AN3046
Application note

Multimaster field-bus applications in homes and buildings

Introduction

The purpose of this application note is to describe a multimaster field-bus protocol, using a power line modem based on the ST7540 (refer to application note AN2451) and an STM32 microcontroller used for interfacing the modem. This protocol is used for managing data exchange between different mains-connected devices. Possible applications are power management, heating or cooling system management, lighting control, alarm systems, and remote control appliances.

The protocol implements a multimaster type, where each device can be a master in a communication starting the transmission. This introduces the need of data collision management.

Contents

List of figures

1 Application description

The application is based primarily on two ST components: an ST7540 power line modem and an STM32F103C8 ARM®-based 32-bit MCU used for interfacing the modem.

1.1 The ST7540 power line modem

1.1.1 Introduction

The ST7540 is a half-duplex synchronous/asynchronous FSK modem designed for power line communication network applications.

Device operation is controlled by means of an internal register, programmable through the synchronous serial interface. There are four ST7540 working modes:

- Data reception
- Data transmission
- Control register read
- Control register write

An internal 24 or 48-bit (in extended mode) control register allows the management of all the programmable parameters. In this application the extended register mode is used.

The ST7540 features two types of host communication interfaces, the SPI and the UART. In this application the SPI interface is used. The ST7540 can access the mains in two different ways: synchronous or asynchronous mode. Synchronous mode (using the SPI) is chosen for this application.

1.1.2 Control register access

Access to the control registers is achieved using the same lines of the mains interface (RxD, TxD, RxTx and CLR/T) plus the REG_DATA line.

With REG_DATA = 1 and RxTx = 0, the data present on TxD is loaded into the control register MSB first. The control register content is updated at the end of the register access section (REG_DATA falling edge).

After the power-on, and each time the working mode of the ST7540 must be changed, using the extended register mode, exactly 48 bits must be transferred to the ST7540 in order to properly write the control register, otherwise writing is aborted. After each writing operation, a control register reading operation is normally performed: with REG_DATA = 1 and RxTx = 1, the content of the control register is sent on the RxD port.

1.1.3 Receiving and transmitting mode

The receive section is active when the RxTx pin = 1 and REG_DATA = 0. The input signal is read on the RX_IN pin, while the transmission mode is set when the RxTx pin = 0 and the REG_DATA pin = 0. In transmitting mode the FSK modulator and the power line interface are turned on, and the ST7540 manages the transmission timing according to the baud rate selected.

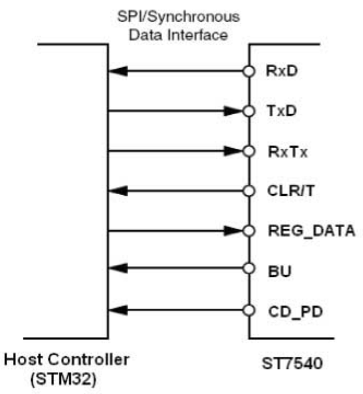
1.1.4 Host communication interface

The host communication interface is shown in *Figure 1*. In order to interface the host controller a five-line interface: RxD, TxD, RxTx, CLR/T and REG_DATA, is used.

The host can achieve mains access by selecting REG_DATA = 0 and the choice between data transmission or data reception is performed by selecting the RxTx line (if RxTx = 1, the ST7540 receives data from the mains, if RxTx = 0, it transmits data over the mains).

In synchronous mode the ST7540 is always the master of the communication and provides the clock reference on the CLR/T line. If the RxTx line is set to 1, and REG_DATA = 0 (data reception), the ST7540 enters into an idle state and the CLR/T line is forced low. After a “bit” time (depending on the programmed baud rate) the modem starts providing received data on the RxD line.

Figure 1. Host communication interface



Two more pins are used by the host, the CD_PD and the BU.

- CD_PD (carrier/preamble detect): in control register extended mode (for detailed information on the control register bit value, see the ST7540 datasheet) the CD_PD line can be used to recognize if a header has been sent during the transmission. The header recognition function is enabled by setting the proper control register pin (control register bit 18 = 1) and the CD_PD line is forced low for one period of the CLR/T line when a frame header is detected. In this mode, the CLR/T and RxD signal are always present, even if no header has been recognized.
- BU (band-in-use): The band-in-use block has a carrier detection (CD_PD) like function but with a different input sensitivity and with a different BandPass filter selectivity (see the ST7540 datasheets for details). This pin is forced high when a signal in the band is detected. This function is enabled only in receiving mode.

