AN3311
Application note

In-home display for ZigBee® smartplug

Introduction

This application note describes the demonstration firmware running on the STM3210C-EVAL for the STM32F107VC to manage a smartplug network system.

An embedded graphical user interface (GUI) based on the “multi-input embedded GUI library 2.0 for STM32F10xxx” described in the AN3128 application note, and working on an LCD TFT 320 x 240 display and 5-position joystick, allows the user to interact with the smartplug system made up of one coordinator and two smartplugs connected.

Section 1 describes the document and library rules.

Section 2 highlights the features of the ZigBee smartplug and explains its hardware interface with a device microcontroller (STM32).

Section 3 describes briefly the “multi-input embedded GUI library”.

Section 4 describes the relevant blocks of the STM3210C-EVAL demonstration board.

Section 5 shows the demonstration firmware/board system setup.

Section 6 describes, in detail, how the “in-home display” firmware is structured, its architecture and its exported APIs.

Section 7 explains how to get started with the system, how to configure and use the IAR workspace, and contains an example application source code.

Section 8 illustrates how the “in-home display” GUI application works.

Section 9 illustrates the hardware schematics.

Contents

List of tables

List of figures

1 Document and library rules

This document uses the conventions described in the sections below.

1.1 Acronyms

Table 1 lists the acronyms used in this document.

Table 1. List of acronyms

Acronym	Meaning
API	Application programming interface
HAL	Hardware abstraction layer
MCU	Microcontroller unit
I^2C	Inter-integrated circuit
SPI	Serial to parallel interface
OOP	Object oriented programming

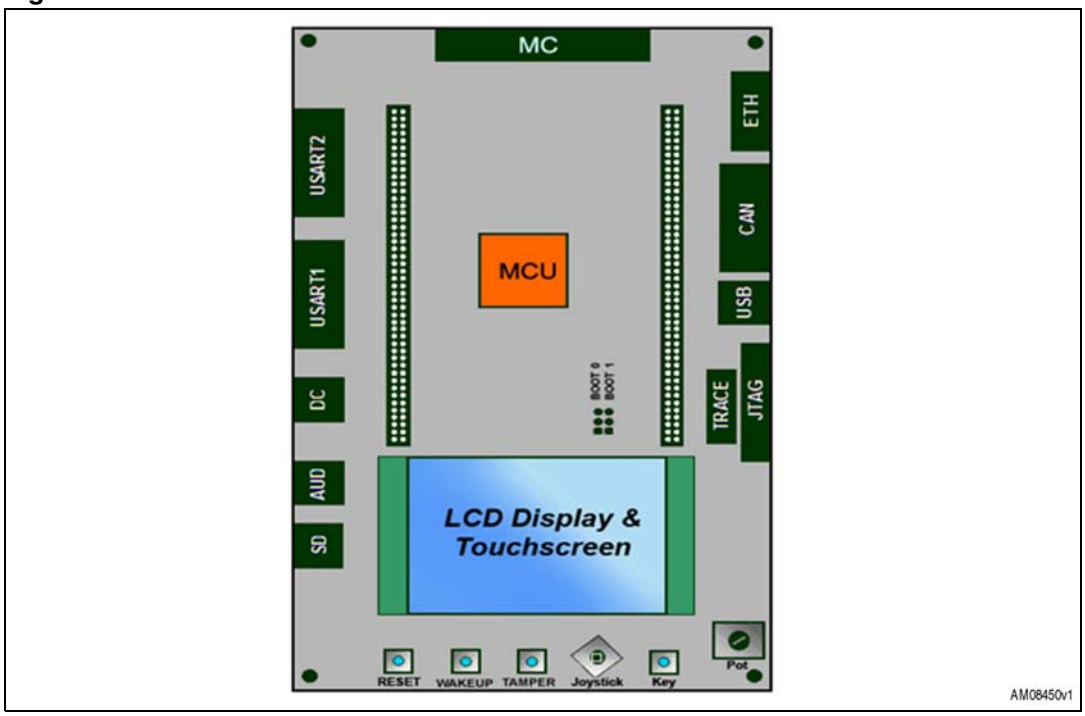
2 ZigBee smartplug

2.1 Smartplug description

The smartplug coordinator is connected via an SPI to the STM3210C-EVAL through an “Ad-Hoc” adapter.

In *Figure 1* it is possible to take a quick look at the STM3210C-EVAL board and a smartplug node block diagram.

Figure 1. STM3210C-EVAL block scheme

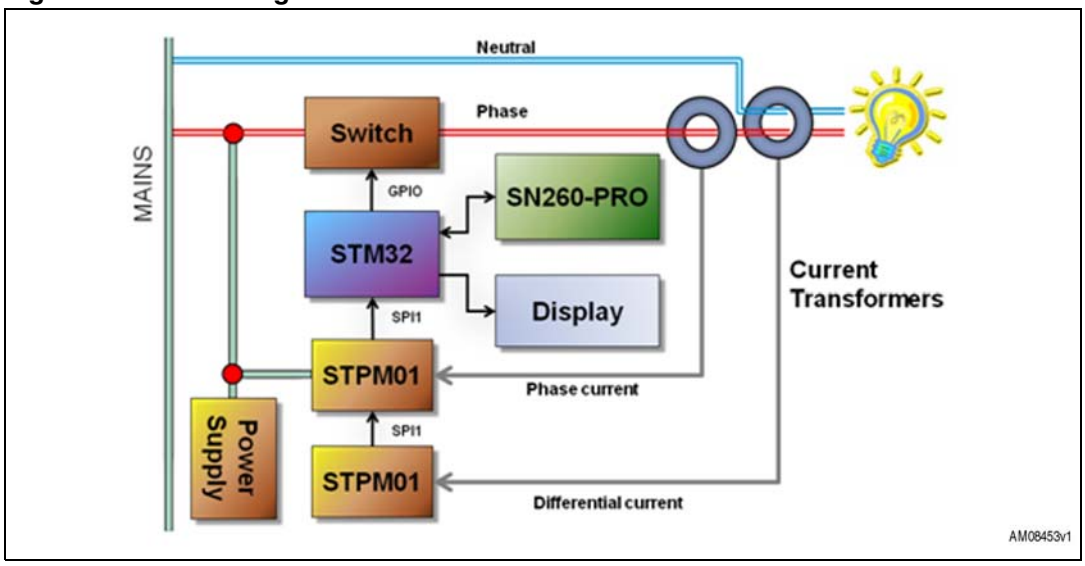


An adapter is connected to the extended connectors CN8 and CN9 on the STM3210C-EVAL (for more detailed information please refer to the UM0600 user manual), it allows the connection of a ZigBee smartplug coordinator and the I^2C/RF dual interface EEPROM M24LR64-r. The Gerber files of the adapter board are included in the setup package of this project.

The ZigBee smartplug board can be used as a guide to build a home/building automation subsystem for energy management. In a typical application, the board is plugged into an electrical wall socket and supplies an electrical load, monitoring the energy consumption; using several smartplugs it is possible to monitor and control the home/building energy consumption socket by socket. The board includes the following functions, shown in the block diagram of *Figure 2*:

- Energy measurement
- Load differential current
- Load driving by relay or TRIAC (dimming)
- ZigBee communication capability.

Figure 2. Block diagram



The STEVAL-IHP001V3 is a smartplug board based on an STM32F10x microcontroller, a SPZB260 ZigBee module, and an STPM01 energy metering IC.

It implements a ZigBee metering node which allows the final user to monitor and manage energy consumption.

The board has been developed as a guide to build a home/building automation subsystem for energy management. In a typical home system implementation, the board is plugged into an electrical wall socket and supplies a home appliance or other generic electrical load.

The current, power, energy, and other information, related to the electrical load connected to the smartplug board, can be displayed locally on an LCD screen, and are sent to a ZigBee data concentrator through the home/building ZigBee network.

2.2 ZigBee module

ZigBee smartplug communication is based on the SPZB260 module with a DIL adapter. The module is FCC compliant (FCC ID:S9NZB260A). The module is based on the SN260 ZigBee network processor which integrates a 2.4 GHz, IEEE 802.15.4 compliant transceiver, as well as IEEE 802.15.4 PHY and MAC. The main features are:

- 0dBm nominal TX output power
- -92dBm RX sensitivity
- +2dBm TX output power in boost mode
- RX filtering for co-existence with IEEE 802.11g and Bluetooth® devices.

For further details please refer to the SPZB260 module and the SN260 network processor datasheet.

Note: For more information, see the UM0608 user manual, STEVAL-IHP001V3 schematics diagram, and AN2993 application note.

3 Multi-input embedded GUI library

3.1 Description

This solution enables users, comfortable with the use of standard microcontrollers, to create higher-end “look and feel” human interfaces by replacing conventional electromechanical switches with touch-sensing controls.

Users can combine touch-sensing functions using multiple configurations (touchscreen, joystick, and keys) with traditional MCU features (communication, beeper, LCD control, etc.).

The E-multi-input graphic library is part of the application firmware.

The graphic objects are a set of controls that can be printed on the screen and associated to an action when pressed.

The library has been developed and tested on an LCD panel of QWGA resolution (320x240) which is the default, but the library is independent of the LCD resolution, although it has not been tested with others.

The library supports touchscreen features and includes a low level driver which handles the analog input (for 12-bit ADC), and a function for the touchscreen calibration based on an algorithm that uses 5 points.

The multi-input embedded GUI firmware library is fully developed in 'ANSI-C' following an OOP approach. This means that the final application uses instances of page and graphic objects according to their public methods and properties. In the end, the PageObj is a structure containing public properties (data fields) and methods (functions pointers). The OOP encapsulation feature is assured. The library has been developed and tested on the “STM3210C” STMicroelectronics demonstration board.

The library can be included in the final application as a library file (multi-input embedded GUI library.a) and used as a black box through its exported public API, or can be included in the final application as source files (.c and .h), if the user wants to debug the library itself, or to change the HAL functions in order to port the library on a different LCD (in model and resolution) from the one attached to STM3210C-EVAL.

For more information on the graphic library see the AN3128, rev. 2, application note.

The calibration process is part of the post-processing layer. The touchscreen must be calibrated at first power-on and/or upon user request.

Once the calibration is done, final adjustments on future power-on of the board are not necessary because the calibration parameters are saved on the Flash memory.

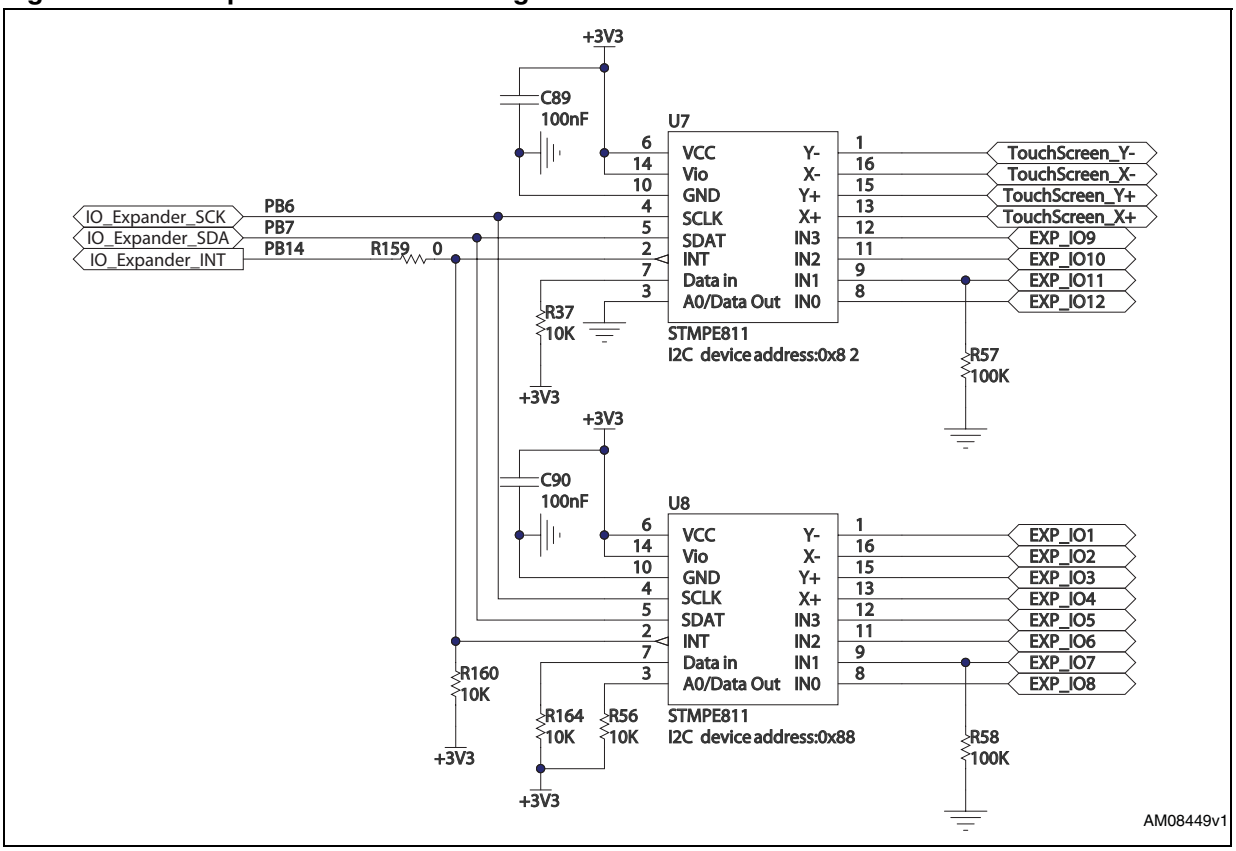
The touchscreen and the joystick are controlled by the STMPE811 devices.

The STMPE811 has a simple 2-wire I^2C digital serial interface which allows the user to access the data in the touchscreen controller register at any time. It communicates via the serial interface with a master controller.

Figure 3 shows how the STM32F10xxx microcontroller (master device) must be connected to the STMPE811 device.

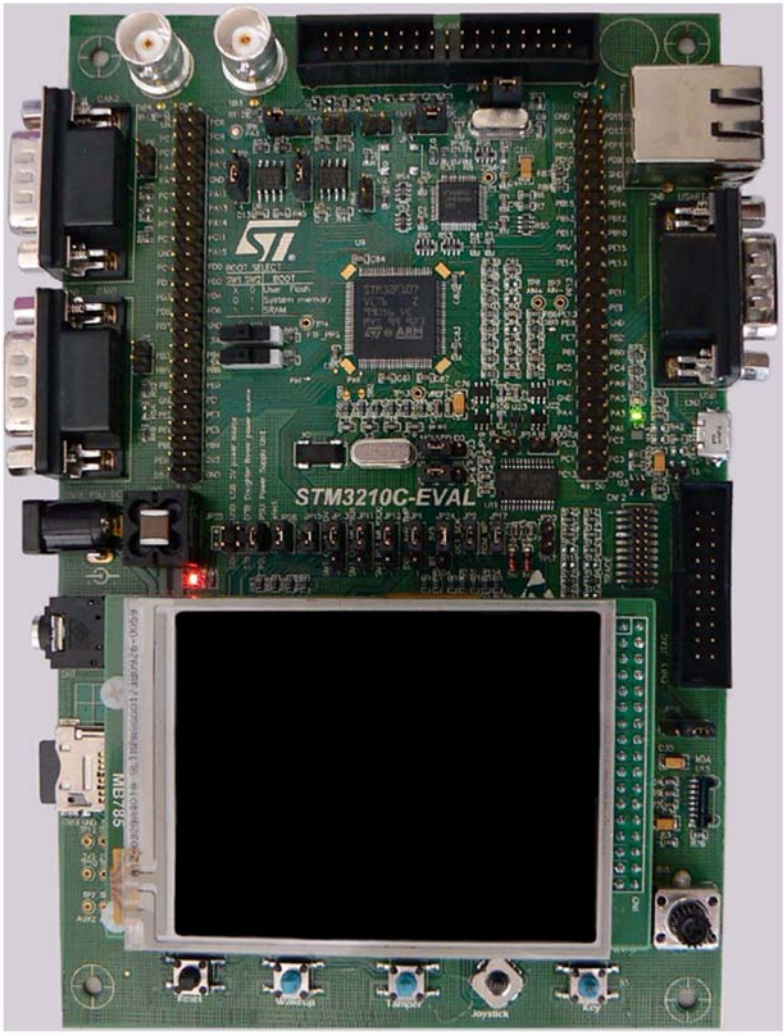
Refer to the STMPE811 datasheet for more information on the register concerning the data of the touched points on the touchscreen.

