
#define CAL_XSLOPEDIV  0.200 // 200 fF/bit
#define CAL_XOFFSETDIV  0.0 //
#define CAL_YSLOPEDIV  0.200 // 200 fF/bit
```

```
#define CAL_YOFFSETDIV  0.0 //

#define CAL_LIMIT2CNTDIF  1000.0 //
#define CAL_LIMIT2CNTSINGLE  500.0 //

#define CAL_CPNSLOPE  0.180 // 180 fF/bit

#endif // __tagdefs__
```

5 Introduction of EVA3011 Hardware

Features PE3011:

- Passive RFID HF transponder chip with integrated sensor monitor
- Compliance with EPC Class 1 Generation 2 (UHF RFID version 1.0.9)
- 8kBit EEPROM read-/writeable access about RF field and/or sensor monitor
- EEPROM memory for capture measurement data with time stamp
- Power supply about RF field (for communication) or with battery (sensor monitoring)
- Intelligent power management for different power domains
- Continuous battery control and automatic shutdown
- Extra signal output for “out-of-limits” detection
- Additional function with SPI interface (for using external devices)
- Real Time Clock to provide an accurate clock signal of 8.738 kHz
- Internal temperature sensor and ADC for external capacitive sensors

5.1 PE3011 General Description

The PE3011 is an integrated circuit for tracking and controlling logistics. It monitors temperature, extern capacitive sensor data and related time data of goods during transport or storage. While not in an RFID reader field and so not being supplied through the reader the system draws the required energy from the battery. While in a HF reader field the system is supplied by the reader’s field energy and communicates to the reader based on the standard protocol. Besides standard ISO15693 communication additional functionality to read out temperature or other data is implemented. The integrated SPI interface allows a communication with other external devices like a microcontroller that can provide additional sensor functionality like digital interfaced sensors.

Memory access is granted through the HF reader controller as well as through the data monitoring controller. Both blocks have the same priority. No started memory access will be interrupted by a request for another access. The started access will be finished first before the new access request will be acknowledged. Detailed information can be found in the PE3011 datasheet.

6 How to design an application

6.1 Given PCB specification

The PCB (printed circuit board) setup in the original state has the following specification parameters. It consists of a network for antenna matching, a battery with corresponding connector to the PE3011, a defined port for external capacitive sensors and an interface for SPI communication.

Supply voltage (battery)	3.3V

It is possible to assemble more devices for additional tasks. This implies an optical indication for an active alarm port.

6.2 SPI Interface

When using a microcontroller with SPI it is possible use also external digital sensors. The measurement data of these sensors can be written to the EEPROM via the SPI and can be read out through the RFID interface.

Note: Refer to the datasheet of PE3011 for read/write commands to EEPROM via SPI.

6.3 Alarm Port