1.2 The STM32F103C8 ARM-based 32-bit MCU

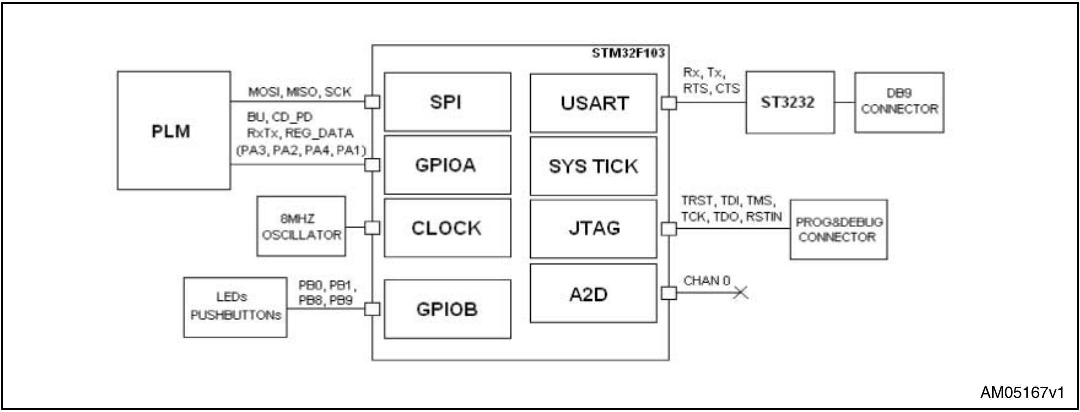
1.2.1 Introduction

The STM32F103C8 performance line incorporates the high performance ARM Cortex™-M3 32-bit RISC core, operating at a 72 MHz frequency, high speed embedded memories and an extensive range of enhanced I/Os and peripherals connected to two different buses. The device is provided with standard and advanced communication interfaces such as I^2Cs, SPIs, USARTs, etc.

The STM32F103C8 performance line embeds a nested vectored interrupt controller able to handle up to 43 maskable interrupt channels and 16 priority levels.

In *Figure 2* the functional diagram of the application is shown, describing all the microcontroller peripherals used.

Figure 2. Functional diagram



1.2.2 Peripheral description

- SPI: the SPI peripheral is used for the ST7540 interfacing, both for the modem configuration at startup and the data communication during normal working mode. The SPI is configured to work at 2400 bps and is set as a slave device. The data alignment is set to the first falling edge of the clock signal. The data exchange is synchronized at the interrupt level of the peripheral.
- USART: this peripheral is used for the data exchange between the dongles and a PC (or another device provided with a standard RS232 port) for transmitting and receiving a data frame through the power line. The USART is also used for higher level configuration of the dongle (local ID, target address, etc.) as described in the following paragraphs. The USART is configured to work at a baud rate of 115200 bps, 8 data bits, 1 stop bit and no parity nor hardware handshake.
- GPIO: the GPIOs are used for the interfacing of the control and the status pins of the power line modem (CD_PD, BU, REG_DATA, etc.), the two pushbuttons and the LEDs.
- ADC: the analog to digital converter is used to obtain a random value used in the application. The “0” channel of the ADC is a floating pin in the application, as shown in *Figure 2*. An analog to digital conversion of the voltage value read on this pin, supplies

a random value, depending on many environmental factors (as this pin is not fixed to any voltage reference).

- SYSTICK: the system tick is a timer used as a time-base generator. The timer prescaler is set to generate an interrupt each 100 µs. In the systick interrupt routine a global variable is incremented and used as a time reference in the application.
- CLOCK: the system clock is set to 72 MHz by using an external 8 MHz quartz and the internal PLL with a multiplication factor of 9. Refer to the STM32F103C8 advanced ARM-based 32-bit MCUs reference manual for details on the clock control.
- JTAG: the JTAG interface allows a standard debug tool to connect to the CPU and download code into the internal RAM or Flash memory. The JTAG port also supports basic run control (single step and setting breakpoints etc.) as well as being able to view the contents of memory locations. The debug tool can connect using the standard 5-pin JTAG plus the RESET and the power pins (Vdd and ground).

1.3 Power line configuration

The ST7540 is a multi-channel and multi-function transceiver. An internal 24 or 48-bit (in extended mode) control register allows the management of all the programmable parameters. In this application 48 bits (extended mode) are used.

The programmable functions are the channel frequency, the interface baud rate, the deviation, the detection method, the sensitivity mode, frame header recognition (in extended mode), etc. Please refer to the ST7540 datasheet in order to see all the possible configurations.

The configuration registers are sent to the PLM through the SPI peripheral after the power-on, sending the MSB first.