Figure 3. I/O expander hardware configuration on the STM3210C-EVAL



4 STM3210C-EVAL demonstration board

Figure 4. STM3210C-EVAL demonstration board



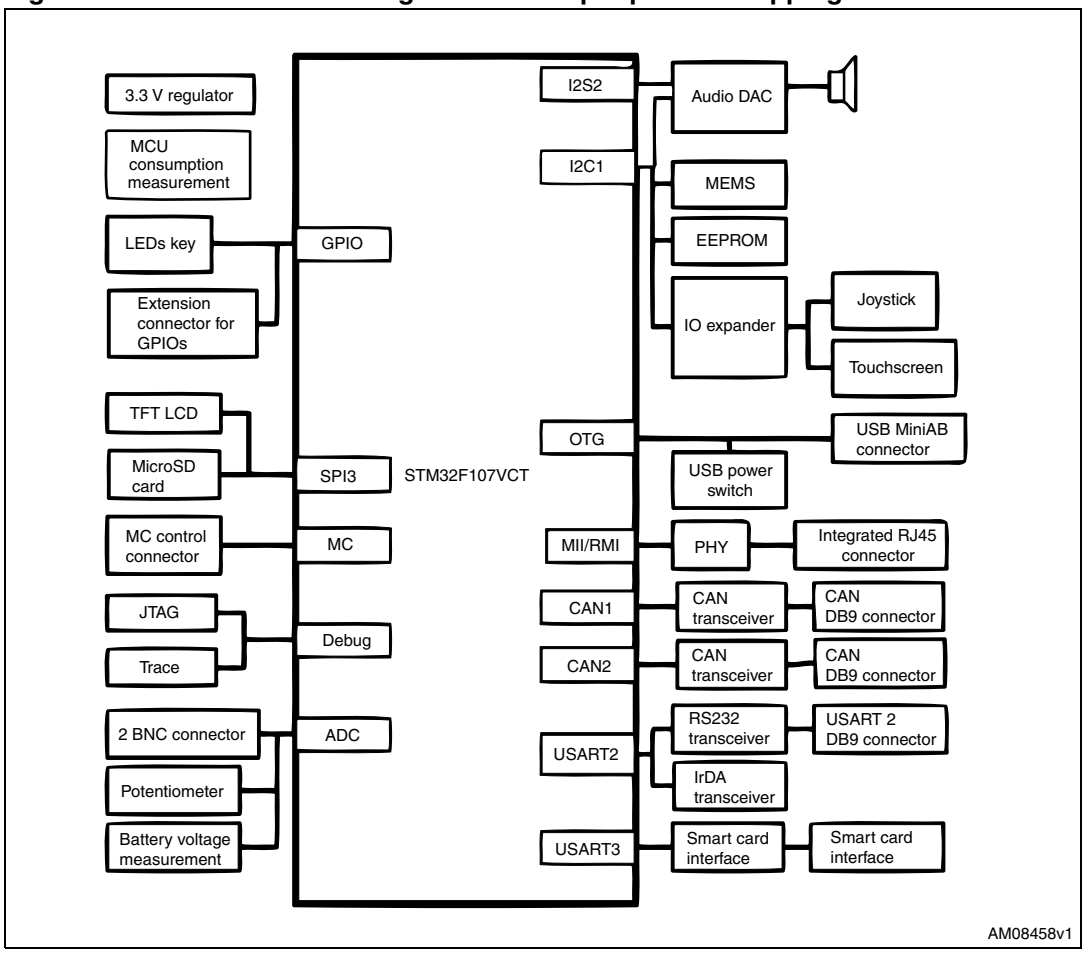
4.1 Features

- Three 5 V power supply options: power jack, USB connector, or daughterboard
- Boot from user Flash, system memory or SRAM
- I2S audio DAC, stereo audio jack
- 512 MByte (or bigger) micro-SD cardTM
- Both type A and B smartcard support
- I^2C compatible serial interface 64-Kbit EEPROM, MEMS and I/O expander
- RS-232 communication
- IrDA transceiver
- USB-OTG full speed, USB mini-AB connector
- IEEE-802.3-2002 compliant Ethernet connector
- Two channels of CAN2.0A/B compliant connection
- Inductor motor control connector
- JTAG and trace debug support
- 3.2" 240x320 TFT color LCD with touchscreen
- Joystick with 4-direction control and selector
- Reset, wake-up, tamper, and user button
- 4 color LEDs
- RTC with backup battery
- MCU consumption measurement circuit
- Extension connector for daughterboard or wrapping board.

4.2 STM32 peripherals mapping

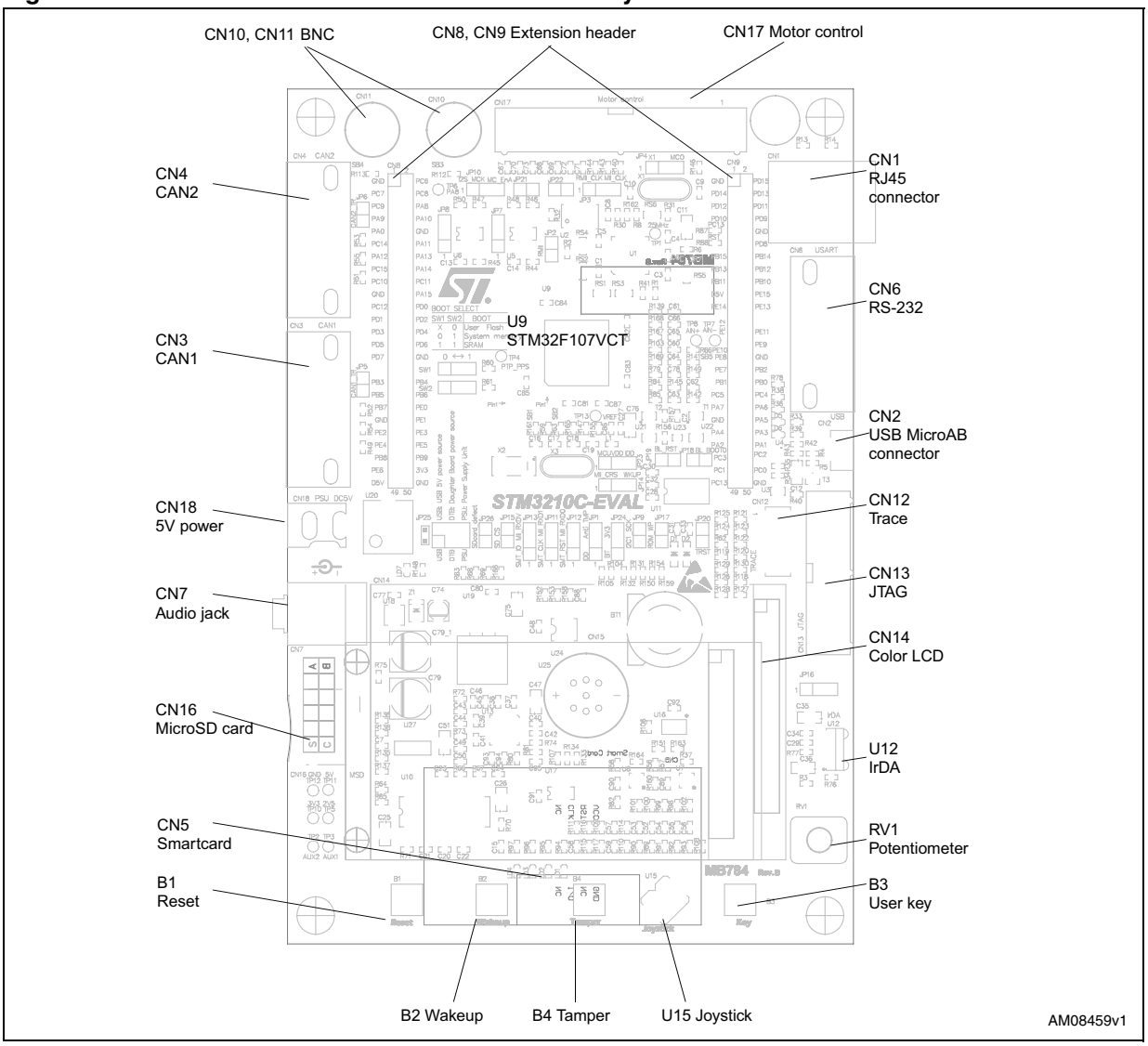
The STM3210C-EVAL demonstration board is designed around the STM32F107VC in a 100-pin TQFP package. The hardware block diagram, *Figure 5*, illustrates the connection between the STM32F107VC and peripherals (LCD, EEPROM, MEMS, USART, IrDA, USB-OTG, Ethernet, audio, CAN bus, smartcard, micro-SD card, and motor control) and these features can be located on the actual demonstration board in *Figure 6*.

Figure 5. Hardware block diagram - STM32 peripherals mapping



For more details of calibration parameters, refer to the STPM01 datasheet on www.st.com.

Figure 6. STM3210C-EVAL demonstration board layout



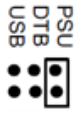
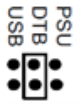
4.3 Power supply

The STM3210C-EVAL demonstration board is designed to be powered by a 5 V DC power supply and protected by PolyZen from a wrong power plug-in event. It is possible to configure the demonstration board to use any of the following three sources for the power supply:

- 5 V DC power adapter connected to CN18, the power jack on the board (PSU on silk screen for power supply unit)
- 5 V DC power with 500 mA limitation from CN2, the USB mini-AB connector (USB on silkscreen)
- 5 V DC power from both CN8 and CN9, the extension connector for the daughterboard (DTB for daughterboard on silkscreen).

The power supply is configured by setting the related jumpers JP24 and JP25, as described in *Table 2*.

Table 2. Power related jumpers

Jumper	Description	Configuration
JP25	JP25 selects one of the three possible power supply resources. For power supply jack (CN18) to the STM3210C-EVAL only, JP25 is set as shown: (Default)	
	For power supply from the daughterboard connectors (CN8 and CN9) to the STM3210C-EVAL only, JP25 is set as shown:	
	For power supply from USB (CN2) to the STM3210C-EVAL only, JP25 is set as shown:	
	For power supply from power supply jack (CN18) to both the STM3210CEVAL and daughterboard connected on CN8 and CN9, JP25 is set as shown to the right (the daughterboard must not have its own power supply connected):	
JP24	V_{bat} is connected to 3.3 V power when JP24 is set as shown: (Default)	
	V_{bat} is connected to battery when JP24 is set as shown:	

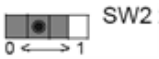
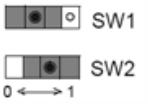
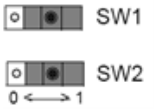
4.4 Boot option

The STM3210C-EVAL board is able to boot from:

- Embedded user Flash
- System memory with boot loader for ISP
- Embedded SRAM for debugging.

The boot option is configured by setting switches SW1 (BOOT1) and SW2 (BOOT0). The BOOT0 can be configured also via the RS-232 connector CN6.

Table 3. Boot related switches

Switch	Boot from	Configuration
SW1 and SW2	STM3210C-EVAL boots from user Flash when SW2 is set as shown on the right. SW1 setting does not matter in this configuration. (Default)	
	STM3210C-EVAL boots from system memory when SW1 and SW2 are set as shown:	
	STM3210C-EVAL boots from embedded SRAM when SW1 and SW2 are set as shown:	

4.5 Clock source

Two clock sources are available on the STM3210C-EVAL demonstration board for STM32F107VC, and RTC is embedded.

- X2, 32 kHz crystal for embedded RTC
- X3, 25 MHz crystal with socket for an STM32F107VC microcontroller, it can be removed from the socket when an internal RC clock is used.

4.6 Reset source

The reset signal of the STM3210C-EVAL board is low active and the reset sources include:

- Reset button, B1
- Debugging tools from JTAG connector CN13 and trace connector CN12
- Daughterboard from CN9
- RS-232 connector CN6 for ISP.

Table 4. Reset related jumper

Jumper	Description
JP20	Enables reset of the STM32F107VC embedded JTAG TAP controller each time a system reset occurs. JP20 connects the TRST signal from the JTAG connection with the system reset signal RESET#. Default setting: **not fitted**.

4.7 Joystick

The joystick is four-directional and includes a selection key.

4.8 Pushbuttons

The following pushbuttons are provided:

- Key: user pushbutton
- Tamper: user pushbutton
- Wake-up: pushbutton used to wake up the processor from low-power mode.

4.9 Storage memory

The ZigBee adapter has a 64-Kbit I^2C/RF dual interface memory (M24LR64-r) on board, and it is connected to the I^2C1 peripheral of the MCU.

The I^2C address of the memory can be set by using the jumpers JP1 and JP2 on the ZigBee adapter and must be different from the one already present on the STM3210C-EVAL (with address 0xA0).

4.10 Development and debug support

The two debug connectors available on the STM3210C-EVAL demonstration board are:

1. CN13, standard 20-pin JTAG interface connector which is compliant with the debug tools of ARM7 and ARM9
2. CN12, SAMTEC 20-pin connector FTSH-110-01-L-DV for both SWD and trace which is compliant with ARM CoreSight™ debug tools

4.11 Display and input devices

The 3.2" TFT color LCD connected to SPI3 and 4 general purpose color LED's (LED 1, 2, 3, 4) are available as display devices.

A touchscreen connected to an I/O expander (U7), a 4-direction joystick with selection key, a general purpose button (B3), a wake-up button (B2), and a tamper detection button (B4) are available as input devices.

Table 5. LCD module

3.2" TFT LCD with touchscreen CN14 (default)		
Pin on CN14	**Description**	**Pin connection**
1	CS	PB2
2	RS	-
3	WR/SCL	PC10
4	RD	-

Table 5. LCD module (continued)

3.2" TFT LCD with touchscreen CN14 (default)		
Pin on CN14	**Description**	**Pin connection**
5	RESET	RESET#
22	BL_GND	GND
23	BL_Control	+5 V
24	VDD	3.3 V
25	VCI	3.3 V
26	GND	GND
27	GND	GND
28	BL_VDD	+5 V
29	SDO	PC11
30	SDI	PC12
31	XL	I/O Expander
32	XR	I/O Expander
33	YD	I/O Expander
34	YU	I/O Expander

4.12 JTAG debugging connector CN13

Figure 7. JTAG debugging connector CN13 viewed from above the PCB

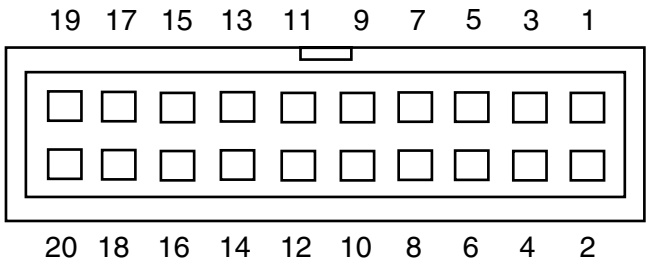


Table 6. JTAG debugging connector CN13

Pin number	Description	Pin number	Description
1	3.3 V power	2	3.3 V power
3	PB4	4	GND
5	PA15	6	GND
7	PA13	8	GND
9	PA14	10	GND

Table 6. JTAG debugging connector CN13 (continued)

Pin number	Description	Pin number	Description
11	RTCK	12	GND
13	PB3	14	GND
15	RESET#	16	GND
17	DBGRQ	18	GND
19	DBGACK	20	GND

4.13 Daughterboard extension connector CN8 and CN9

Two 50-pin male headers, CN8 and CN9, can be used to connect a daughterboard or standard wrapping board to the STM3210C-EVAL demonstration board. All 80 GPIOs are available on it. The space between these two connectors and the power position, GND and RESET pin are defined as a standard, which allows to develop common daughterboards for several demonstration boards. The standard width between CN8 pin1 and CN9 pin1 is 2700 mills (68.58mm). This standard was implemented on the majority of demonstration boards.

