UM1070
User manual

C library for ST7570/80 and STM32™

1 Introduction

This document describes the complete library (driver) for interfacing the ST power line modem, ST7570, with a microcontroller. The library is designed to be universal and easily adapted to any microcontroller with minor small changes. The minimum requirements for the microcontroller are a UART and a timer. The library was tested and compiled for the STM32 microcontroller in IAR programming environment.

As an example of the complete communication node containing a modem and a microcontroller the STEVAL-PCC012V1 and EVALKITST7570-1 were used. An example application, which is part of the library, can be run on the mentioned communication node directly without any necessary adaptations. The library was tested on this communication node.

The library implements all the ST7570 commands that this device offers.

Using this library with other platforms and microcontrollers and its interconnection is described in the ST7570 datasheet, and the UM0934 and UM1038 user manuals, released together with this user manual.

www.BDTIC.com/ST

Contents

www.BDTIC.com/ST

List of tables

www.BDTIC.com/ST

2 Library details

- Project created in: IAR programming environment version 5.50
- Language: C
- Reference: ST7570 datasheet and ST7580 databrief - for all the details regarding the commands.
  UM0934 user manual, section 5.3.
  UM1038 user manual.

3 Definitions and constants (userinterface.c)

All the necessary constants are defined in this file.

4 Global data structure and function (in cmd_msg.h)

Global data structure exported from cmd_msg.h

```
struct ST7570_Status_type{
 unsigned char AccessLayerMode_PHY_MAC;
};

extern struct ST7570_Status_type ST7570_Status;
```

Table 1. ST7570_Status structure member description

Field	Note
AccessLayerMode_PHY_MAC	This field must be set by user. The value is specified using PHY or MAC modem layer. Values: PHY_layer = 1 or MAC_layer = 2.

Function exported from cmd_msg.h

Called by user:

int WRA_PlcInit(**int** iCom, **unsigned char** ucMode, **unsigned char** ucMode2, **unsigned char** ucDigitalGain, **unsigned long** ulSymbol_0, **unsigned long** ulSymbol_1, **unsigned char** ucADC, **int** iAccessLayerModeLoc, **unsigned int** uiCurrentControl);

www.BDTIC.com/ST

Table 2. WRA_PlcInit function parameters description

Parameter	Meaning
iCom	Com port used to communicate with PLM. The parameter is present only for legacy reasons; its value is not used.
ucMode	Node mode, baud rate, bit mapping, see ST7570 datasheet.
ucMode2	PII, see ST7570 datasheet.
ucDigitalGain	Target output gain, see ST7570 datasheet.
ulSymbol_0	Frequency of symbol “0”.
ulSymbol_1	Frequency of symbol “1”.
ucADC	Padding bytes for future needs.
iAccessLayerModeLoc	The value is specified using PHY or MAC modem layer. Values: PHY_layer = 1 or MAC_layer = 2.
uiCurrentControl	Value: 1 means current control enabled, see ST7570 datasheet, page 80.

Table 3. WRA_PlcInit function output values

Value	Meaning
1	No error, correct ACK byte received.
-101	NAK received from the modem.
-102	Neither ACK nor NAK byte received.
-103	Modem status failure; not acknowledged because no frame sent.
-104	No answer from the modem, no byte received from FPMA.

