UM2066
User manual

Getting started with the STPM3x

Introduction

This user manual describes guidelines to quickly set up a metrology application using the STPM3x devices (STPM32, STPM33 and STPM34) and to access measurement data from the device.

Further details on each step of the application design process (schematics, layout, calibration etc.) can be found in the reference documents listed below:

STPM32, STPM33, STPM34 datasheet

AN4470 (application note about the STPM3x application calibration)

UM1748 (user manual of the EVALSTPM34, EVALSTPM33, EVALSTPM32 evaluation board)

UM1719 (user manual of the STPM3x evaluation software)

Contents

List of tables

List of figures

1 Application block diagram

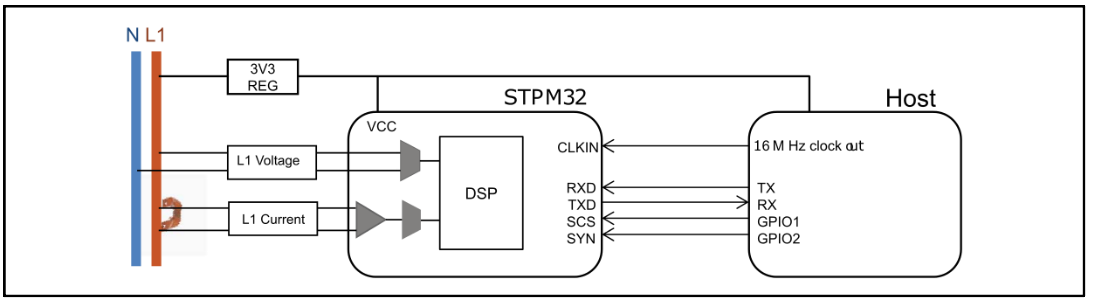
1.1 Monophase system

The STPM32 (single phase device) has been used in the application block diagram below. The STPM33/34 can be used in a mono phase system with CT sensors, but also with shunt, if there is a common electrical node between both channels.

1.1.1 UART interface

To select UART peripheral at startup, SCS signal must be set high before V_{CC} and EN rise. SCS must be maintained high until a couple of clock period (16 MHz) reaches the device. Then, the device is locked in UART mode until a reset by EN pulse or a new power-on sequence is performed. UART connections between host and the STPM3x are described below.

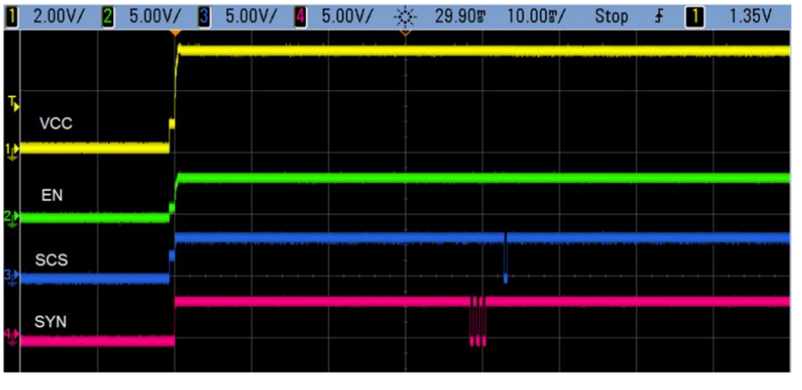
Figure 1: Connection through UART peripheral



16 MHz clock signal can be replaced by an external crystal with the same frequency between XTAL1 and XTAL2 pins. Please refer to the datasheet for connection details.

SCS signal is used to reset communication peripheral and SYN signal is used to reset the DSP. These reset pulses must be generated once, just after the power-on sequence.

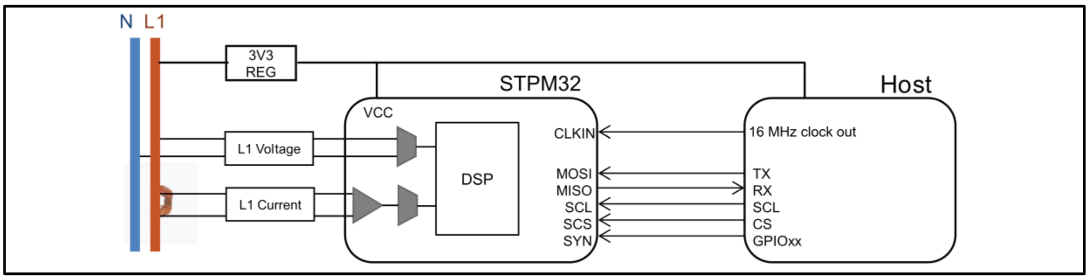
Figure 2: Timing example



1.1.2 SPI interface

To select SPI peripheral at startup, SCS signal must be set held low before V_{CC} and EN rise. SCS must be kept low until a couple of clock period (16 MHz) reaches the device. The device is locked in SPI mode until a reset by EN pulse or a new power-on sequence is performed. Then, SCS must be driven low during each data exchange according to usual SPI protocol. SPI connections between host and the STPM3x are described below.