Each pin on CN8 and CN9 can be used by a daughterboard after disconnecting it from the corresponding function block on the STM3210C-EVAL demonstration board.

Table 7. Daughterboard extension connector CN8

Pin	Description	Alternate function	How to disconnect with function block on STM3210C-EVAL board
1	GND	-	
3	PC7	MC	Disconnect STM3210C-EVAL board from motor power drive board
5	PC9	USB power switch On	Remove R36
7	PA9	USB VBUS	Remove R78
9	PA0	MC/Ethernet/WKUP	Keep JP14 open. Disconnect STM3210C-EVAL board from motor power drive board.
11	PC14 via SB1	32 kHz oscillator	Remove R161, close SB1
13	PA12	USB_DP	Remove R43 or disconnect USB cable
15	PC15 via SB2	32 kHz oscillator	Remove R59, close SB2
17	PC10	SPI1_CLK	
19	GND	-	
21	PC12	SPI1_MOSI	
23	PD1	CAN1_TX	
25	PD3	LD3	Remove R96
27	PD5	USART2_TX	
29	PD7	LD1	Remove R94
31			

Table 7. Daughterboard extension connector CN8 (continued)

Pin	Description	Alternate function	How to disconnect with function block on STM3210C-EVAL board
33	PB3	TDO/SWO	
35	PB5	CAN2_RX	Remove R45
37	PB7	I2C1_SDA	Remove R132
39	GND	-	
41	PE2	Trace_CK	
43	PE4	Trace_D1	
45	PB8	MC	Disconnect STM3210C-EVAL board from motor power drive board
47	PE6	Trace_D3	
49	D5V		
2	PC6	I2S_MCK/MC	JP10 open
4	PC8	MC	Disconnect STM3210C-EVAL board from motor power drive board
6	PA8	MCO	JP4 open
8	PA10	USB_ID	Remove R38 or disconnect USB cable
10	GND		
12	PA11	USB_DM	Remove R42 or disconnect USB cable
14	PA13	TMS/SWDIO	
16	PA14	TCK/SWCLK	
18	PC11	SPI1_MISO	Remove R135 and LCD
20	PA15	TDI	
22	PD0	CAN1_RX	Remove R44
24	PD2	MC	Disconnect STM3210C-EVAL board from motor power drive board.
26	PD4	LD4	Remove R97
28	PD6	USART2_RX	Keep JP16 open
30	GND		
32			
34	PB4	TRST	Keep JP20 open
36	PB6	CAN2_TX/ I2C1_SCK	Keep JP9 open
38	PE0	MC/Micro-SD card detection	Remove micro-SD card. Disconnect STM3210C-EVAL board from motor power drive board.
40	PE1	USB_Overcurrent	Remove R35
42	PE3	Trace_D0	
44	PE5	Trace_D2	

Table 7. Daughterboard extension connector CN8 (continued)

Pin	Description	Alternate function	How to disconnect with function block on STM3210C-EVAL board
46	PB9	User button	Remove R104
48	3.3 V		
50	GND		

Table 8. Daughterboard extension connector CN9

Pin	Description	Alternate function	How to disconnect with function block on STM3210C-EVAL board
1	GND	-	
3	PD14	MC	Keep JP22 open. Disconnect STM3210C-EVAL board from motor power drive board.
5	PD12	Ethernet	Remove RS2
7	PD10	Ethernet/smartcard	Keep JP11 open
9	PC13 button B3	IDD_CNT_EN / Anti-tamper button B4	Keep JP1 open
11	RESET#	-	
13	PB15	I2S_DIN	
15	PB13	I2S_CK/Ethernet	Remove RS3
17	PB11	Ethernet	Remove RS3
19	D5V	-	
21	PE14	Smartcard_CMDVCC	Remove R166
23			
25	PE12	MC	Disconnect STM3210C-EVAL board from motor power drive board
27	PE10	MC	
29	PE8	MC	
31	PE7	Smartcard_OFF	Remove R69
33	PB1	MC	Remove R168
35	PC5	VBAT_voltage	Remove R154
37	PA7	MC	Remove R169
39	GND		
41	PA4	Micro-SD card/ Audio_DAC	Keep JP15 open
43	PA2	Ethernet	Remove R162
45	PC3	Ethernet	Remove RS1

Table 8. Daughterboard extension connector CN9 (continued)

Pin	Description	Alternate function	How to disconnect with function block on STM3210C-EVAL board
47	PC1	Ethernet	
49	PC13	IDD_CNT_EN / Anti-tamper button B4	Keep JP1 open
2	PD15	MC	Disconnect STM3210C-EVAL board from motor power drive board
4	PD13	LD2	Remove R95
6	PD11	Ethernet	Remove RS1
8	PD9	Ethernet/smartcard	Keep JP12 open
10	GND		
12	PD8	Ethernet/smartcard	Keep JP13 open
14	PB14	IO_Expander_INT	Remove R159
16	PB12	Ethernet/audio	Remove RS3
18	PB10	Ethernet	Remove RS2
20	PE15	MC	Remove R139
22	PE13	MC	Disconnect STM3210C-EVAL board from motor power drive board
24			
26	PE11	MC	Disconnect STM3210C-EVAL board from motor power drive board
28	PE9	MC	
30	GND		
32	PB2	MC	Remove R168
34	PB0	MC	Remove R167
36	PC4	Potentiometer	Remove R103
38	PA6	IDD_Measurement	Remove R79
40	PA5	MC	Remove R84
42	PA3	MC/Ethernet	Keep JP10 open
44	PA1	Ethernet	Keep JP3 open
46	PC2	Ethernet	Remove RS1
48	PC0	MC/smartcard	Remove R165
50	GND		

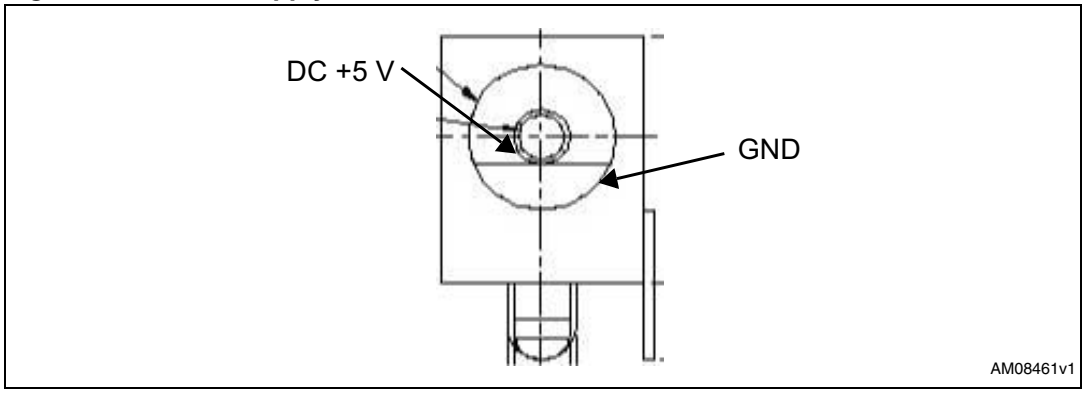
4.14 TFT LCD connector CN14

One 34-pin male header CN14 is available on the board to connect LCD module board MB785.

4.15 Power connector CN18

The STM3210C-EVAL demonstration board can be powered from a DC 5 V power supply via the external power supply jack (CN18) shown in *Figure 8*. The central pin of CN18 must be positive.

Figure 8. Power supply connector CN18 viewed from the front



For more information on STM3210C-EVAL, please refer to the UM0600 user manual.

5 Demonstration firmware system setup

5.1 Hardware requirements

a) ZigBee adapter board
b) ZigBee module SPZB260-PRO
c) VDC/2A isolated power supply is recommended
d) One JTAG programmer/debugger dongle (J-Link from SEGGER or IAR Systems™ is recommended). It is unnecessary if no modifications to the firmware code have been performed.

5.2 STM3210C-EVAL demonstration board setup

Set up the STM3210C-EVAL board as follows:

- Close jumper JP19
- Keep JP15 open
- Remove R79, R84, and R169.

5.3 STM3210C-EVAL and ZigBee adapter with M24LR62-r memory

Table 9. ZigBee adapter pin description

STM32 pin no.	Pin name	STM3210C-EVAL I/O assignment	Extension connector pin no.	ZigBee adapter I/O assignment
-	VDD	3.3 V	CN8 - Pin 48	VCC_3V3
-	VSS	GND	CN9 - Pin 50	GND
29	PA4	SPI1_NSS	CN9 - Pin 41	ZIG_SS
30	PA5	SPI1_CLK	CN9 - Pin 40	ZIG_SCLK
31	PA6	SPI1_MISO	CN9 - Pin 38	ZIG_MISO
32	PA7	SPI1_MOSI	CN9 - Pin 37	ZIG_MOSI
92	PB6	CAN2_TX/I2C1_SCK	CN8 - Pin 36	I2C1_SCK
93	PB7	I2C1_SDA	CN8 - Pin 37	I2C1_SDA
2	PE3	Trace_D0	CN8 - Pin 42	ZIG_HOST_INT
3	PE4	Trace_D1	CN8 - Pin 43	VCC-GPIO (for M24LR64-r)
4	PE5	Trace_D2	CN8 - Pin 44	ZIG_WAKE
5	PE6	Trace_D3	CN8 - Pin 47	ZIG_RSTB

5.4 How to navigate the demo menu

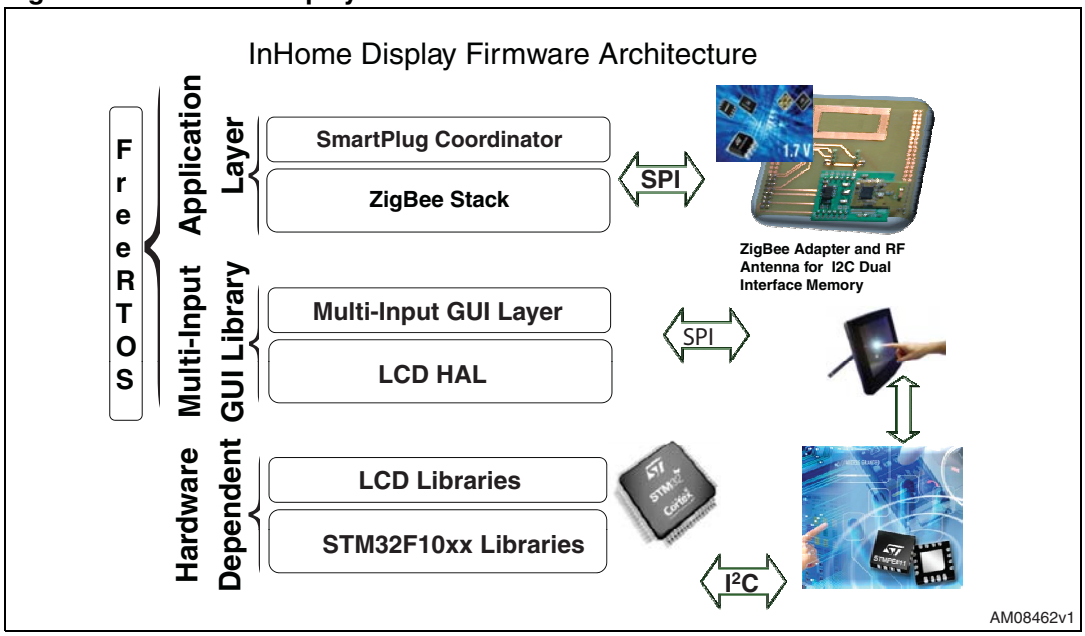
Four basic elements constitute the hardware of the system user interface: a 320 x 240 TFT LCD display, a resistive touchscreen, a 5-way (left, right, up, down, selection) micro-joystick and a pushbutton “Key”.

The joystick is primarily used for navigating between the various menu screens, within a screen and between screen items. In addition, the joystick allows the selection and editing of item values.

6 In-home display firmware

6.1 Firmware architecture

Figure 9. In-home display firmware architecture



This section describes the firmware implementation. The tasks/functions are described in the following format:

Table 10. Function description format

Function name	The name of the function
Function prototype	Prototype declaration of the function
Behavior description	Brief explanation of how the function is executed
Input parameter {x}	Description of the input parameters
Output parameter {x}	Description of the output parameters
Return value	Value returned by the function
Required preconditions	Requirements before calling the function
Called functions	Other library functions called

6.2 main.c

6.2.1 vSmartPlugSamplingTask

Table 11 describes the vSmartPlugSamplingTask task:

Table 11. vSmartPlugSamplingTask task

Function name	vSmartPlugSamplingTask
Function prototype	void vSmartPlugSamplingTask(void *pvParameters)
Behavior description	Read and sample the plug consumption values
Input parameter {x}	None
Output parameter {x}	None
Return value	None
Required preconditions	None
Called functions	No API/HAL layer functions;

Example:

```
static void vSmartPlugSamplingTask( void *pvParameters)
{
    CoordinatorType* pObjCoordinator = GetCoordinatorObj();
    int i=0;

    sinkAdvertise();
    pObjCoordinator = GetCoordinatorObj();

    //Get the Smart-Plug list
    ppSmartPlugList = pObjCoordinator->GetSmartPlugList(pObjCoordinator);

    while(1)
    {
        if ( ppSmartPlugList[0] )
        {
            smart_points1[i%N_SAMPLES] = ppSmartPlugList[0]->Energy;
            smart_points1B[i%N_SAMPLES] = ppSmartPlugList[0]->Power;
        }
        if ( ppSmartPlugList[1] )
        {
            smart_points2[i%N_SAMPLES] = ppSmartPlugList[1]->Energy;
            smart_points2B[i%N_SAMPLES] = ppSmartPlugList[1]->Power;
        }
        i++;
        if(i == N_SAMPLES)
        {
            i=0;
            for(int j=0; j<N_SAMPLES; j++)
            {
              smart_points1[j]  = 0;
              smart_points1B[j] = 0;
              smart_points2[j]  = 0;
              smart_points2B[j] = 0;
            }
        }

        if( i%8 == 0 )
```

```
            sinkAdvertise();

        vTaskDelay(2000); //20 sec
    }
}
```

The application simulates the behavior of the system in a 24 hour period considering an hour as equal to 20 seconds. After 24 hours (simulated), it resets the array values, so the GraphCharts and the histograms show a maximum value set to zero.

6.2.2 vGraphicLibraryTask

Table 12 describes the vGraphicLibraryTask task:

Table 12. vGraphicLibraryTask task

Function name	vGraphicLibraryTask
Function prototype	void vGraphicLibraryTask(void *pvParameters)
Behavior description	Check if an input device has been stimulated (touchscreen or joystick/button)
Input parameter {x}	None
Output parameter {x}	None
Return value	None
Required preconditions	None
Called functions	No API/HAL layer functions;

This function checks if an input device has been stimulated (touchscreen or joystick/button) and calls the graphic library relative function according to the object pressed.