www.BDTIC.com/ST

5 Commands

Table 4. Commands

Group	Command	Code	Related c-file
Synchronization	CMD_SynchroIndication CMD_DesynchroRequest CMD_SynchroStatus	10h 11h 85h	cmd_synchro.h
Reset	CMD_ResetRequest	21h	cmd_reset.h
MIB	CMD_WriteDBRequest CMD_WriteDBConfirm CMD_WriteDBError CMD_ReadDBRequest CMD_ReadDBConfirm CMD_ReadDBError	41h 42h 43h 90h 91h 92h	cmd_mib.h
Data	CMD_DataIndication CMD_DataRequest CMD_DataConfirm	50h 51h 52h	cmd_data.h
Alarm	CMD_AlarmRequest CMD_AlarmConfirm CMD_AlarmIndication	88h 89h 8Ah	cmd_alarm.h
Spy	SPY_No_SubframeIndication SPY_SubframeIndication SPY_SearchSynchroIndication SPY_SynchroFoundIndication SPY_No_AlarmIndication SPY_AlarmIndication	A0h B0h C0h D0h E0h F0h	cmd_spy.h
Error	CMD_SyntaxError	20h	cmd_error.h
Init	CMD_Init	78h	cmd_init.h

All CMD_snd_xxx are called by user.

All CMD_rcv_xxx are called by **int** CheckFPMAevents(**int** panel).

If CMD_rcv_xxx commands were called, this can be checked in corresponding global data structure field.

5.1 cmd_synchro.h

Global data structure exported from cmd_synchro.h

struct ST7570_Status__Synchro_type{

unsigned char Last_Snd_CMD_CODE;

unsigned char Last_Rcv_CMD_CODE;

unsigned char LastSndErrCode;

www.BDTIC.com/ST

```
unsigned char Last_STATUS;

unsigned char Last_SYNC;
unsigned long Last_S0;
unsigned long Last_N0;
unsigned long Last_S1;
unsigned long Last_N1;
unsigned char Last_PGA;
unsigned char Last_PHASE;

unsigned char Last_PAD;
unsigned short Last_SA;
unsigned short Last_DA;

unsigned char Last_CAUSE;
unsigned short Last_ADD1;
unsigned short Last_ADD2;
};
extern struct ST7570_Status__Synchro_type ST7570_Status__Synchro;
```

Table 5. ST7570_Status__Synchro structure member description

Field	Note
Last_Snd_CMD_CODE	Code of the last “Send” command that was called by user.
Last_Rcv_CMD_CODE	Code of the last “Received” command.
LastSndErrCode	Result of the last “Send” command. See *Table 1* for values(1).

1. See the ST7570/80 datasheet for the meaning of the remaining fields.

Commands:

Called by user: automatically updates: Last_Snd_CMD_CODE, LastSndErrCode

void CMD_snd_DesynchroRequest(**void**);

void CMD_snd_PHY_SynchroStatus(**void**);

Not called by user: automatically updates: Last_Rcv_CMD_CODE

www.BDTIC.com/ST

void CMD_rcv_SynchroIndication(**char** *LocalPortFrame);

- PHY: Last_S0, Last_N0, Last_S1, Last_N1, Last_PGA, Last_PHASE
- MAC: Last_SYNC according this value:
  - LM_SYNC_FOUND: Last_S0, Last_N0, Last_S1, Last_N1, Last_PGA, Last_PHASE
  - LM_SYNC_CONF: Last_SA, Last_DA
  - LM_SYNC_LOSS: Last_CAUSE, Last_ADD1, Last_ADD2

void CMD_rcv_PHY_SynchroStatus(**char** *LocalPortFrame);

- PHY: Last_STATUS

5.2 cmd_reset.h

Global data structure exported from cmd_reset.h

struct ST7570_Status__Reset_type{

unsigned char Last_Snd_CMD_CODE;

unsigned char Last_Rcv_CMD_CODE;

unsigned char LastSndErrCode;

unsigned char Last_RESET;

};

extern struct ST7570_Status__Reset_type ST7570_Status__Reset;

See *Table 1* and *Table 4* for structure fields and their values.