Figure 3: Connection through SPI peripheral



16 MHz clock out signal can be replaced by an external crystal with the same frequency between XTAL1 and XTAL2 pins. Please refer to the datasheet for connection details.

SYN signal is used to reset the DSP. Three reset pulses must be generated once, just after the power-on sequence.

1.1.3 Power-on timing

Recommendations for the power-on timing including reset pulses are provided below.

Figure 4: Power-on timing

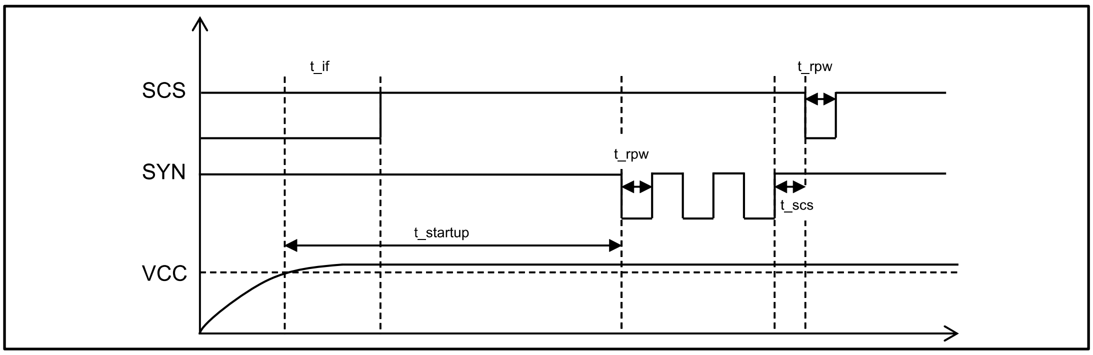


Table 1: Power-on procedure

Symbol	Parameter	Min.	Typ.
t_if	Time for interface choice locking	4 µs	10 ms
t_startup	Time between power-on and reset	35 ms	35 ms
t_rpw	Reset pulse width	4 µs	1 ms
t_scs	Delay from SYN to SCS	4 µs	1 ms

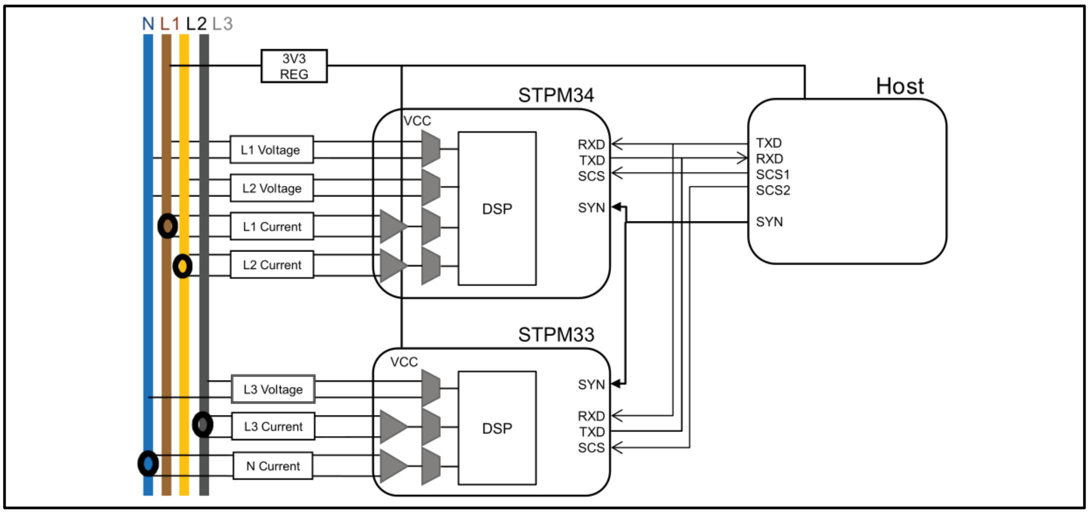
1.2 Polyphase systems

1.2.1 Current transformer Rogowski coil sensors

When CT or Rogowski coil transducers are used for the current sensing, since they isolate each line connected to the STPM3x, an additional isolation is not needed.

A connection example in UART mode in the figure below:

Figure 5: Polyphase system with CTs



As for monophase systems, SCS signal is used to reset UART block. If a single UART is used on host side, SCS is also the chip select signal when high (1 SCS per chipset should be connected).