Example:

```
static void vGraphicLibraryTask( void *pvParameters )
{
  /* Menu Initialization*/
  MENU_DeInit();
  CursorShow(195, 50);

  /* Infinite main loop -------------------------------------------------------*/
  while (1)
  {
      ProcessTouchScreenData();

      /*Time out calculate for power saving mode*/
      TimeOutCalculate();
      #ifdef USE_STM3210C_EVAL
              if (restart_calibration==1)
              {
                TS_Calibration();
                restart_calibration = 0;
                MENU_DeInit();
              }

              CursorReadJoystick(IOEXP_MODE);
              TSC_Read();
      #else
```

```
                CursorReadJoystick(POLLING_MODE);
        #endif
        vTaskDelay(2);
    }
}
```

6.2.3 prvApplicationTask

Table 13 describes the prvApplicationTask task:

Table 13. prvApplicationTask task

Function name	prvApplicationTask
Function prototype	void prvApplicationTask (void *pvParameters)
Behavior description	Start the Smartplug Coordinator Application routine
Input parameter {x}	None
Output parameter {x}	None
Return value	None
Required preconditions	None
Called functions	No API/HAL layer functions;

This function starts the smartplug coordinator application routine.

Example:

```
void prvApplicationTask( void * pvParameters )
{
    /* The parameters are not used in this task. */
    ( void ) pvParameters;

    for( ;; )
    {
      applicationTick();
      /* Wait until it is time to move onto the next string. */
      vTaskDelay(4);
    }
}
```

6.2.4 prvSetupHardware

Table 14 describes the prvSetupHardware function:

Table 14. prvSetupHardware function

Function name	prvSetupHardware
Function prototype	void prvSetupHardware (void)
Behavior description	Initialize the hardware peripheral of the STM32 MCU, and ZigBee hardware configuration
Input parameter {x}	None
Output parameter {x}	None
Return value	None

Table 14. prvSetupHardware function (continued)

Required preconditions	None
Called functions	No API/HAL layer functions;

This function is intended to initialize the hardware peripherals of the STM32 MCU, and the ZigBee hardware configuration.

Example:

```
static void prvSetupHardware( void )
{
   SystemInit();

   /* Enable GPIOA, GPIOB, GPIOC, GPIOD, GPIOE and AFIO clocks */
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                          RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD |
                          RCC_APB2Periph_GPIOE | RCC_APB2Periph_AFIO, ENABLE );

   /* Set the Vector Table base address at 0x08000000 */
   NVIC_SetVectorTable( NVIC_VectTab_FLASH, 0x0 );

   NVIC_PriorityGroupConfig( NVIC_PriorityGroup_4 );

   /* Configure HCLK clock as SysTick clock source. */
   SysTick_CLKSourceConfig( SysTick_CLKSource_HCLK );

   /* Initialize the board */
   vBoardInit();

   EZSP_Init(); //init OBJ Coordinator
}
```

6.2.5 vBoardInit

Table 15 describes the vBoardInit function:

Table 15. vBoardInit function

Function name	vBoardInit
Function prototype	void vBoardInit (void)
Behavior description	Initialize the hardware peripheral of the STM3210C-EVAL board
Input parameter {x}	None
Output parameter {x}	None
Return value	None
Required preconditions	None
Called functions	No API/HAL layer functions;

This function is intended to initialize the hardware peripheral of the STM3210C-EVAL board and the graphic library hardware parameters.

Regarding the file menu.c, it just uses the API function of the graphic library and some others of the “ZigBee stack for smartplug”, for more information please refer to the AN3128, rev. 2, application note.

7 Getting started with the system

7.1 Configure IAR tool for building, debugging, and programming the application

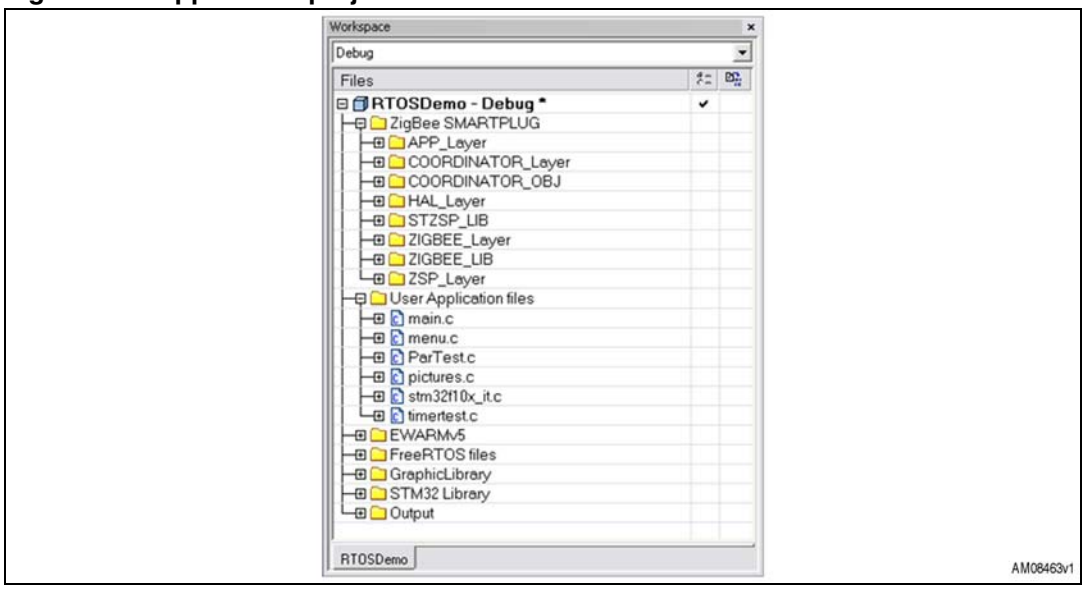
Together with the firmware library package, an example application is delivered in order to provide the final user with a real example of “In-home display” application usage.

The delivered example application has been written and developed using IAR EWARM 5.40 IDE and can be built for both STM32F10xxx medium-density and high-density microcontroller families.

The workspace is created using the IAR embedded Workbench 5.40 IDE, using the ARM®-based 32-bit STM32F10xx firmware library (ver. 3.1.2) CMSIS compliant, and in C language.

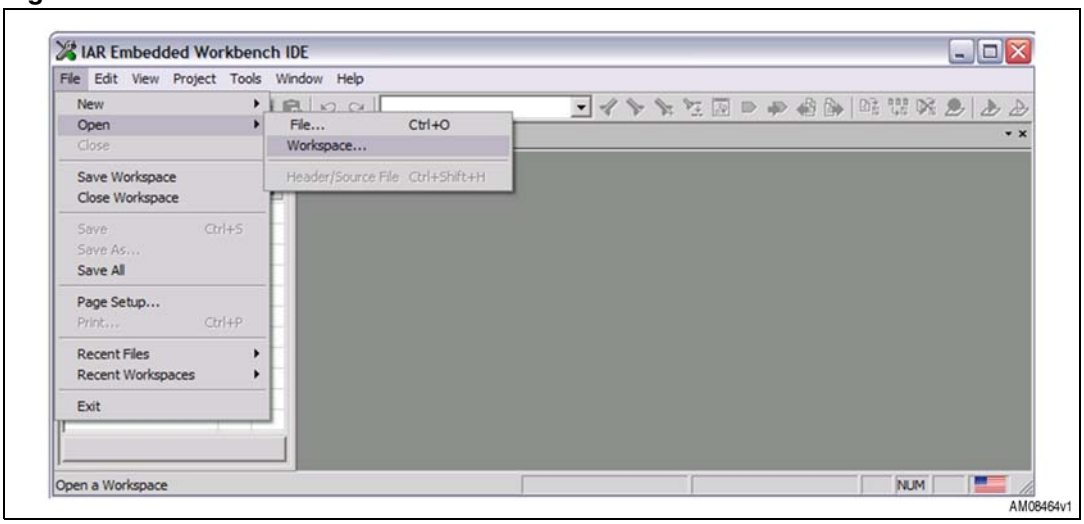
The tree structure of the project is organized separating and grouping the source files with the header files, both for the project files and the library files, as shown in *Figure 10*.

Figure 10. Application project files



In order to load the project, click on File\Open\Workspace and in the window that appears select, in \project folder\EWARM, the file “Project.eww”.

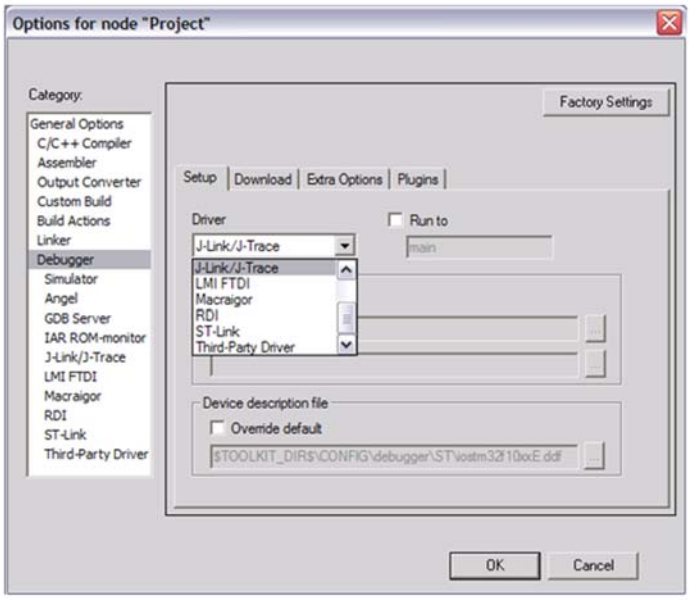
Figure 11. IAR embedded workbench main window



On the main node, where the program name located in the files window is shown, right click the mouse button and select options. In the window which appears, select the debugger item in the category list box, and select the proper debugging tool in the driver list box, then press the Ok button. In the proposed example the J-Link dongle is used (*Figure 12*).

Press the Make icon or click on Project\Rebuild All. No error or warning should appear once compiling has completed. Connect the J-Link tool to the USB port of the PC, and connect the flat cable with the programming adapter. Plug the adapter into the dongle connector. Press the debug icon, CTRL+D or click project\debug. The debugger starts to download the firmware to the dongle through the J-Link debugger\programmer. Press the Go button, F5 or click debug\go in order to execute the firmware in debug mode. To run the dongle in standalone mode, press the stop debugging icon, CTRL+SHIFT+D or click debug\stop debugging. Then remove the J-link adapter from the dongle and reset the board by unplugging and plugging the power cable back in.

Figure 12. IAR embedded workbench debugger options



In order to use the In-home display application project, it is necessary to:

- Include all the firmware delivered in the In-home display package containing the FreeRTOS core files. See *Figure 10*
- Create the desired menu application functions in the file menu.c
- Put inside “picture.c” the HEX dump of the pictures to be used with the application GUI
- Implement a main function as described in the following section.