PLM working mode, set by the configuration, is the following:

- Frequencies: 132.5 kHz
- Baud rate: 2400 bps
- Deviation: 0.5
- WatchDog, Tx timeout: disabled
- Frequency detection time: 500 µs
- Preamble with conditioning
- Mains interfacing mode: synchronous
- Output clock, output voltage level freeze: off
- Header recognition: enabled
- Frame len counter: disabled
- Header length: 16 bits
- Extended register: enabled
- Sensitivity: normal
- Input filter: enabled

The carrier/preamble detection block is a digital frequency detector circuit used to manage the mains access and to detect an incoming signal. It notifies the host controller to the presence of a carrier modulated at the programmed baud rate for at least 4 consecutive symbols (“1010” or “0101” are the symbol sequences detected). As the used control register mode is the extended mode, a header recognition is signaled by forcing CD_PD low for one period of CLR/T line. The CLR/T and RxD signal are always present, even if no header has been recognized.

The header value to be used as reference is sent through the SPI during the PLM initialization phase, and it is located in the second and third control byte. The header value set in this application is 0xE389.

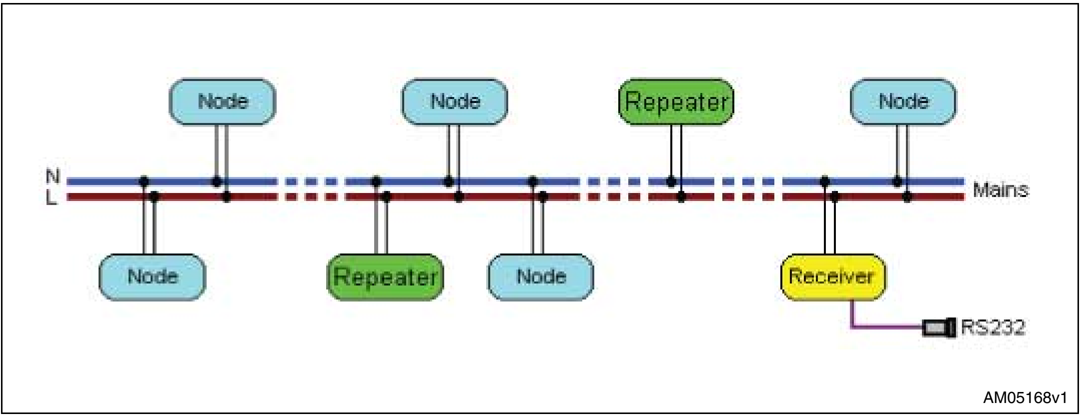
2 Communication protocol description

2.1 Network architecture

2.1.1 General description

The system architecture is shown in *Figure 3*. Several network devices, called nodes, and repeaters or receivers (or in a more generic way, also referred to as dongles), are connected to the mains, creating a network for data exchanging.

Figure 3. System architecture



The network setup is done by using a dedicated PC GUI software (see PC GUI software description, *Section 5*).

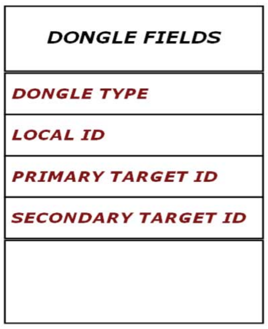
The implemented network model is a multiple transmitters (multimaster devices) and one receiver (slave device) type.

The network is composed of a group of dongles programmed as nodes (transmitters) which send data to a Dongle Programmed as a receiver. Each node can directly address the receiver or, if the network conditions are adverse (too much noise injected in the electrical network, low output impedance devices connected close to the PLM, relevant long distance from a node and the receiver, etc.), it can address a dongle, called a repeater, which, once having received a data frame from one of the nodes, re-transmits the data frame to its target device. The target device of a repeater can be the receiver or another repeater. This mechanism realizes a multi-hop networking. With the provided PC software it is possible to configure each dongle offline by setting all its items. In this manner, the network is statically defined.

2.1.2 Node description

A node is a dongle connected somewhere in the electrical line which is able to transmit a data frame each time the pushbutton is pressed (fixed data frame) or, alternatively, each time a command is sent through its USART by using the PC software GUI (user-defined data frame). *Figure 4* shows the parameters associated to the node:

Figure 4. Dongle parameters



- Dongle type: this field is used to set the working mode of the dongle (node, repeater or receiver)
- Local ID: represents the device address (each device must have a different address in the same network)
- Primary target ID: is the pointed dongle address (which has no meaning for a receiver) and can be the receiver or a repeater address both for a node or a repeater dongle
- Secondary target ID: is the address of an alternative device which is addressed by the dongle when three (this value is set by a #define directive) attempts of addressing the primary remote ID fails. All the other bytes must be set to “0”.