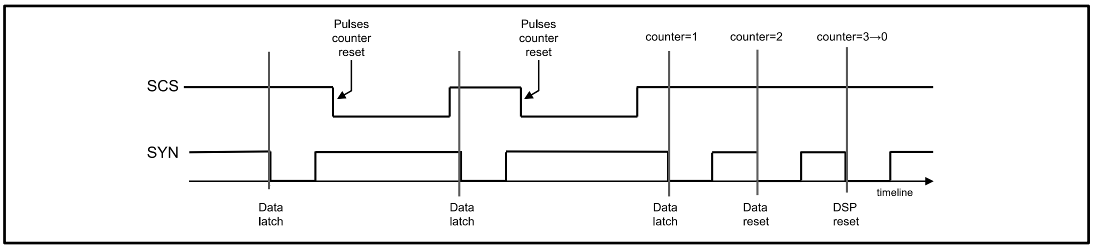
Three pulses of SYN signal are used to reset the DSP at first power-on. After the power-on sequence, SYN signal can be used to synchronize the measurements among all phases (see section below).

1.2.2 SYN synchronous data latch

SYN latching can be used to synchronously sample all data results (for all phases) into transmission latches. The transmission latches are flip-flops holding the data in the communication interface, so the host can then read the register values without time constraint, while DSP registers of the device are continuously updated. Note that SCS must be high during the pulses.

As SYN pin is also used to reset data registers and the DSP, a state machine counts the number of pulses in order to correctly interpret the command.

Figure 6: SYN functionality

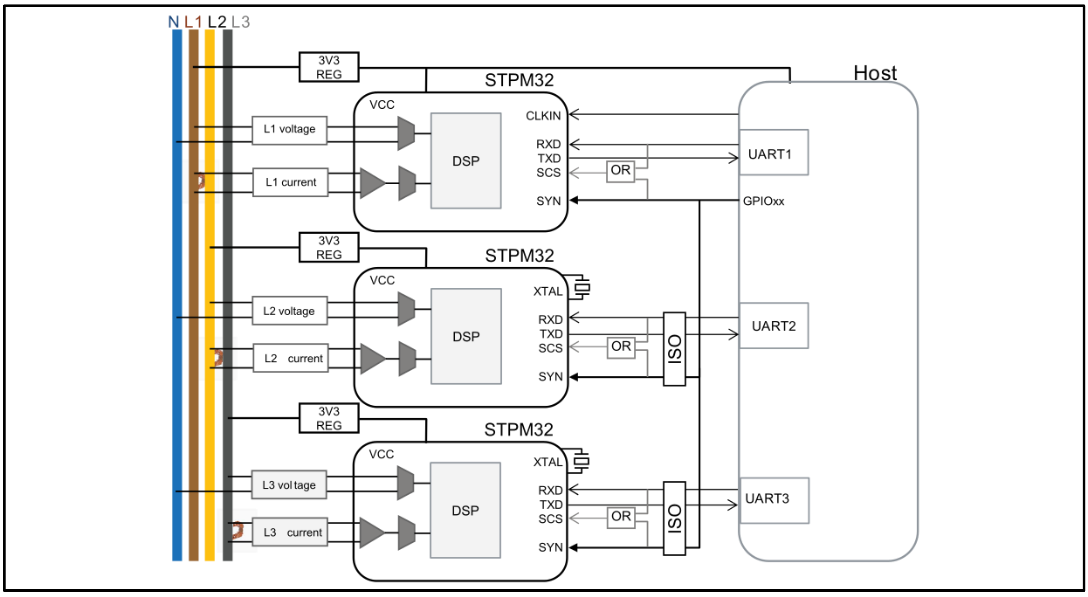


After each data latch, the pulse counter must be reset by a pulse on SCS.

1.2.3 Shunt sensors (adding isolation)

If the shunt resistor is used for current sensing, the STPM3x is connected to different phases, so they must be isolated one from the others.

Figure 7: Polyphase system with shunt



To limit the number of lines to isolate, SCS pulse can be generated from RXD + SYN, by adding an OR gate or two diodes only.

To implement global reset, at startup the FW should perform the following procedure:

- Send 3 pulses on SYN
- Send 1 pulse on SCS:
  - set SYN to “0”
  - send 0x00 on TXD (micro side)
  - wait a delay
  - set back SYN to “1”

Example code:

```
GPIO_ResetBits( GPIO9, SYN_STPM32)
USART_SendData(USART3,0x00)
INIT_WaitMicroSecond(10000)
GPIO_SetBits( GPIO9, SYN_STPM32)
```

1.3 Schematics

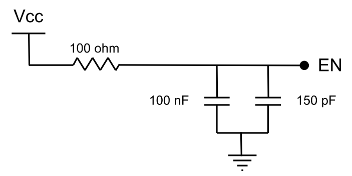
Some filtering components must be added to final schematics. Evaluation board schematics could be some examples:

On st.com website, refer to:

- EVALSTPM33/34_schematics
- EVALSTPM32_schematics
- Evaluation board layout

EN pin should be filtered to pass EMC surge tests as shown in evaluation board schematics.