7.2 Example application - main.c

An example of a main application is given below. The main function contains an example of the In-home display application initialization/configuration and implements the classic operations:

```
/* Standard includes. */
#include <stdio.h>

/* Scheduler includes. */
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "semphr.h"

/* Library includes. */
#include "stm32f10x.h"
#include "stm32f10x_it.h"
#include "STM3210c_eval_lcd.h"
#include "LcdHal.h" //SB

/* Demo app includes. */
#include "flash.h"
#include "partest.h"
#include "common.h"
#include "micro_clocks_irqs.h"
#include "micro.h"

#include "menu.h"
#include "graphicObject.h"
#include "cursor.h"
#include "stm32d-eval_io_expander.h"
#include "LcdHal.h"
#include "TscHal.h"
#include "pictures.h"
#include "stm32f10x.h"
#include "stm32f10x_rcc.h"
#include "misc.h"
#include "SmartPlugObj.h"
#include "CoordinatorObj.h"
#include "COORDINATOR_Layer.h"
#include "ZSP_Layer.h"

#ifdef USE_STM3210E_EVAL
 #include "stm3210e_eval_lcd.h"
#elif USE_STM3210C_EVAL
 #include "stm3210c_eval_lcd.h"
 #include "touchscreen.h"
#endif

/* Private define ------------------------------------------------------------*/
```

```
/* Touchscreen Controller and Joystick DEFINES */
#define TSC_GPIO_PORT_SOURCE         GPIO_PortSourceGPIOB
#define TSC_GPIO_PIN_SOURCE          GPIO_PinSource14
#define JOY_GPIO_PORT_SOURCE         GPIO_PortSourceGPIOB
#define JOY_GPIO_PIN_SOURCE          GPIO_PinSource14
#define JOY_GPIO_SELECT_PORT         GPIOG
#define JOY_GPIO_SELECT_PIN          GPIO_Pin_7
#define JOY_GPIO_RIGHT_PORT          GPIOG
#define JOY_GPIO_RIGHT_PIN           GPIO_Pin_13
#define JOY_GPIO_LEFT_PORT           GPIOG
#define JOY_GPIO_LEFT_PIN            GPIO_Pin_14
#define JOY_GPIO_UP_PORT             GPIOG
#define JOY_GPIO_UP_PIN              GPIO_Pin_15
#define JOY_GPIO_DOWN_PORT           GPIOD
#define JOY_GPIO_DOWN_PIN            GPIO_Pin_3
#define JOY_GPIO_RCC_APB_PERIPH1     RCC_APB2Periph_GPIOG
#define JOY_GPIO_RCC_APB_PERIPH2     RCC_APB2Periph_GPIOG
#define JOY_GPIO_RCC_APB_PERIPH3     RCC_APB2Periph_GPIOG
#define JOY_GPIO_RCC_APB_PERIPH4     RCC_APB2Periph_GPIOG
#define JOY_GPIO_RCC_APB_PERIPH5     RCC_APB2Periph_GPIOD
#define TSC_EXTI_IRQ_CHANNEL         EXTI15_10_IRQn
#define JOY_EXTI_IRQ_CHANNEL         EXTI15_10_IRQn
#define TSC_EXTI_LINE                EXTI_Line14
#define JOY_EXTI_LINE                EXTI_Line14
#define TSC_GPIO_PORT                GPIOA
#define TSC_GPIO_PIN                 GPIO_Pin_14
#define TSC_I2C_DEVICE_REGISTER      0x82
#define JOY_I2C_DEVICE_REGISTER      0x88
#define M24LR64_I2C_DEVICE_REGISTER 0x24
#define TSC_I2C_PORT                 I2C1

/* User Button GPIO Port and Pin*/
#ifdef USE_STM3210C_EVAL
  #define USER_BUTTON_PORT           GPIOB
  #define USER_BUTTON_PIN            GPIO_Pin_9
#elif USE_STM3210E_EVAL
  #define USER_BUTTON_PORT           GPIOG
  #define USER_BUTTON_PIN            GPIO_Pin_8
#endif

/* LCD Controller DEFINES */
#define LCD_CTRL_PORT_NCS            GPIOB
#define LCD_GPIO_DATA_PORT           GPIOC
#define LCD_CTRL_PIN_NCS             GPIO_Pin_2
#define LCD_CTRL_PIN_NWR             GPIO_Pin_15
#define LCD_CTRL_PIN_RS              GPIO_Pin_7
#define LCD_GPIO_PIN_SCK             GPIO_Pin_10
#define LCD_GPIO_PIN_MISO            GPIO_Pin_11
#define LCD_GPIO_PIN_MOSI            GPIO_Pin_12
#define LCD_GPIO_RCC_APB_PERIPH      RCC_APB2Periph_GPIOC
#define LCD_GPIO_RCC_APB_PERIPH_NCS RCC_APB2Periph_GPIOB
#define LCD_GPIO_REMAP_PORT          GPIO_Remap_SPI3
#define LCD_RCC_APB_PERIPH           RCC_APB1Periph_SPI3
#define LCD_RCC_AHB_PERIPH           RCC_AHBPeriph_FSMC
#define LCD_SPI_PORT                 SPI3

#ifdef USE_STM3210C_EVAL
  #define LCD_CONNECTION_MODE         GL_SPI
#elif  USE_STM3210E_EVAL
  #define LCD_CONNECTION_MODE         GL_FSMC
#endif

#define RCC_AHBPeriph_FSMC            ((uint32_t)0x00000100)
```

```
/* Exported variables-------------------------------------------------------*/
LCD_HW_Parameters_TypeDef* pLcdParam;
TSC_HW_Parameters_TypeDef* pTscParam;
JOY_HW_Parameters_TypeDef* pJoyParam;
BTN_HW_Parameters_TypeDef* pBtnParam;
extern volatile GL_u8 touch_done;
GL_s16 smart_points1[N_SAMPLES];
GL_s16 smart_points1B[N_SAMPLES];
GL_s16 smart_points2[N_SAMPLES];
GL_s16 smart_points2B[N_SAMPLES];
SmartPlugType** ppSmartPlugList;
SmartPlugType* pCurrentSmartPlug;
extern GL_Page_TypeDef pageS2B;
extern GL_Page_TypeDef pageS2C;

/* Private macro -----------------------------------------------------------*/
/* Private variables -------------------------------------------------------*/
static volatile ErrorStatus HSEStartUpStatus /*= SUCCESS*/;
volatile GL_u8 restart_calibration = 0;
GPIO_InitTypeDef GPIO_InitStructure;
static u32 TimingDelay;
static xTaskHandle xEmberTaskHandle, xApplTaskHandle, xSmartSamplingTaskHandle,
xGraphLibTaskHandle;

/* The time between cycles of the 'check' functionality (defined within the tick
hook. */
#define mainCHECK_DELAY         ( ( portTickType ) 5000 / portTICK_RATE_MS )

/* Task priorities. */
#define EMBER_Task              ( tskIDLE_PRIORITY + 3 )
#define APPLICATION_Tick_Task   ( tskIDLE_PRIORITY + 2 )
#define mainINIT_TASK_PRIORITY  ( tskIDLE_PRIORITY + 1 )

/* The period of the system clock in nano seconds. This is used to calculate the
jitter time in nano seconds. */
#define mainNS_PER_CLOCK ((unsigned portLONG)((1.0 /
                          (double)configCPU_CLOCK_HZ)*1000000000.0))

/*-----------------------------------------------------------*/

/* Configure the hardware for the demo. */
static void prvSetupHardware( void );
static void vEmberTickTask( void *pvParameters );
extern void EZSPInit(void);

static void prvApplicationTask( void * pvParameters );
void vApplication_EXIT2_ISRFunc( void );
void vApplication_USART0_ISRFunc( void );

/* Configure the hardware for the demo. */
static void prvSetupHardware( void );

/* Configure the board, LCD, Joystick, Button, GPIO and other peripherals as required
by the demo. */
static void vBoardInit( void );

/* Check Touchscreen, Joystick, Button in polling mode as required by the demo. */
static void vGraphicLibraryTask(void *pvParameters);

/* Check the SmartPlug Energy data as required by the demo. */
static void vSmartPlugSamplingTask( void *pvParameters);
```

```
/*
 * Configures the high frequency timers - those used to measure the timing
 * jitter while the real time kernel is executing.
 */
extern void vSetupHighFrequencyTimer( void );

u16 ticksSinceLastHeard[APPLICATION_ADDRESS_TABLE_SIZE];
EmberNetworkStatus networkState = EMBER_NO_NETWORK;

/* insert  Start */
xSemaphoreHandle xSemaphore_Ember = NULL;
xSemaphoreHandle xSemaphore_Serial = NULL;
extern void EZSP_Init(void);
extern void emberTick(void);
extern void processSerialInput(void);
extern void applicationTick(void);
/* insert  End */
/*-------------------------*/

int main( void )
{
#ifdef DEBUG
  debug();
#endif

  pLcdParam = NewLcdHwParamObj ();
  pTscParam = NewTscHwParamObj ();
  pJoyParam = NewJoyHwParamObj ();
  pBtnParam = NewBtnHwParamObj ();

  prvSetupHardware();

  /* Start the tasks. FreeRTOS API. */

  xTaskCreate( vEmberTickTask, "EMBER", configMINIMAL_STACK_SIZE, NULL, EMBER_Task,
               xEmberTaskHandle );
  xTaskCreate( vSmartPlugSamplingTask, ( signed portCHAR * ) "SmartPlugSampling",
               configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY,
               xSmartSamplingTaskHandle );
  xTaskCreate( vGraphicLibraryTask, ( signed portCHAR * ) "GraphicLibrary",
               configMINIMAL_STACK_SIZE*2, NULL, tskIDLE_PRIORITY+1,
               xGraphLibTaskHandle );

  /* Start the scheduler. */
  vTaskStartScheduler();

//  emberLeaveNetwork();

/* Will only get here if there was insufficient memory to create the idle task. The
idle task is created within vTaskStartScheduler(). */
  for( ;; );
}
```

The initialization process is charged with preparing the basic mechanism of the system:

- Hardware peripheral configuration and initialization
- ZigBee stack initialization
- FreeRTOS task creation
- Starting system.

The clock distribution and the interrupt settings are two components that are strongly dependent on the target project. An example of clock rate may be 72 MHz as the maximum speed of the current STM32 microcontroller. It can be decreased to reduce the power consumption. The clock rate assumptions are:

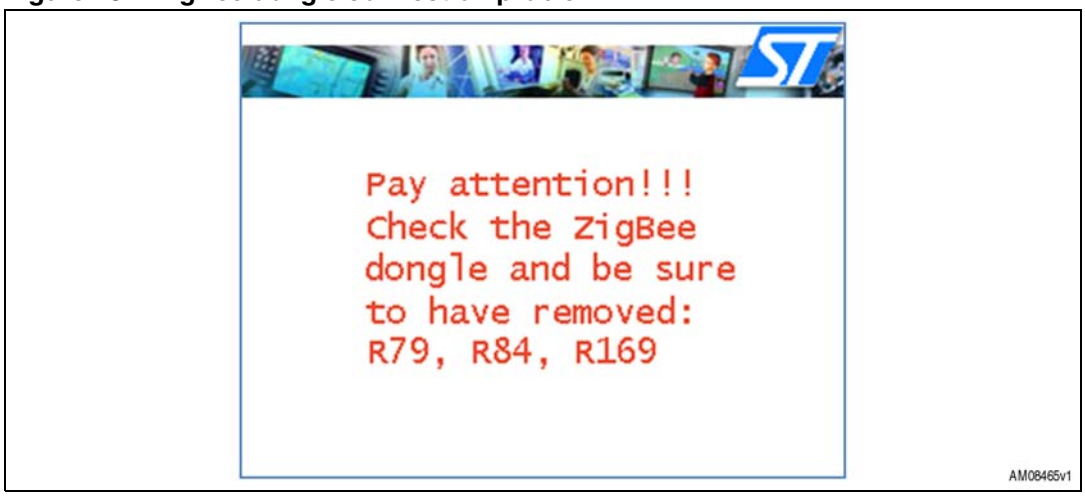
- System HCLK - 72 MHz
- Low speed peripheral PCLK1 - 72 MHz
- High speed peripheral PCLK2 - 36 MHz
- Analog to digital converter ADCCLK - 36 MHz

The interrupt setting situation is very similar to clock distribution. The library functions involved with interrupt managing do not take the priorities into account; they only perform very necessary and absolutely common settings to make them serviceable.

8 In-home display GUI application

If the ZigBee dongle is not connected or the resistors R79, R84, and R169 have not been removed from the STM3210C-EVAL, the following screen is shown:

Figure 13. ZigBee dongle connection problem

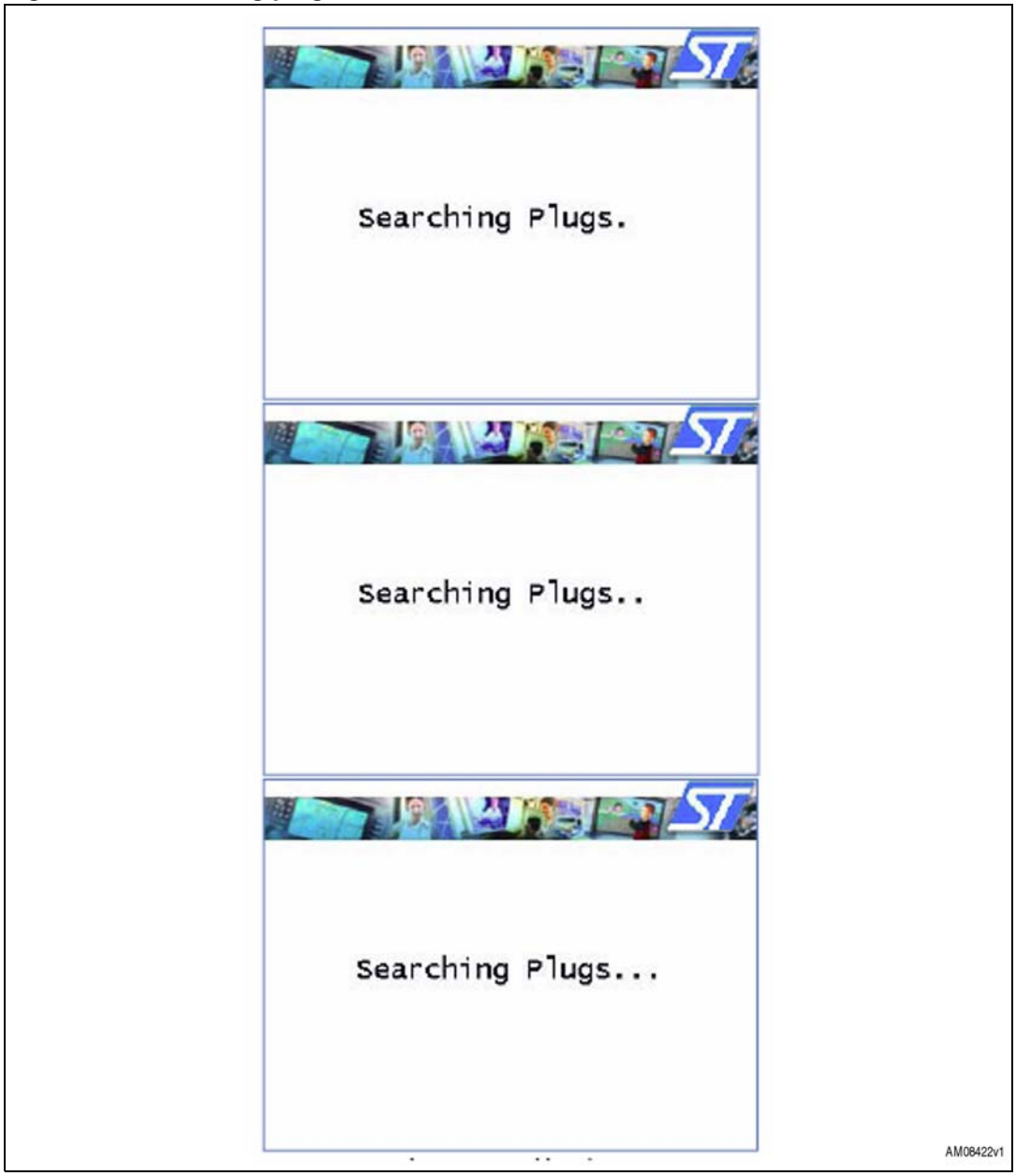


After a board reset, if the firmware is correctly loaded into the Flash memory and the board power is correctly supplied, the main screen is displayed as shown in *Figure 14*.

Figure 14. Main menu

If Search Plugs is selected, the coordinator scans for smartplug devices for a certain length of time and the display shows the following screens in loop:

Figure 15. Searching plugs



After that, if some plugs are connected, it adds them to the smartplug list and then shows the home screen.

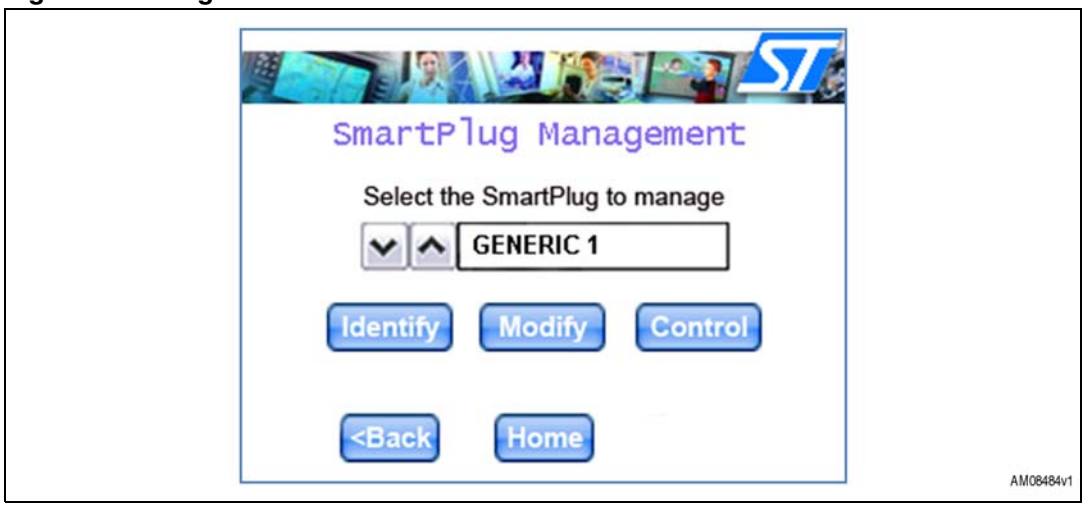
If you click on Management two situations are possible:

- If no plug is connected to the smartplug coordinator, the following screen is shown:

Figure 16. No plug detected

- Once more plugs are detected, the following screen is shown:

Figure 17. Plugs detected

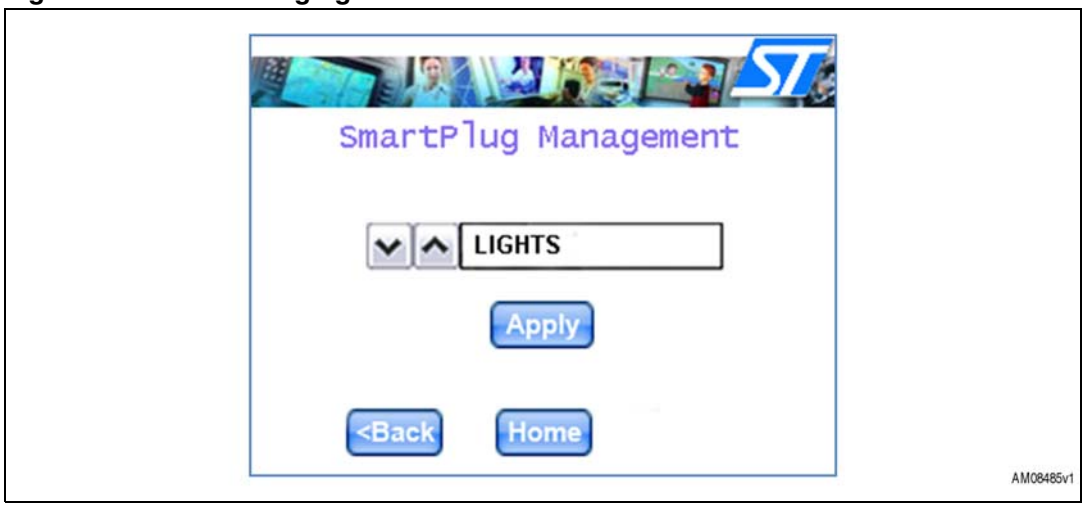


There are three main buttons: identify, modify, and control. They allow to manage the smartplug device.