Commands:

Called by user: automatically updates: Last_Snd_CMD_CODE, LastSndErrCode

void CMD_snd_ResetRequest(char autoReconfiguration);

Not called by user: automatically updates: Last_Rcv_CMD_CODE, Last_RESET

void CMD_rcv_ResetRequest (**char** *LocalPortFrame);

5.3 cmd_mib.h

Global data structure exported from cmd_mib.h

struct ST7570_Status__MIB_type{

unsigned char Last_Snd_CMD_CODE;

unsigned char Last_Rcv_CMD_CODE;

unsigned char LastSndErrCode; //Error about success in sending of command

unsigned char Last_ERROR; //Error reported by modem by DBError message

unsigned short Last_snd_INDEX;

www.BDTIC.com/ST

```
 unsigned short Last_rcv_INDEX;
};
extern struct ST7570_Status__MIB_type ST7570_Status__MIB;
```

See *Table 1* and *Table 4* for structure fields and their values.

Commands:

Called by user: automatically updates: Last_Snd_CMD_CODE, LastSndErrCode, Last_snd_INDEX;

```
void CMD_snd_WriteDBRequest(unsigned short usiIndex, unsigned int *uiValue);
void CMD_snd_ReadDBRequest(unsigned short usiIndex);
```

Not called by user: automatically updates: Last_Rcv_CMD_CODE,

- Last_rcv_INDEX

```
void CMD_rcv_WriteDBConfirm(char *LocalPortFrame);
void CMD_rcv_ReadDBConfirm(char *LocalPortFrame);
```

- Last_ERROR

```
void CMD_rcv_WriteDBError(char *LocalPortFrame);
void CMD_rcv_ReadDBError(char *LocalPortFrame);
```

5.4 cmd_data.h

Global data structure exported from cmd_data.h

```
struct ST7570_Status__Data_type{
 unsigned char Last_Snd_CMD_CODE;
 unsigned char Last_Rcv_CMD_CODE;
 unsigned char LastSndErrCode;
 unsigned char Last_CONFIRM_CODE;

 unsigned char Last_P_SDU[38];
 unsigned short Last_ASK0;
 unsigned short Last_ASK1;
 unsigned short Last_FSK;
 unsigned long Last_SNR0;
 unsigned long Last_SNR1;

 unsigned char Last_CREDIT;
 unsigned long Last_ADDRESS;
```


www.BDTIC.com/ST

unsigned char Last_PAD;

unsigned char Last_M_SDU[242];

unsigned char Decoded_IC;

unsigned char Decoded_CC;

unsigned char Decoded_DC;

unsigned short LastReceivedMessageLength;

};

extern struct ST7570_Status__Data_type ST7570_Status__Data;

See *Table 1* and *Table 4* for structure fields and their values.

Commands:

Called by user: automatically updates: Last_Snd_CMD_CODE, LastSndErrCode

void CMD_snd_PHY_DataRequest(**char** *P_SDU, **int** P_SDU_Length);

void CMD_snd_MAC_DataRequest(**char** IC, **char** DC, **char** CC, **long** SA, long DA, **char** *M_SDU, **int** M_SDU_Length);

Not called by user: automatically updates: Last_Rcv_CMD_CODE

void CMD_rcv_DataIndication(**char** *LocalPortFrame);

- PHY: Last_ASK0, Last_ASK1, Last_FSK, Last_SNR0, Last_SNR1, Last_P_SDU
- MAC: Last_CREDIT, Last_ADDRESS, Last_PAD, Last_M_SDU, Decoded_IC, Decoded_CC, Decoded_DC

void CMD_rcv_DataConfirm(**char** *LocalPortFrame);

- Last_CONFIRM_CODE

5.5 cmd_alarm.h

Global data structure exported from cmd_alarm.h

struct ST7570_Status__Alarm_type{

unsigned char Last_Snd_CMD_CODE;

unsigned char Last_Rcv_CMD_CODE;

unsigned char LastSndErrCode;

www.BDTIC.com/ST

```
 unsigned long Last_CNT;
 unsigned long Last_S0;
 unsigned long Last_N0;
 unsigned long Last_S1;
 unsigned long Last_N1;
 unsigned char Last_PAD;

 unsigned char Last_CONFIRM_CODE;
};
extern struct ST7570_Status__Alarm_type ST7570_Status__Alarm;
```

See *Table 1* and *Table 4* for structure fields and their values.