Figure 8: Filter on EN pin



1.4 Register conversion to real value

1.4.1 Circuit parameters

In the metrology section of the application only analog front end parameters can be changed. Their values are needed to adapt signal dynamic to the device input range, and therefore are important to convert register values to “real” values:

- Resistors R_1 and R_2 used to divide input voltage (Ω)
- Sensor sensitivity K_S used to measure the current (mV/A)
- Current sensor gain A_I to be set in the register DFE_CR1 and DFE_CR2 (2, 4, 8 or 16)
- LED division factor (LPW) to be set in the register DSP_CR1 and DSP_CR2 (LPW=4 by default)

The STPM3x evaluation software provides a “Design Wizard” tool which can be used to correctly select these parameters according to the input signal range.

1.4.1.1 R_1 and R_2 choice

R_1 and R_2 are voltage divider resistors, used to reduce the magnitude of the line voltage so it can fit the STPM3x ADC input level dynamic.

The values of R_1, R_2 must be chosen according to the expected voltage range (maximum value expected on input circuit), to keep the peak input voltage lower than the STPM3x input range so to avoid the STPM3x ADC saturation.

The input range for ADCs with amplification stage of gain A_V is normally $\pm V_{REF}/A_V$, but for second order sigma-delta ADC only half of this range is linear, then the valid input range to be considered is $\pm V_{REF}/2A_V$. For the STPM3x, this range, as specified in the datasheet, is ±300 mV.

This means:

$$V_{MAX\_RMS} \cdot \sqrt{2} \cdot \frac{R2}{R1 + R2} < \frac{V_{REF}}{2A_V}$$

Then R_1 and R_2 should be chosen as follows:

$$\frac{V_{REF} \cdot (R1 + R2)}{2 \cdot \sqrt{2} \cdot R2 \cdot A_V} > V_{MAX\_RMS}$$

Example:

In ST's evaluation boards, R_1 = 810 kΩ and R_2 = 470 Ω

- Max. input voltage = 366 V_{RMS}
- Register LSB value = 36 mV_{RMS}

1.4.1.2 K_s and A_i choice

K_S, the current sensor sensitivity, and A_I, the current channel gain, have an impact on range and resolution of current to be measured by the STPM3x ADC. Their values must be chosen according to the desired current range (maximum value expected on the input circuit). The higher the sensitivity and the channel gain are, the higher the resolution of the current measurement is and therefore the accuracy at low currents, but the maximum measurable current is lower.

The input range of the ADC with amplification stage with gain A_I is $\pm V_{REF}/2A_I$. This range, as specified in the datasheet, is either ±300 mV, ±150 mV, ±75 mV or ±37.5 mV (respectively, for gain x2, x4, x8, x16).

This means:

$$I_{MAX\_RMS} \cdot \sqrt{2} \cdot Ks < \frac{V_{REF}}{2A_I}$$

Then K_S should be chosen as follows:

$$\frac{V_{REF}}{2 \cdot \sqrt{2} \cdot Ks \cdot A_I} > I_{MAX\_RMS}$$

The minimum current is usually an important application parameter, it is calculated four times the current RMS register LSB:

$$I_{MIN\_RMS} = \frac{4V_{REF}}{0.875 \cdot Kint \cdot Ks \cdot A_I \cdot 2^{17}}$$

RMS register resolution is reported above. Power and energy have higher resolution and accuracy at low currents, because they are calculated directly from raw voltage and current ADC data, which have 28 bit resolution.

Example:

In ST's evaluation boards, K_S = 2.4 mV/A and A_I = 16

- Max. input current = 11 A_{RMS} (88 A_{RMS} with A_I = 2)
- Register LSB value = 0.27 mA_{RMS} (2.2 mA_{RMS} with A_I = 2)
- Minimum input current = 1.09 mA_{RMS} (8.72 mA_{RMS} with A_I = 2)

1.4.1.3 LPW choice

The LED output can be used for calibration and/or for energy-reading. LPW bits divide LED output pulse frequency. A lower LPW value increases the number of pulses per kWh (C_p).

If the application needs a fixed C_p (e.g. 64000 pulses/kWh), please refer to datasheet section “Application design” and to “Design Wizard” tool provided by the evaluation software. If LED output is not used, the default register value of LPW can be left.

All analog front end parameters (R_1, R_2, K_S, A_I) have also an impact on the final C_p.

1.4.2 Conversion formula examples

All formulas reported below are present in the datasheet.

1.4.2.1 Voltage and current measurements

Measured values can easily be calculated by multiplying the decimal values contained in the register by the LSB value. For I_{RMS} and V_{RMS} for instance:

Figure 9: RMS register

Row	Address	(R)ead (W)rite (L)atch	Data								Name
			31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0	
36	48	RL	C1 RMS Data [16:0]					V1 RMS Data [14:0]			dsp_reg14

Note that $V_{RMS}[14:0]$ is 15 bit long and $C_{RMS}[16:0]$ is 17 bit long.