The Identify button is useful to activate the LED flashing on the relative plug, so this way the user can easily identify which plug they are going to manage through the GUI.

If Modify is clicked, the user can change the label of the selected plug, through the screen shown in *Figure 18*, choosing the new label from a list contained in a combobox.

Figure 18. Label changing



The user should scroll the list of the labels and click on the Apply button in order to save the information and set the label also in the plug device via ZigBee protocol.

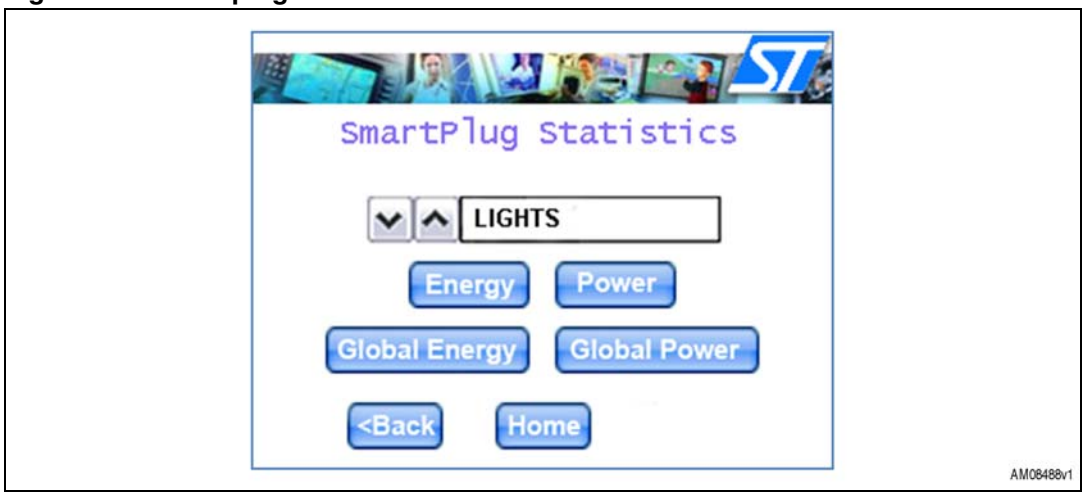
When Control is clicked, it is possible to switch ON/OFF the smartplug, in case of relay type, or regulate the output power, in case of TRIAC type. The following screens show both possibilities:

Figure 19. TRIAC smartplug management

Figure 20. Relay smartplug management

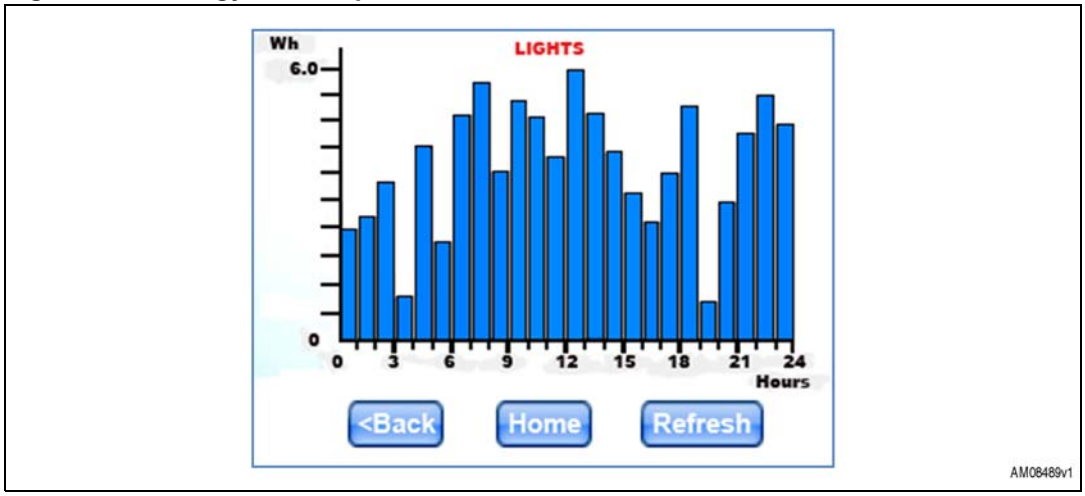
In the home screen, by clicking on Statistics, it is possible to see the electrical consumption of the single smartplug device or the total consumption related to all plugs connected to the smartplug coordinator. The following screen is shown:

Figure 21. Smartplug statistics



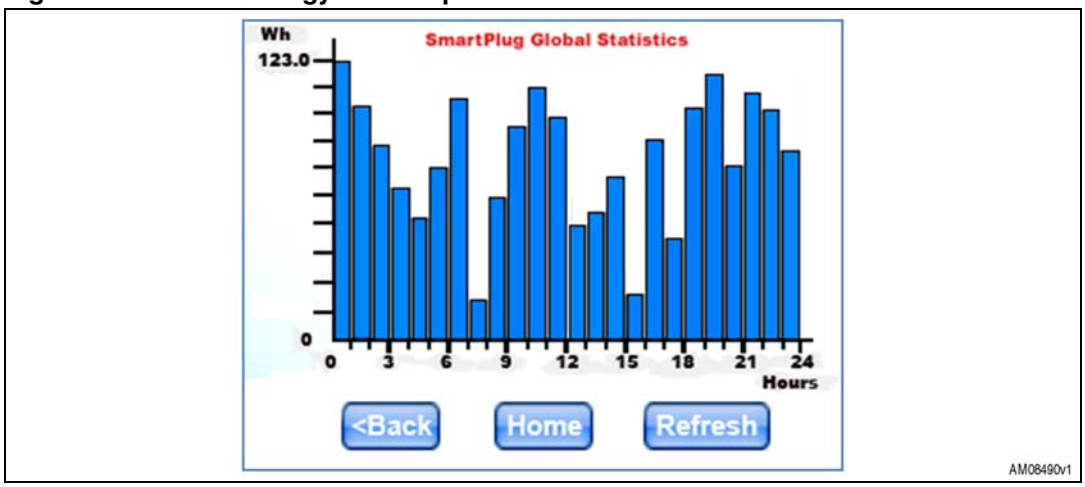
When Energy is clicked, the following screen, representing the energy consumption of the selected smartplug device, is shown:

Figure 22. Energy consumption



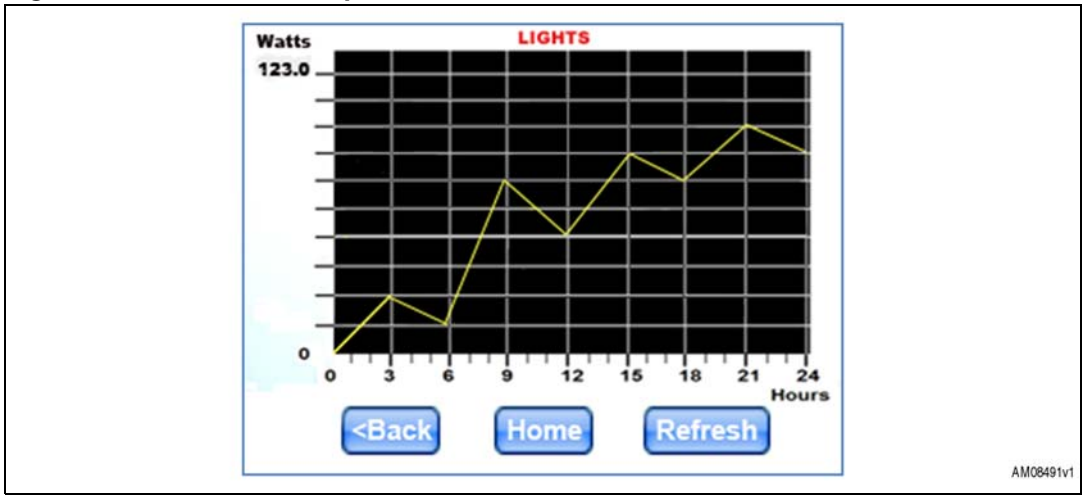
When Global Energy is clicked, the following screen, representing the energy consumption of the whole smartplug network, is shown:

Figure 23. Global energy consumption



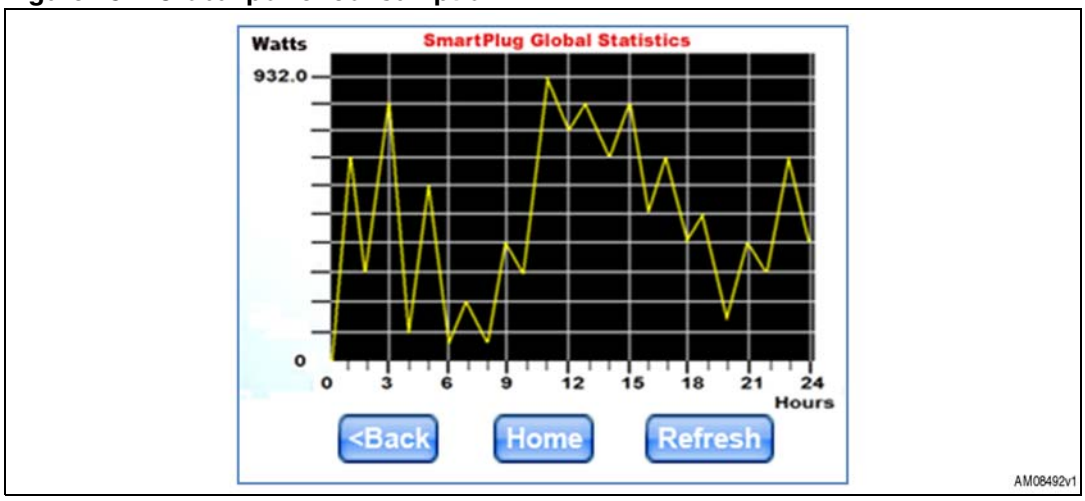
When Power is clicked, the following screen, representing the power consumption of the selected smartplug device, is shown:

Figure 24. Power consumption



When Global Power is clicked, the following screen, representing the power consumption of the whole smartplug network, is shown:

Figure 25. Global power consumption



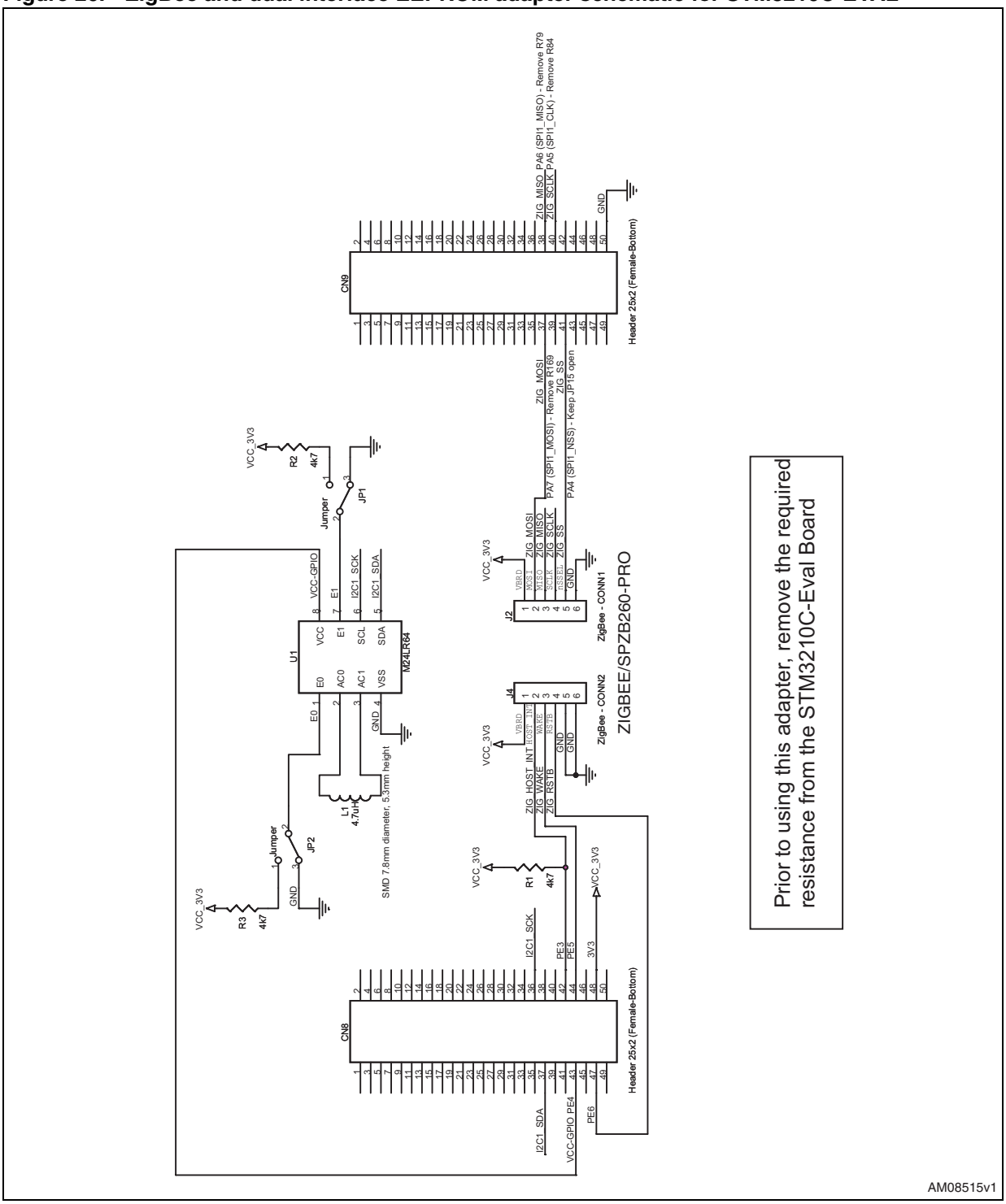
When the Refresh button is clicked, it is possible to make a refresh of the graph chart in relation to the sampled points of the power consumption.

When the Home button is clicked, it returns to the home screen.