Commands:

Called by user: automatically updates: Last_Snd_CMD_CODE, LastSndErrCode

void CMD_snd_AlarmRequest(**void**);

Not called by user: automatically updates: Last_Rcv_CMD_CODE

void CMD_rcv_AlarmConfirm(**char** *LocalPortFrame);

- Last_CONFIRM_CODE

void CMD_rcv_AlarmIndication(**char** *LocalPortFrame);

- Last_CNT, Last_S0, Last_N0, Last_S1, Last_PAD

5.6 cmd_spy.h

Global data structure exported from cmd_spy.h

```
struct ST7570_Status__Spy_type{
 unsigned char Last_Rcv_CMD_CODE;
 unsigned long Last_S0;
 unsigned long Last_N0;
 unsigned long Last_S1;
 unsigned long Last_N1;
 unsigned short Last_ASK0;
 unsigned short Last_ASK1;
 unsigned short Last_FSK;
 unsigned char Last_PGA;
 unsigned char Last_P_SDU[38];
 unsigned long Last_SNR0;
```


www.BDTIC.com/ST

```
 unsigned long Last_SNR1;
 unsigned long Last_AL_S0;
 unsigned long Last_AL_N0;
 unsigned long Last_AL_S1;
 unsigned long Last_AL_N1;
 unsigned long Last_ALARM;
};
extern struct ST7570_Status__Spy_type ST7570_Status__Spy;
```

See *Table 1* and *Table 4* for structure fields and their values.

Commands:

Called by user:

None.

Not called by user: automatically updates: Last_Rcv_CMD_CODE,

void SPY_rcv_PHY_No_SubframeIndication(**char** *LocalPortFrame);

- Last_S0, N0, S1, N1

void SPY_rcv_PHY_SubframeIndication(**char** *LocalPortFrame);

- Last_S0, N0, S1, N1, ASK0, ASK1, FSK, PGA, SNR0, SNR1

void SPY_rcv_PHY_SearchSynchroIndication(**char** *LocalPortFrame);

- None

void SPY_rcv_PHY_SynchroFoundIndication(**char** *LocalPortFrame);

- Last_S0, N0, S1, N1, ASK0, ASK1, FSK, PGA

void SPY_rcv_PHY_No_AlarmIndication(**char** *LocalPortFrame);

- Last_ AL_S0, AL_N0, AL_S1, AL_N1, AL_ALARM

void SPY_rcv_PHY_AlarmIndication(**char** *LocalPortFrame);

- Last_ AL_S0, AL_N0, AL_S1, AL_N1, AL_ALARM

5.7 cmd_error.h

Global data structure exported from cmd_error.h

```
struct ST7570_Status__Error_type{
 unsigned char Last_Rcv_CMD_CODE;
 unsigned char Last_ERROR;
};
extern struct ST7570_Status__Error_type ST7570_Status__Error;
```

See *Table 1* and *Table 4* for structure fields and their values.

www.BDTIC.com/ST

Commands:

Called by user:

None.

Not called by user: automatically updates: Last_Rcv_CMD_CODE, Last_ERROR

void CMD_rcv_SyntaxError(**char** *LocalPortFrame);

5.8 cmd_init.h

struct ST7570_Status__Init_type{

unsigned char Last_Snd_CMD_CODE;

unsigned char Last_Rcv_CMD_CODE;

unsigned char LastSndErrCode;

unsigned char Last_RESULT;

};

extern struct ST7570_Status__Init_type ST7570_Status__Init;

See *Table 1* and *Table 4* for structure fields and their values.