LSB values are provided by formulas below:

Voltage RMS LSB value:

$$LSB_{VRMS} = \frac{V_{ref} \cdot (1 + \frac{R_1}{R_2})}{cal_v \cdot A_v \cdot 2^{15}} [V]$$

Current RMS LSB value

$$LSB_{IRMS} = \frac{V_{ref}}{cal_I \cdot A_I \cdot 2^{17} \cdot K_s \cdot K_{int}} [A]$$

For example, if DSP_REG14 = 0x00F4 1652, converted values are:

- $V1_{RMS}$ = Dec [0x1652] * V_{ref} * (1+R_1/R_2) / (cal_V * A_V* 2^{15})

$V1_{RMS}$ = 5714 * 1.2 * (1+810000/470) / (0.875 * 2 * 2^{15}) = 206.2 V

- $I1_{RMS}$ = Dec [0x01E8] * V_{ref} / (cal_I * A_I * 2^{17} * K_S * k_{int})

$I1_{RMS}$ = 488 * 1.2 / (0.875 * 2 * 2^{17} * 2.4 * 10^{-3} * 1) = 1.06 A

1.4.2.2 Other measurements

For each register, its LSB value is reported in the datasheet. If the register is unsigned, the calculation is the same as for RMS values.

If the register is signed, it has to be binary complemented to get the value, then this signed value can be multiplied by register LSB. For example, if the value of DSP_REG3= 0x0002B99B, converted value of current is:

- $I1_{MOM}$ = SignedDec[0x0002B99B] * V_{ref} / (cal_I * A_I * 2^{23} * K_S * k_{int})
  $I1_{MOM}$ = 178587 * 1.2 / (0.875 * 2 * 2^{23} * 2.4 * 10^{-3} * 1) = 6.08 A

If the value of DSP_REG3 = 0xFFFFF414, converted value of current is:

- $I1_{MOM}$ = SignedDec[0xFFFFF414] * V_{ref} / (cal_I * A_I * 2^{23} * K_S * K_{int})
- SignedDec [0xFFFFF414] = -1*(2^{24}- Dec [0xFFF414]) = -1 * (16777216-16774164) = -3052
- $I1_{MOM}$ = -3052 * 1.2 / (0.875 * 2 * 2^{23} * 2.4 * 10^{-3} * 1) = -0,104 A

In this case, the most significant byte is just padding.

An excel file is available on demand, contact: smart-grid-emea@st.com to assist you about other value calculations (power, energy, LED frequency, register overflow).

2 UAR SPI communication protocol

2.1 Data communication protocol

In this section UART and SPI protocol are described as per default configuration of the STPM3x.

Default configuration for SPI is as follows:

- Polarity = 1
- Phase = 1
- CRC enabled = true (can be disabled)
- CRC poly = 0x07 (can be changed)
- MSB first (can be changed)

Default configuration for UART is as follows:

- Baud rate = 9600 (can be changed)
- Parity: none
- Stop: 1 bit
- CRC enabled = true (can be disabled)
- CRC poly = 0x07 (can be changed)
- LSB first

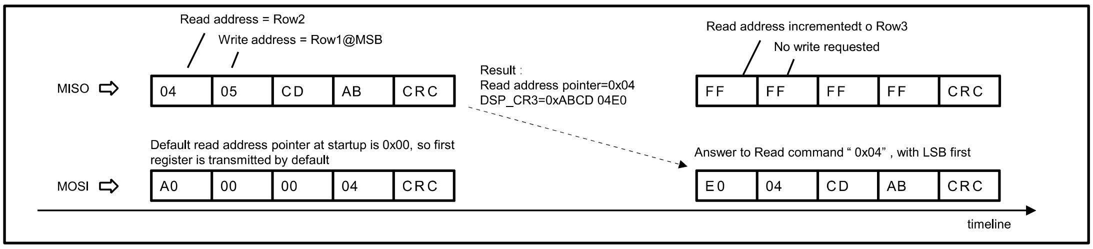
Please refer to datasheet for details about different settings.

At startup, the memory reading pointer is set to row at address 0.

Communication example in the default configuration, at startup:

- Default value of first register = 0x040000A0
- Default value of dsp_cr3 (address=0x04) = 0x000004E0
- To write 0xABCD to MSB bytes and read back register value, the frames in the picture below have to be sent

Figure 10: Frame sent and received from the device



2.2 Bit flow on physical links

SPI: if the first 4 bytes of previous example are probed, at a bit level, the following happens.

Figure 11: SPI bit flow

MOSI	0 0 0 0 0 1 0 0	0 0 0 0 0 1 0 1	1 1 0 0 1 1 0 1	1 0 1 0 1 0 1 1
	0x04	0x05	0xCD	0xAB
MISO	1 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0
	0xA0	0x00	0x00	0x04

UART: for the same example, at a bit level the following happens.