9 Schematics

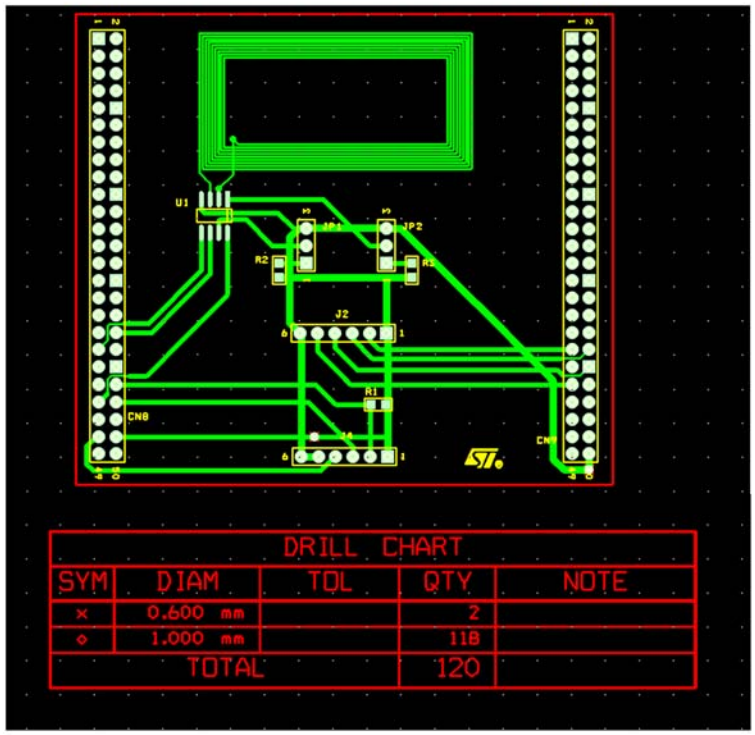
9.1 Smartplug board schematics and layout

Figure 26. ZigBee and dual interface EEPROM adapter schematic for STM3210C-EVAL



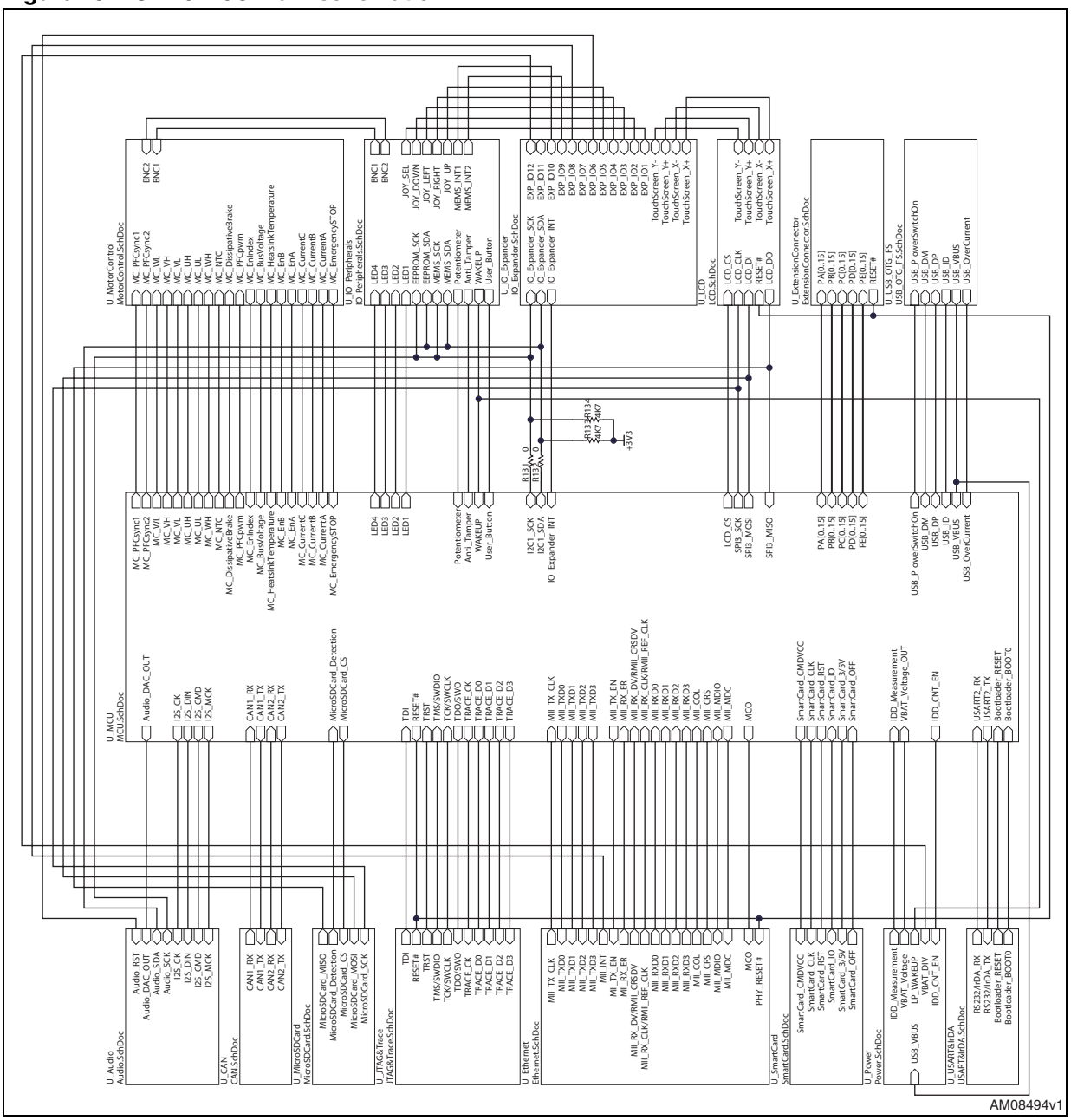
9.2 ZigBee/RF adapter for smartplug and dual interface memory