#define init_OK 0

Commands:

Called by user: automatically updates: Last_Snd_CMD_CODE, LastSndErrCode

void CMD_snd_Init(**char** Step1_or_Step2);

Not called by user: automatically updates: Last_Rcv_CMD_CODE, Last_RESULT

void CMD_rcv_Init(char *LocalPortFrame);

www.BDTIC.com/ST

6 Library support files

6.1 time_counter.h

This file implements the timers being used for precise timing when communicating with the modem or can be used by the user for timing events in their own application. User can use up to six timers numbered from 0 to 5:

```
typedef enum{
 TimerCounter0 = 0,
 TimerCounter1 = 1,
 TimerCounter2 = 2,
 TimerCounter3 = 3,
 TimerCounter4 = 4,
 TimerCounter5 = 5,
} TimerNumberEnum;

//Results
#define TimerElapsed 1
#define TimerNotElapsed 0
#define TimerDisabled 2000000000

//Time in milliseconds
#define SHORTTIME 50
#define ANSWERTIME7590 50
#define TIMERSMAXCOUNT 6
```

void TIMER_timeToElapse(**TimerNumberEnum** TimerNo, **int** ms);

- set the timer TimerNo to ms milliseconds, enables it and starts countdown.

int TIMER_timeElapsed(**TimerNumberEnum** TimerNo);

- if the timer TimerNo reaches zero, this function returns TimerElapsed value
- if the timer is disabled, this function returns TimerElapsed value
- if the timer has not reached zero yet, this function returns TimerNotElapsed

void TIMER_DisableTimer(**TimerNumberEnum** TimerNo);

- disables the timer TimerNo

void TIMER_waitFor(**TimerNumberEnum** TimerNo, **int** ms);

- finishes after ms milliseconds using the timer TimerNo.

www.BDTIC.com/ST

void TIMER_HW_Init(**void**);

- configures the timer or SysTick hw used for timing.

6.2 history.h

This file implements a simple circular data logging system. It logs 4 values: ErrorCode, StatusCode, CommandCodeSnd, and CommandCodeRcv whenever putIn function is called. Array of the logged bytes, ErrorEvidence, can be investigated in the programming environment during debugging. Array capacity is 4 x 255 bytes.

```
#define maxerrors 255
unsigned char pointer;

struct{
 unsigned char Error;
 unsigned short Status;
 unsigned char CommandSnd;
 unsigned char CommandRcv;
} ErrorEvidence[maxerrors];
```

void initErrorList(**void**);

- inits the data logger.

void putIn(**unsigned char** ErrorCode, **unsigned short** StatusCode, **unsigned char** CommandCodeSnd, **unsigned char** CommandCodeRcv);

- inserts the new values in to the ErrorEvidence array.

6.3 serial_port.h

This file contains functions that take care of incoming packets which were received by the power line modem. According to received packet, the main function calls corresponding CMD_rcv_xxx functions from other files from *Section 5*.

Function exported from serial_port.h

Called by user:

int CheckFPMAevents(**int** panel);

This function must be called by the user regularly in order to be able to receive any incoming message. This function calls CMD_rcv_xxx commands according to the data received by the modem over power line.

www.BDTIC.com/ST

Table 6. CheckFPMAevents function parameters description

Parameter	Meaning
panel	The parameter is present only for legacy reasons; its value is not used.

6.4 stm32_uart.h

This file implements a buffered UART interface.

The beginning of the file is dedicated to the physical definition of the pins of the microcontroller used for UART interface:

#define TXD_RXD_remap 1 // 1: Remap (TX/PB6, RX/PB7)

#define TXD_pin GPIO_Pin_6 //used UART

#define TXD_port GPIOB //used UART

#define RXD_pin GPIO_Pin_7 //used UART

#define RXD_port GPIOB //used UART

#define TREQ_pin GPIO_Pin_9 //T_REQ

#define TREQ_port GPIOB //T_REQ

Functions for buffered UART (parameter portNumber is present for legacy reasons only, has no influence on functionality):

void UART_init(**void**);

- configures UART interface of the used microcontroller.

int GetInQLen (**int** portNumber);

- gives the length of the data present in the input buffer for the UART.

int FlushOutQ (**int** portNumber);

- clears the output buffer for the UART.