Figure 12: UART bit flow

MOSI	0 0 1 0 0 0 0 0	1 0 1 0 0 0 0 0	1 0 1 1 0 0 1 1	1 1 0 1 0 1 0 1
	0x04	0x05	0xCD	0xAB
MISO	0 0 0 0 0 1 0 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0
	0xA0	0x00	0x00	0x04

2.3 CRC calculation

CRC is calculated on the bit train (so from left to right) using a default polynomial 0x07 ($x^8+x^2+x^1+1$ = 100000111) with 0 as initial value.

As an example, given the following four bytes to send: 0xFF08FAF7 (which means writing to address 0x08 data value of 0xF7FA) the frame buffer is built as follows:

TX_Frame_buff[0] = 0xFF;

TX_Frame_buff[1] = 0x08;

TX_Frame_buff[2] = 0xFA;

TX_Frame_buff[3] = 0xF7;

TX_Frame_buff[4] = 0xXX; //CRC.

Data to write is sent least significant byte first.

Note that even if CRC received from the host is wrong, the answer of the STMP3x is in any case the default value of the 1st register (0x040000A0). Then, an interruption can be enabled to activate a wrong CRC detection (see datasheet for details).

2.3.1 CRC calculation for SPI

Here is the code of ‘C’ functions, they can be copied as they are in the application firmware:

```
TX_Frame_buff[4] = CalcCRC8(TX_Frame_buff);
#define CRC_8 (0x07)
#define STPM3x_FRAME_LEN (5)
static u8 CalcCRC8(u8 *pBuf)
{
  u8 i;
  CRC_u8Checksum = 0x00;
  for (i=0; i<STPM3x_FRAME_LEN-1; i++)
  {

    Crc8Calc(pBuf[i]);
  }
  return CRC_u8Checksum
}
static void Crc8Calc (u8 u8Data)
{
  u8 loc_u8Idx;
  u8 loc_u8Temp;
  loc_u8Idx=0;
  while(loc_u8Idx<8)
  {
    loc_u8Temp = u8Data^CRC_u8Checksum;
    CRC_u8Checksum<<=1;
    if(loc_u8Temp&0x80)
    {
      CRC_u8Checksum^=CRC_8;
    }
    u8Data<<=1;
    loc_u8Idx++;
  }
}
```

The CRC result for our frame example is: 0xDD.

SPI frame with CRC: 0xFF08FAF7DD.

Figure 13: SPI frame with CRC: 0xFF08FAF7DD

0xFF	0x08	0xFA	0xF7	CRC 0xDD
1 1 1 1 1 1 1 1	0 0 0 0 1 0 0 0	1 1 1 1 1 0 1 0	1 1 1 1 0 1 1 1	1 1 0 1 1 1 0 1

2.3.2 CRC calculation for UART

Due to the UART peripheral architecture, we must compute the CRC on reversed bits of the frame and reverse the CRC computed afterwards.

Here is the code of ‘C’ functions, they can be copied as they are in the application firmware.

The purpose is to reverse the bits of frame bytes, store them into a temp frame, compute the CRC of the temp frame, and finally reverse the bit of the CRC.

```
void FRAME_for_UART_mode(u8 *pBuf)
{
  u8 temp[4],x,CRC_on_reversed_buf;
  for (x=0;x<(STPM3x_FRAME_LEN-1);x++)
  {
    temp[x] = byteReverse(pBuf[x]);
  }
  CRC_on_reversed_buf = CalcCRC8(temp);
  pBuf[4] = byteReverse(CRC_on_reversed_buf);
}
static u8 byteReverse(u8 n)
{
  n = ((n >> 1) & 0x55) | ((n << 1) & 0xaa);
  n = ((n >> 2) & 0x33) | ((n << 2) & 0xcc);
  n = ((n >> 4) & 0x0F) | ((n << 4) & 0xF0);
  return n;
}
```

Applied to our example, UART CRC is **0xFC** (0x3F bit reversed calculated on 0xFF105FEF).

Figure 14: UART frame with CRC: 0xFF08FAF7FC

1 1 1 1 1 1 1 1	0 0 0 1 0 0 0 0	0 1 0 1 1 1 1 1	1 1 1 0 1 1 1 1	0 0 1 1 1 1 1 1
0xFF	0x08	0xFA	0xF7	CRC 0xFC
Reversed 0xFF	Reversed 0x10	Reversed 0x5F	Reversed 0xEF	Reversed 0x3F

3 Using the evaluation board

3.1 SPI interface: the EVALSTPM3x with USB board STEVAL-IPE023V1

The USB board includes a galvanic isolation between the metrology board and the embedded microcontroller, making safe the connection to a PC.