The alarm port becomes active, when user defined sensor limits are violated and a measurement cycle is running. Refer to the datasheet of PE3011 for an active alarm. For getting an optical indication when the alarm port is activated, it is necessary to assemble resistor R4 and the LED.

When the alarm port gets activated, the signal is switched to ground and the LED illuminates.

<u>Calculation for series resistor:</u>

$$R_4 = \frac{V_{BAT} - V_F}{I_F}$$

For a red LED with low current the following parameters have to be accounted for:

Forward voltage V_F=1,8V , forward current I_F=2mA, battery voltage 3V:

$$R_4 = \frac{3V - 1{,}8V}{2mA} = 600Ohm$$

R_3=560Ohm shall be chosen here.

Note: The LED can only be applied, when using a 3V battery.

7 Description

The Evaluation Kit was designed to help understand and evaluate the features of the PE3011 HF RFID IC. Used external devices are standard components and do not represent a completely fine-tuned OEM application. The BOM (bill of material) for a final application may look different.

The PE3011 is a passive HF RFID transponder circuit with an integrated data monitor.

7.1 Evaluation Board Specifications

Table 2 – Operation Parameter

Parameter	Symbol	Min	Typ	Max	Unit	Notice
Operating temperature	T_{amb}	-40	27	80	°C	
Frequency RF field	f_{rf}		13,56		MHz	
Battery Voltage	V_{BAT}	2,4	3,3	3,6	V	
Current Alarm Port (PE3011)	I_{Alarm}		5		mA	Open drain output

7.2 Schematic

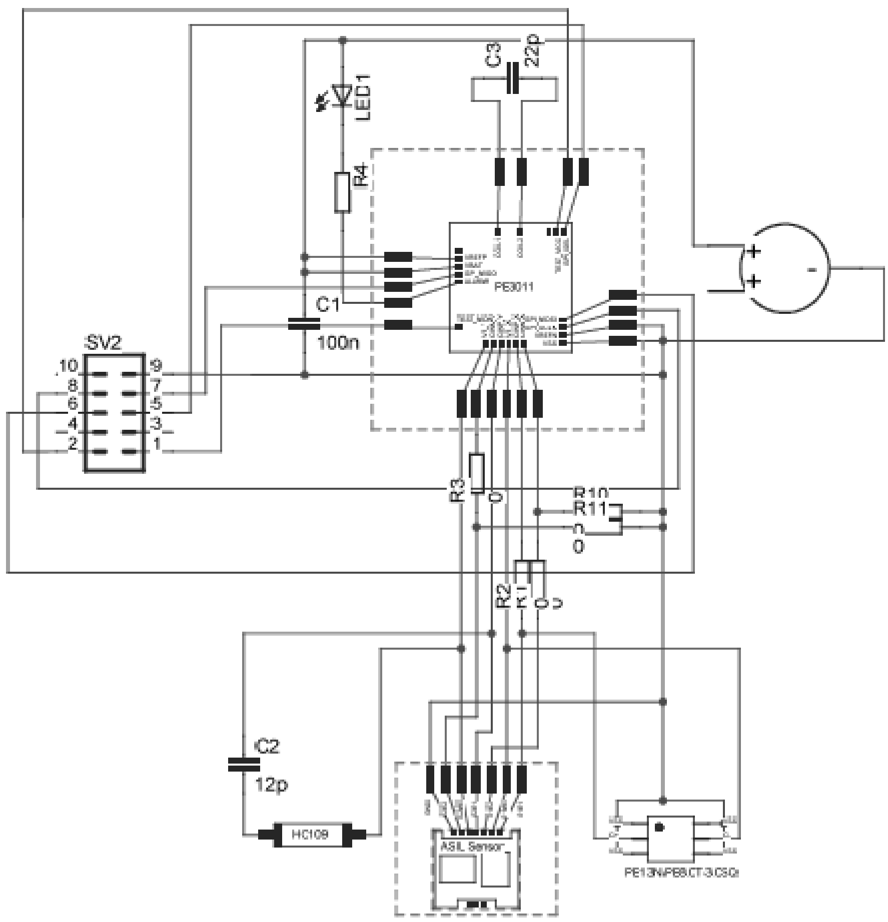


Figure 2 - Schematic of the Evaluation Kit

7.3 Layout

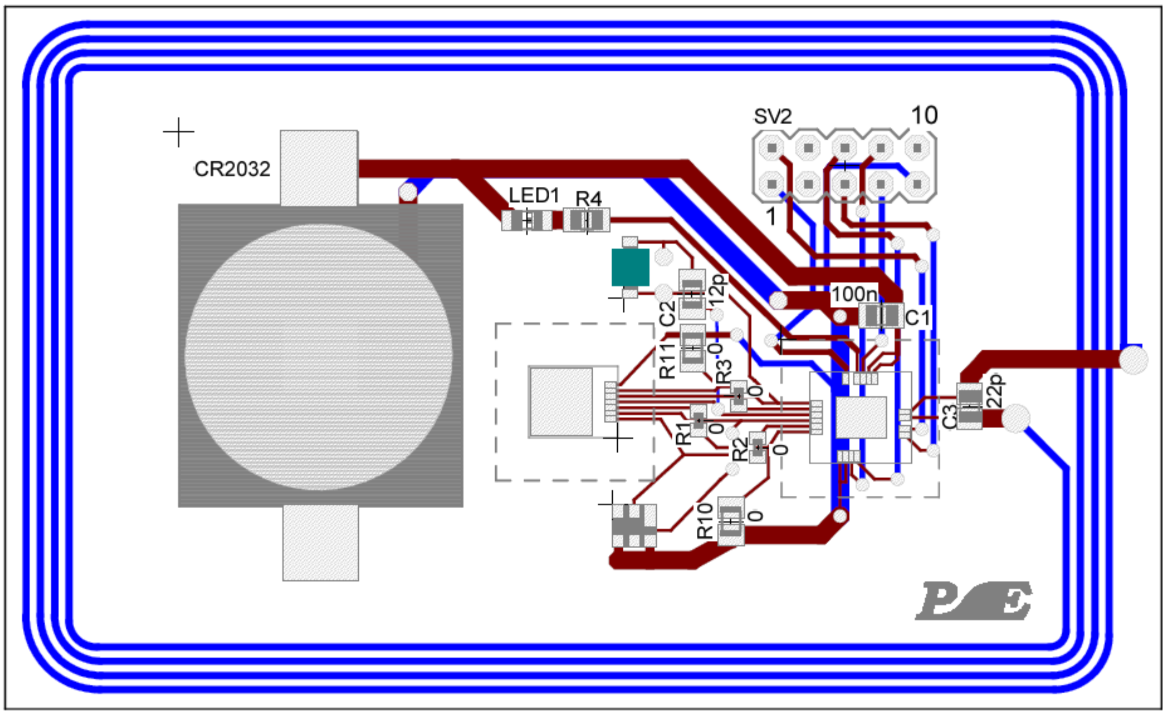


Figure 3 - Layout of the Evaluation Kit

8 Operation

8.1 Using only HF interface

The HF interface is accessible at any time on the EVA3011. The battery has not to be populated for this operational mode. The PE3011 can be controlled with a HF Reader. The antenna network is optimized for good electro-magnetic performance of the antenna with the chip. It is recommended to use the GUI3011 software to configure the PE3011 registers. The chip shall respond on any standard ISO15693 Reader by sending the UID.

The configuration and measurement commands are described in the PE3011 datasheet.

8.2 Using only data monitor

For using only the data monitor the battery has to be mounted (default).
The PE3011 can be controlled via the SPI interface or the RF field when the GUI3011 software will be used.

The configuration and measurement commands are described in the PE3011 datasheet.

8.3 Using HF interface and data monitor

For using both HF interface and data monitor the same requirements are necessary as described in chapter 8.2

This mode is recommended for measurement tasks.

Accessing the EEPROM is possible in two ways (RFID interface and data monitor). Both have the same priority. If an access to the EEPROM occurs, then memory is locked until the access finished.

The configuration and measurement commands are described in the PE3011 datasheet.

9 Notes

10 Contact

Germany

Stuttgart

Productivity Engineering
Process Integration GmbH
Behringstrasse 7
D-71083 Herrenberg
Germany
Phone.: +49 (0) 70322798 0
Fax: +49 (0) 70322798 29
Email: info@pe-gmbh.com
Web: www.pe-gmbh.com

Dresden

Productivity Engineering GmbH
Branch
Sachsenallee 9
D-01723 Kesselsdorf
Germany
Phone.: +49 (0) 35204777 00
Fax: +49 (0) 35204777 000
Email: info@pe-gmbh.com