int FlushInQ (**int** portNumber);

- clears the input buffer for the UART.

int ComSetEscape (**int** portNumber, **int** escapeCode);

- sets T_REQ signal to logical value according to escapeCode: SETRTS: 0 V, CLRRTS: V_{CC}.

int ComWrtByte (**int** portNumber, **char** byte);

- writes one byte to the buffered UART.

int ComWrt (**int** portNumber, **char** buffer[], size_t count);

- writes array buffer of size_t length to buffered UART.

www.BDTIC.com/ST

int ComRd (**int** portNumber, **char** buffer[], **int** count);

- reads data from buffered UART to array buffer. Count indicates number of bytes read.

char ComRdByte (**int** portNumber);

- reads one byte from buffered UART.

www.BDTIC.com/ST

7 Communication example

This communication example demonstrates two nodes (Client and Server node). The first node sends a packet every two seconds and an LED blinks for 300 ms when modem confirms the data was sent. As soon as the second node receives the data, an LED also blinks for 300 ms.

This example application uses the ST7570 library described in the sections above and two files: main.c and fucntions.c.

7.1 main.c

This file contains only the state machine that calls the corresponding function of each state from function.c file:

```
ActualStateEnum ActualState = MODEM_CHECK;

 while(1){
   switch(ActualState){
   case MODEM_CHECK: ActualState = modem_state_check();      break;

   case CLIENT_MASTER_DATA_SEND_req:       ActualState =
client_master_data_send_req();      break;
   case CLIENT_MASTER_DATA_CONFIRM_ind:    ActualState =
client_master_data_confirm_ind(); break;
   case CLIENT_MASTER_DATA_RECEIVE_ind:    ActualState =
client_master_data_receive_ind();     break;

   case SERVER_SLAVE_DATA_SEND_req:       ActualState =
server_slave_data_send_req();     break;
   case SERVER_SLAVE_DATA_RECEIVE_ind:    ActualState =
server_slave_data_receive_ind();     break;

   case IDLE_STATE:;    break;
   }
   ActualState = check_external_events(ActualState); //check for incoming packet
 }
```


www.BDTIC.com/ST

7.2 functions.c

This file contains the corresponding functions for each state of the state machine implemented in main.c:

```
ActualStateEnum modem_state_check(void){
 while(GetFPMAstatus(NO_EFFECT_HARD_WIRED_UART1_REMAPED, 0, NULL, 0,
NULL)! = 0){
 }
 InitDevice();
 if(Demonstration_data.nodeIdentification == DEVICE_CLIENT_MASTER_NODE)
 return CLIENT_MASTER_DATA_SEND_req;
else
 return SERVER_SLAVE_DATA_SEND_req;
}

//---------------------------- DEVICE_CLIENT_MASTER_NODE ----------------------
ActualStateEnum client_master_data_send_req(void){
 ST7570_Status__Data.Last_CONFIRM_CODE = LP_NOT_VALID;
 CMD_snd_PHY_DataRequest(&(Demonstration_data.dataToSend), 1);
 TIMER_timeToElapse(TimerCounter3, 700); //Confirm response interval is 290 - 560 ms
 return CLIENT_MASTER_DATA_CONFIRM_ind;
}

ActualStateEnum client_master_data_receive_ind(void){
 return IDLE_STATE;
}

ActualStateEnum client_master_data_confirm_ind(void){
 if(TIMER_timeElapsed(TimerCounter3))
     return IDLE_STATE;
   switch(ST7570_Status__Data.Last_CONFIRM_CODE){
   case LP_NOT_VALID:
     return CLIENT_MASTER_DATA_CONFIRM_ind;
   case LP_OK:
     if(Demonstration_data.dataToSend == 0x11)
```