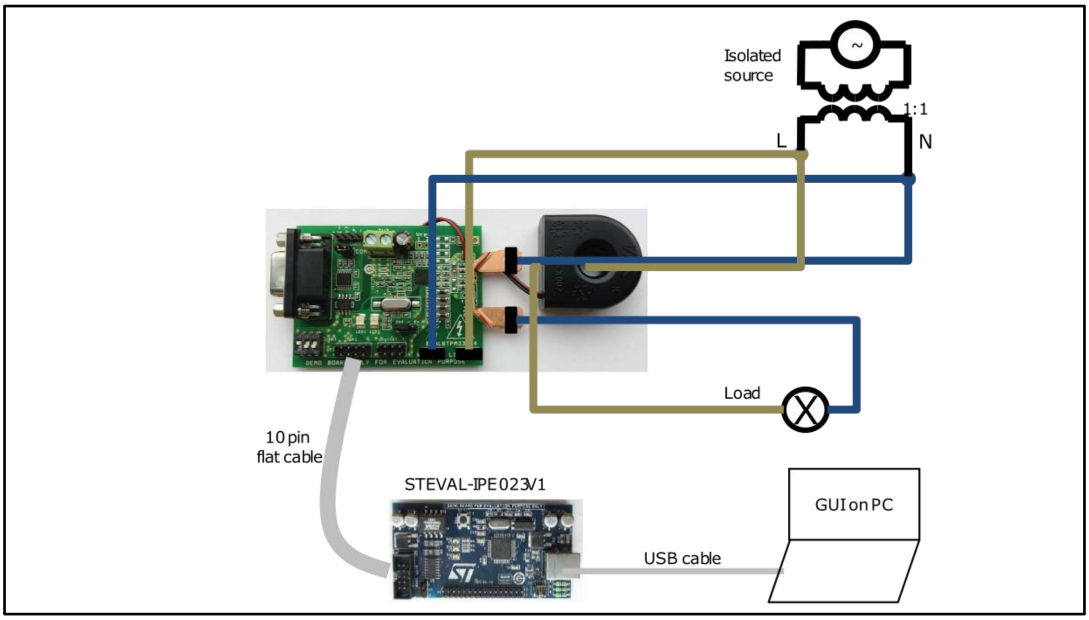
It also provides a 3.3 V, enough to supply the board.

The embedded microcontroller implements the bridge function from SPI (STPM3x side) to USB interface (PC side).

3.1.1 Electrical connection

The STPM34 evaluation board can be connected as below:

Figure 15: Connection to the STEVAL-IPE023V1



See user manual UM1748 for connection details.

USB board must be connected first to the PC through USB cable, then to the STPM3x evaluation board through flat cable. SWC1 must be in “off” position.

3.2 UART interface

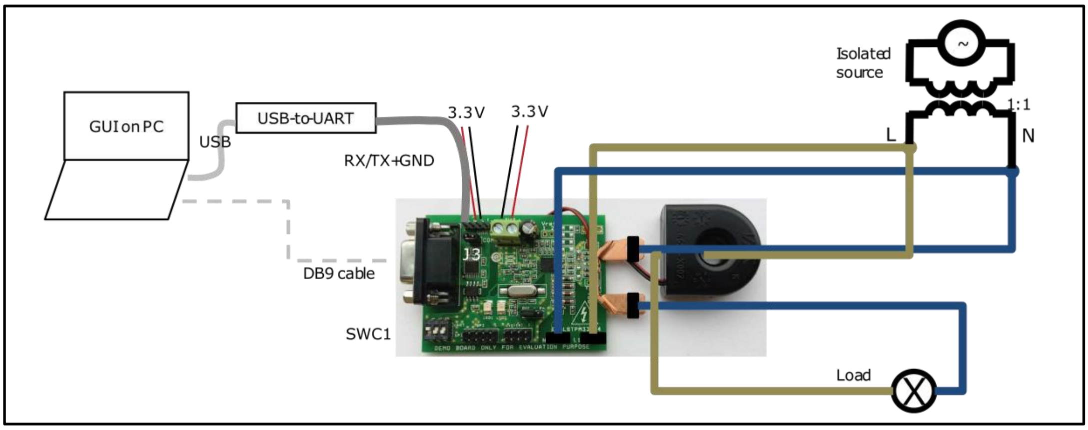
RS232 port is insulated from the board through Si8621 transceiver to safely connect the board to a PC.

Note that a 3.3 V must be provided to UART section in addition to the one to be provided to the rest of the board. In case of a shunt sensor, the last one must be insulated. In case the current sensor is CT or Rogowski coil, it can have the same power supply.

3.2.1 UART electrical connection

The STPM34 evaluation board can be connected as below:

Figure 16: Connection to UART interface



See user manual UM1748 for connection details.

SWC1 must be in “on” position; if DB9 cable is used, enable RS232 port by inserting J3 jumper.

3.3 Evaluation software

For both SPI and UART mode, see GUI user manual UM1719 on www.st.com.