Figure 27. ZigBee and dual interface EEPROM adapter layout for STM3210C-EVAL



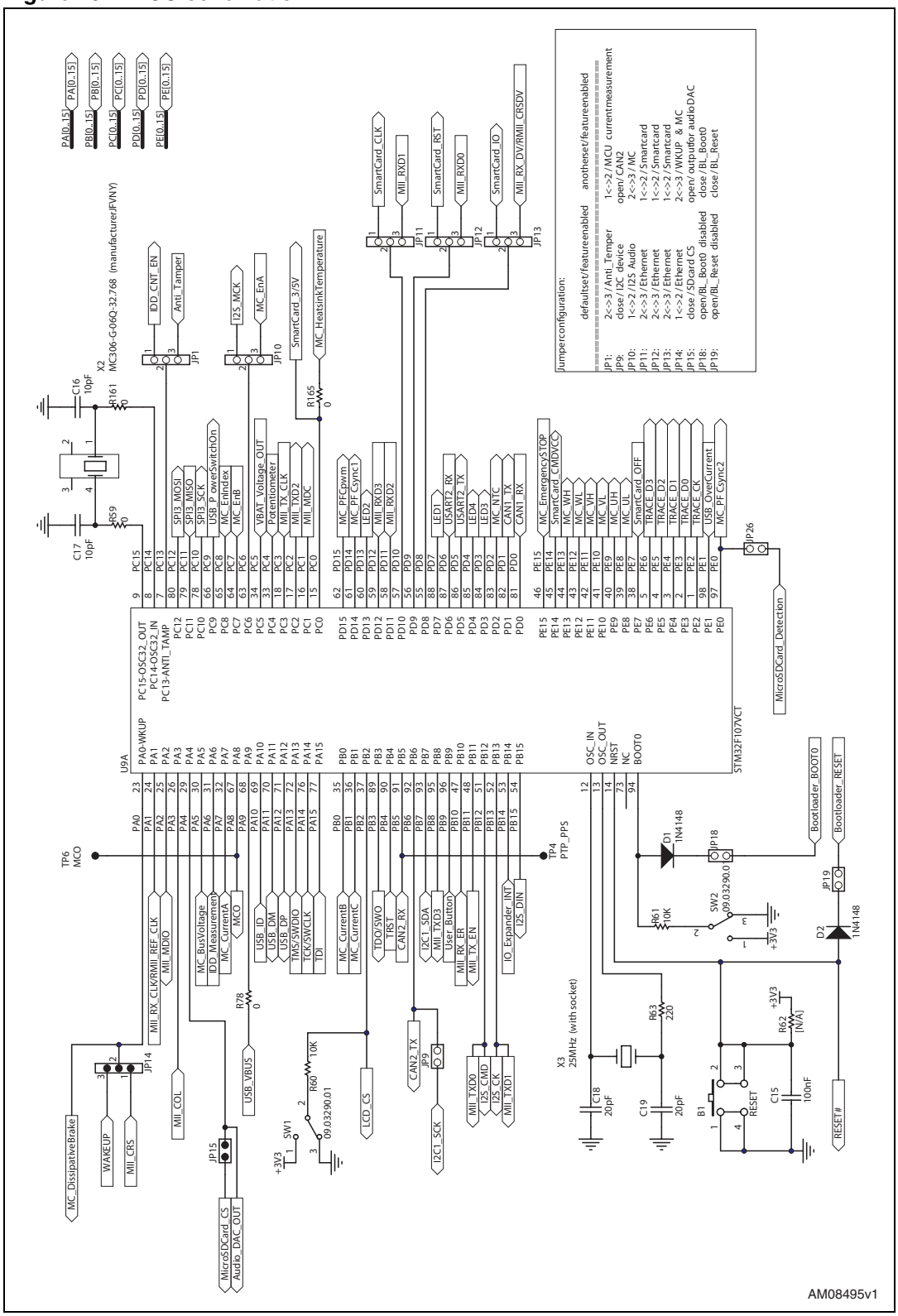
9.3 STM3210C

Figure 28. STM3210C main schematic



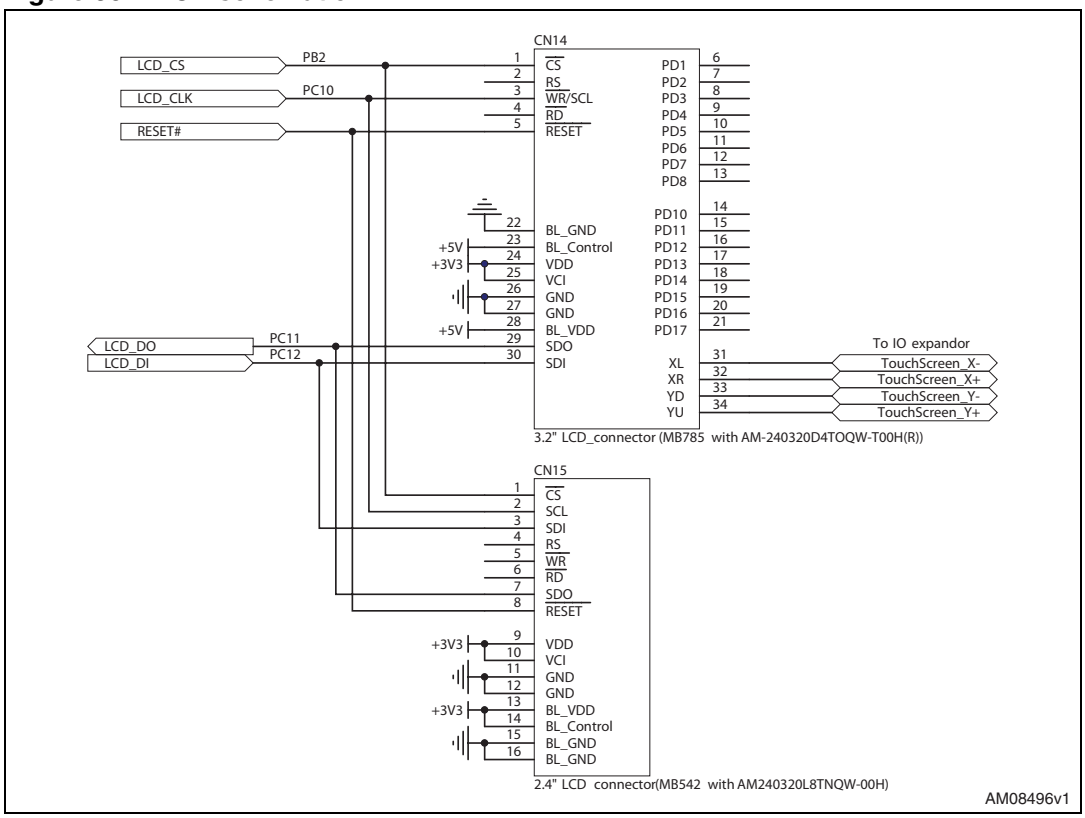
9.4 MCU

Figure 29. MCU schematic



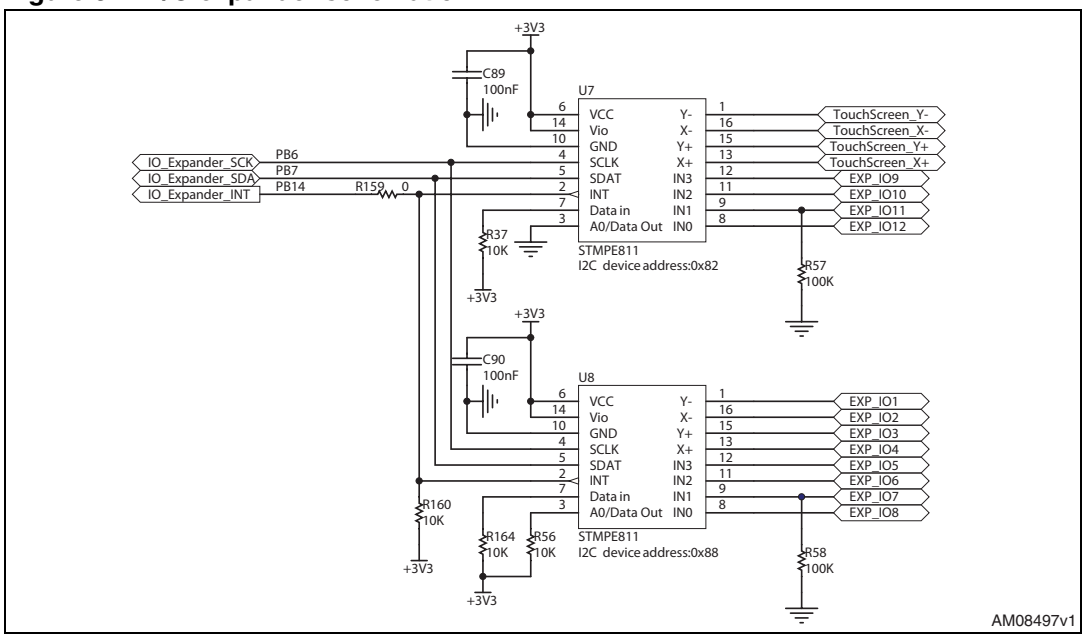
9.5 LCD

Figure 30. LCD schematic



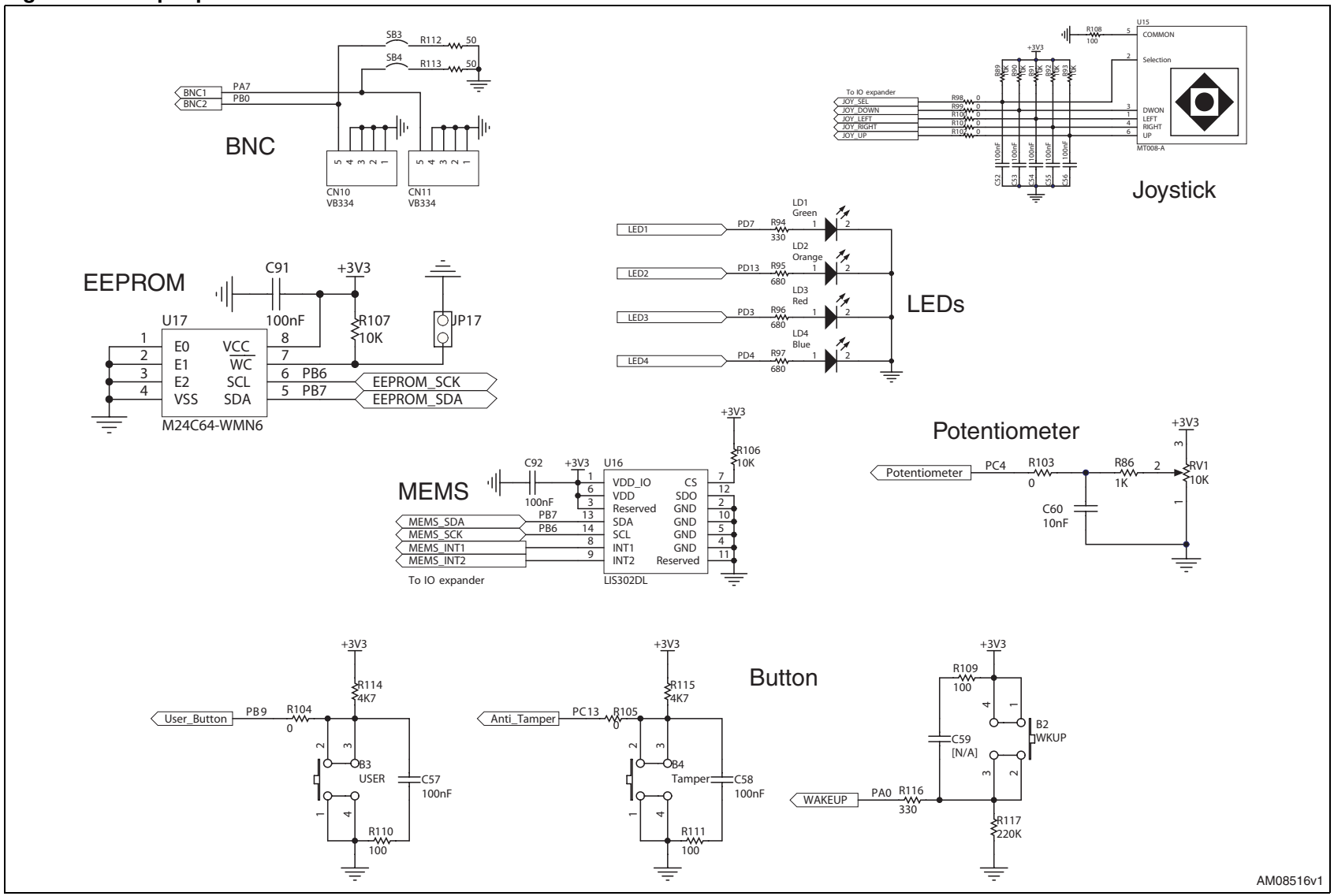
9.6 I/O expander

Figure 31. I/O expander schematic



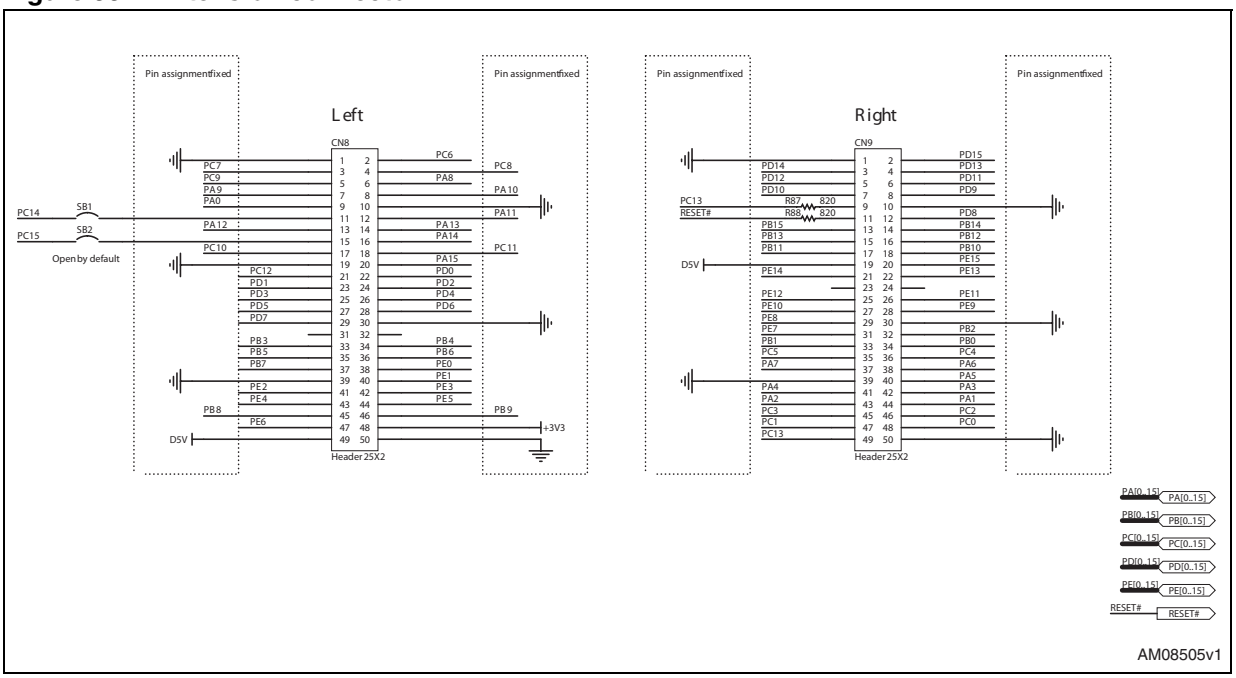
9.7 I/O peripherals

Figure 32. I/O peripherals



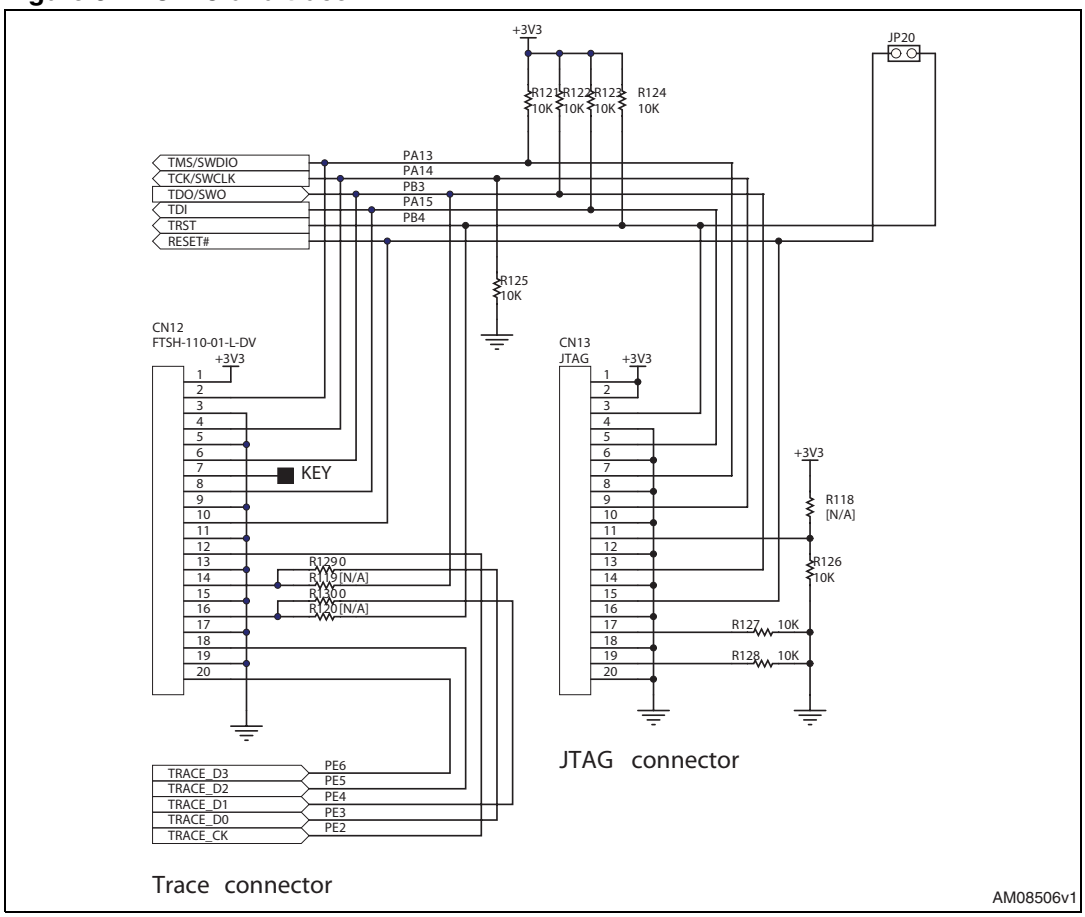
9.8 Extension connector

Figure 33. Extension connector



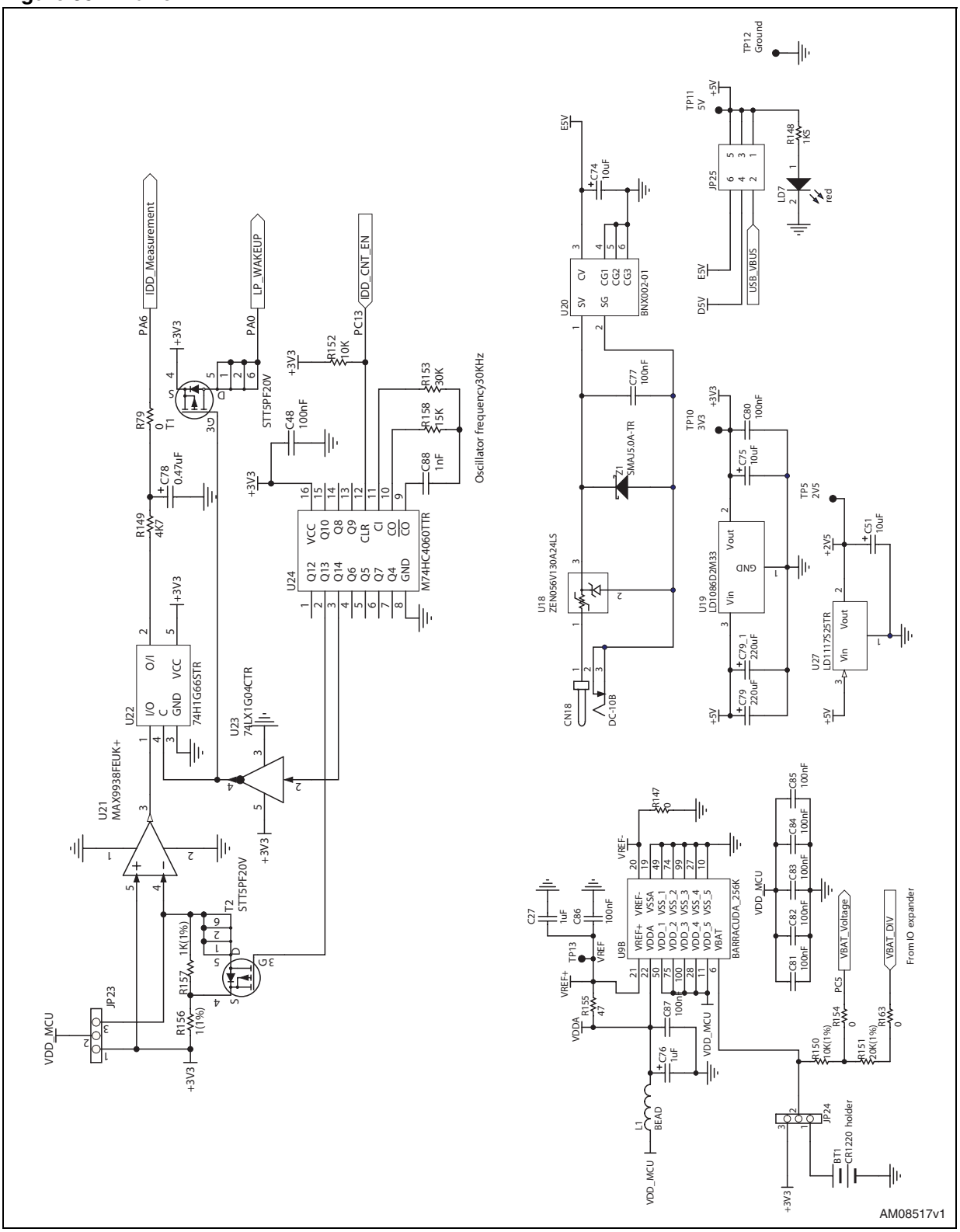
9.9 JTAG and trace

Figure 34. JTAG and trace



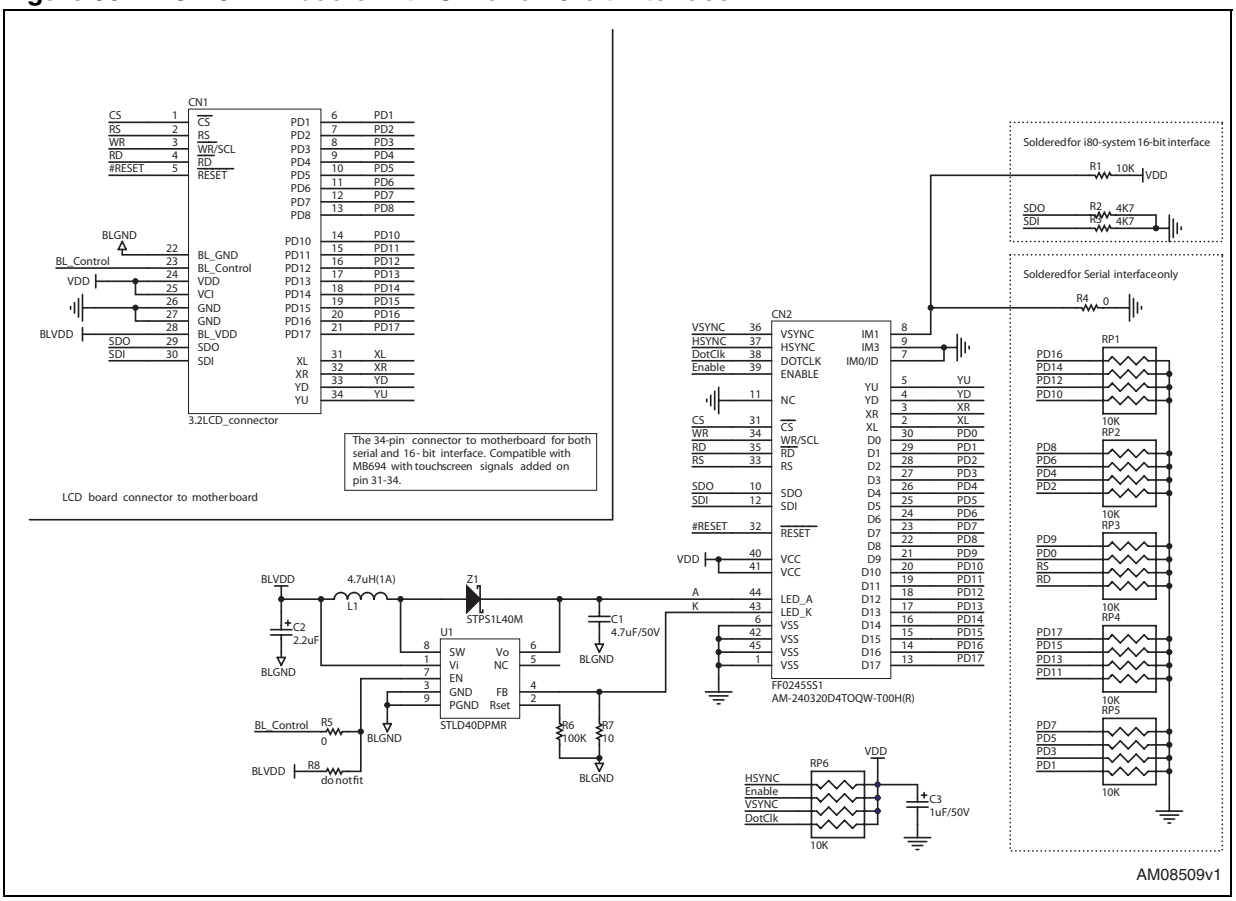
9.10 Power

Figure 35. Power



9.11 LCD 3.2" module with SPI and 16-bit interface 1

Figure 36. LCD 3.2" module with SPI and 16-bit interface 1



10 References

1. AN3128 application note
2. RM0008 reference manual
3. STM32F10xFWLib 3.2.1 help file
4. STM811 datasheet
5. TN0074 technical note
6. UM0608 user manual
7. STEVAL-IHP001V3 schematics diagram
8. AN2993 application note
9. M24LR64-r datasheet
10. AN2972 application note.

11 Revision history

Table 16. Document revision history

Date	Revision	Changes
14-Dec-2010	1	Initial release.