www.BDTIC.com/ST

```
        GPIO_ResetBits(ORANGE_LED1_Port, ORANGE_LED1_Pin);
      else
        GPIO_ResetBits(RED_LED_Port, RED_LED_Pin);
      TIMER_timeToElapse(TimerCounter2, 300);
    default:
      return IDLE_STATE;
  }
}

//---------------------------- DEVICE_SERVICE_SLAVE_NODE ----------------------
ActualStateEnum server_slave_data_send_req(void){
 return IDLE_STATE;
}

ActualStateEnum server_slave_data_receive_ind(void){
 if(ST7570_Status__Data.Last_P_SDU[0] == 0x11)
    GPIO_ResetBits(ORANGE_LED1_Port, ORANGE_LED1_Pin);
 else
    GPIO_ResetBits(RED_LED_Port, RED_LED_Pin);
   return IDLE_STATE;
}

//====================== END - Exchange data
==================================

ActualStateEnum check_external_events(ActualStateEnum InState){
 int incomming_packet_COMMAND_ID;
 ActualStateEnum returnValue = InState;

 if(Demonstration_data.nodeIdentification == DEVICE_CLIENT_MASTER_NODE)
    if(TIMER_timeElapsed(TimerCounter1))
 {
    TIMER_timeToElapse(TimerCounter1, 2000); //timer to send message every 2 s
    returnValue = CLIENT_MASTER_DATA_SEND_req;
 }
```


www.BDTIC.com/ST

```
 incomming_packet_COMMAND_ID = check_incomming_packets();

 if(incomming_packet_COMMAND_ID == CMD_DATA_INDICATION_CODE){
    TIMER_timeToElapse(TimerCounter2, 300);    //timer to switch on the LED
    returnValue = SERVER_SLAVE_DATA_RECEIVE_ind;
 }

 if(incomming_packet_COMMAND_ID == CMD_SYNCHRO_INDICATION_CODE)
    TIMER_timeToElapse(TimerCounter4, 5000);

 if(TIMER_timeElapsed(TimerCounter4)){   //automatic desynchro request every 5 s
    CMD_snd_DesynchroRequest();
    TIMER_DisableTimer(TimerCounter4);
 }

 if(TIMER_timeElapsed(TimerCounter2)){    //timer to switch off the LEDs
    GPIO_SetBits(ORANGE_LED1_Port, ORANGE_LED1_Pin);
    GPIO_SetBits(RED_LED_Port, RED_LED_Pin);
    TIMER_DisableTimer(TimerCounter2);
 }

 return returnValue;
}

int check_incomming_packets(void){
 if(RX_buffer_internal_not_empty){
    return CheckFPMAevents(0);
    }
 return 0;
}
```


www.BDTIC.com/ST

8 Real application

Figure 1 shows an application consisting of the ST7570 power line board and STM32 demonstration board. This setup represents the complete node. More details about interconnection of different platforms to power line demonstration boards can be found in the UM1038 user manual.