4 Power measurements

Power data registers inside the STPM3x can be:

- Instantaneous measurements for active power only (ph1_reg10 to ph1_reg11)
- Filtered measurements (ph1_reg5 to ph1_reg9) for active, reactive and apparent power

4.1 Instantaneous power values

Power is defined as:

$$p(t) = v(t) \cdot i(t) = V_P cos(\omega t) \cdot I_P cos(\omega t + \varphi) = V_{RMS}\sqrt{2}cos(\omega t) \cdot I_{RMS}\sqrt{2}cos(\omega t + \varphi) =$$
$$= V_{RMS} \cdot I_{RMS} \cdot (\cos\varphi + \cos(2\omega t + \varphi))$$

where:

V_P is the peak voltage in volts

I_P is the peak current in amperes

V_{RMS} is the root-mean-square voltage in volts

I_{RMS} is the root-mean-square current in amperes

ω is the angular frequency

φ is the phase angle between the current and voltage sine waves

These values can be used to perform calculation on power harmonic content.

The bandwidth of wide band waveforms and measurements is 3.6 kHz, which means up to 72^{th} harmonics of a 50 Hz network. The bandwidth for the fundamental harmonic measurement is 80 Hz.

4.2 Filtered power values

Removing the ripple at 2ωt through a low-pass filter, the DC component of power is obtained:

$$P = V_{RMS} \cdot I_{RMS} \cdot \cos\varphi$$

Note that a residual ripple due to the cos (2wt+φ) component may still be present in power. It generates a 100 Hz oscillation around the real average value.

This oscillation has no impact on energy, as it is a pure sinusoidal power component which is integrated over time.

If a very accurate average power measurement is needed, it can be calculated as:

$$P = \frac{E_{t1} - E_{t0}}{t_1 - t_0}$$

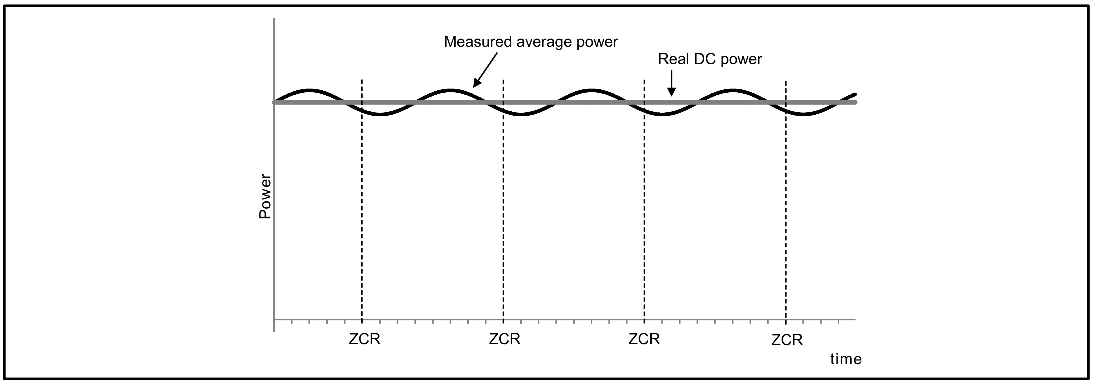
where:

$E_{t1} – E_{t0}$ are energy values taken in time instants t_0 and t_1

$t_1 – t_0$ time interval is a multiple of a line half-period (10 ms for 50 Hz)

In case the host does not have an accurate base of time, it can use ZC outputs from the STPM3x as an interruption to ensure that energy is always divided by a whole number of line periods. See plot below.

Figure 17: Power waveform



4.3 Power factors

Power factor is calculated as cosφ, where φ is the phase angle between voltage and current.

Phase angle value φ (in register Cx_PHA[11:0]) is calculated from zero-crossing information; it measures the delay between voltage and current for the fundamental harmonic. So this information of power factor is valid only for sinusoidal waveforms.

For non-sinusoidal waveforms, since power values may be affected by residual oscillation, an accurate power factor can be calculated as follows:

$$P = \frac{\Delta E_{ACT\_WB}}{\Delta E_{APP}}$$

where

$E_{t1} – E_{t0}$ are energy values taken in time instants t_0 and t_1, and $t_1 – t_0$ time interval is a multiple of a line half-period (10 ms for 50 Hz).

Apparent energy must be calculated from the RMS apparent power calculated as S_{RMS}* = V_{RMS}* I_{RMS}. It can be selected by AEM bit in register DSP_CR1 and DSP_CR2.

5 Calibration

See AN4470 for details about calibration procedure.

The procedure is automatically implemented in the STPM3x evaluation software available online.

An excel file is also available on demand (contact: smart-grid-emea@st.com), with automatic calculations of calibration values (CHV, CHI, PHV, PHC) based on your design parameters.

6 Revision history

Table 2: Document revision history

Date	Revision	Changes
01-Jun-2016	1	Initial release.
11-Sep-2017	2	Update *Section 1.2.3: "Shunt sensors (adding isolation)"*.