Figure 1. ST7570 and STEVAL-PCC012V1, block diagram

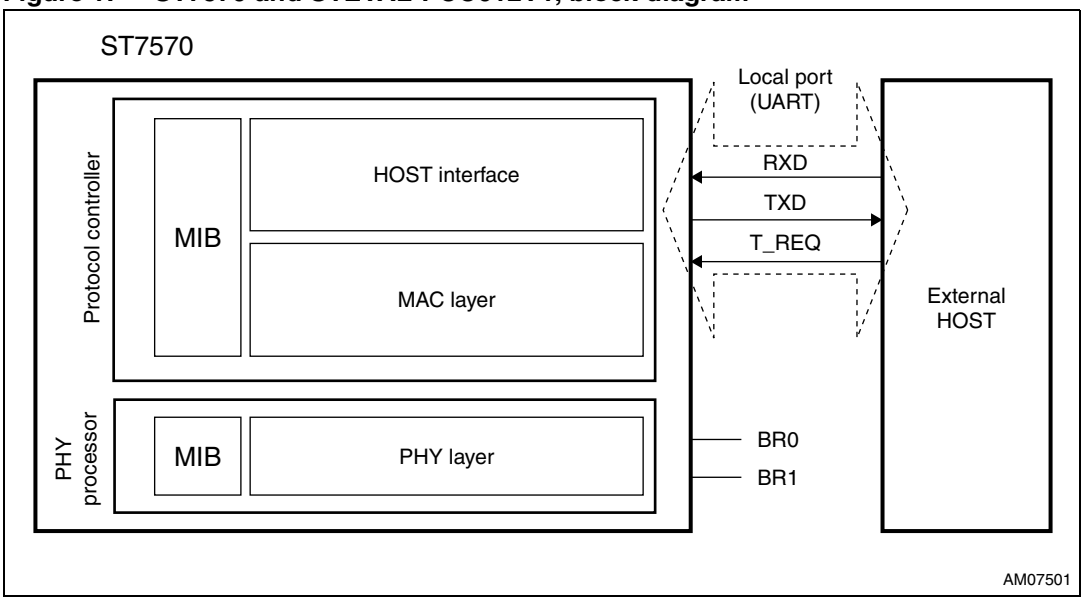
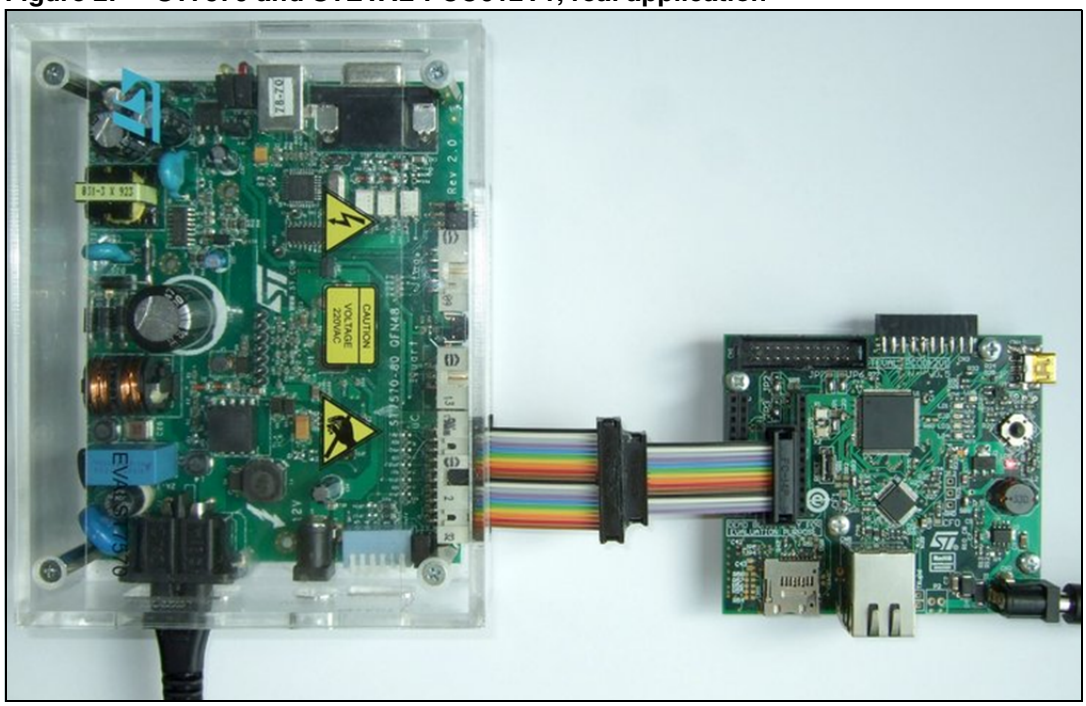


Figure 2. ST7570 and STEVAL-PCC012V1, real application



www.BDTIC.com/ST

9 Revision history

Table 7. Document revision history

Date	Revision	Changes
20-Apr-2011	1	Initial release.

www.BDTIC.com/ST

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2011 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com

www.BDTIC.com/ST