A status LED is present on the dongle board, indicating the data frame transmission, error condition, programming command execution, etc. See *Section 3* for details on LED status indications.

2.1.3 Repeater description

A repeater is a dongle interposed between a node and a receiver used when a node cannot directly reach the receiver, implementing a kind of hopping structure. The node addresses this repeater, sending the data frame to it; then, the repeater forwards the received data frame to the receiver. Each time the node has delivered the frame to the repeater, the communication flow is considered as concluded and the delivery task is shifted from the node to the repeater.

2.1.4 Receiver description

The receiver is a device which never starts a communication. Its task is to receive data frames sent by nodes and repeaters in the network. In order to see the received data frames, a PC GUI interface can be used (PLM Frame Manager, see *Section 5*) or the standard Windows HyperTerminal application can be used instead.

2.1.5 Network working principles

Each node connected to the line is a master device, so it can start a communication at any time. This could cause a conflict between another master that wants to start communication at the same moment (or before a transmission is completed). In order to avoid this conflict certain mechanisms were implemented.

The first mechanism is the verification of the BU before any transmission. If the network is busy, no transmission is performed by a node.

Once the channel is free, all the nodes that were awaiting a transmission calculate a random time before any transmission. This random time is called backoff time and it is composed of a random part, calculated by means of the ADC, giving a time contribution varying between 2 and 150 ms, and a fixed part of about 200 ms.

The random part should guarantee a very low probability of two transmissions happening at the same time. However, it is still possible, as the implemented conflict management type is not CSMA/CD (carrier sense multiple access with collision detect) because the PLM has no signal indicating an ongoing conflict (also known as the “listen while talking” mechanism), but it is a CSMA/CA (collision avoidance) type, using the BU signal and a timeout for conflict avoidance.

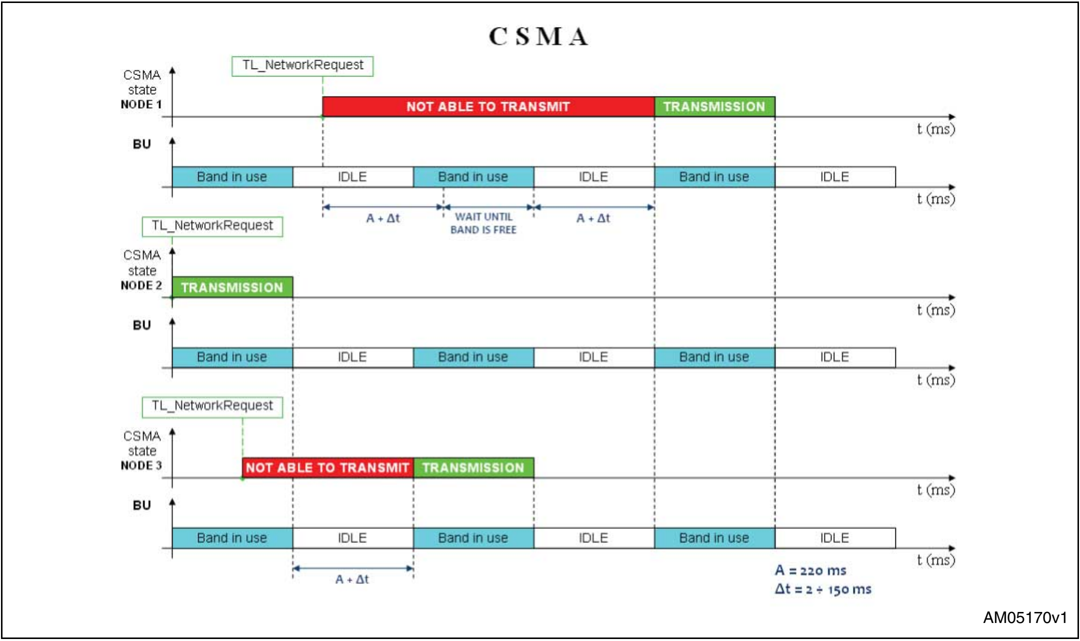
Each data frame that a node sends through the line is acknowledged by the target device (the receiver or a repeater) by means of the back transmission of an ACK frame towards the node. The ACK frame time transmission is around 100 ms.

In order to guarantee the completion of a couple of data frames/ACK frames, the backoff time must be chosen slightly longer than the ACK time (in this application it equals 150 ms, considering some delays introduced in the communication state machine)

Once the backoff time has elapsed, if the band is still in use, the node again calculates another backoff time, until the band is free.

Figure 5 shows an example of a net with three nodes that have to transmit a data frame at the same time.

Figure 5. Example of a CSMA/CA conflict management



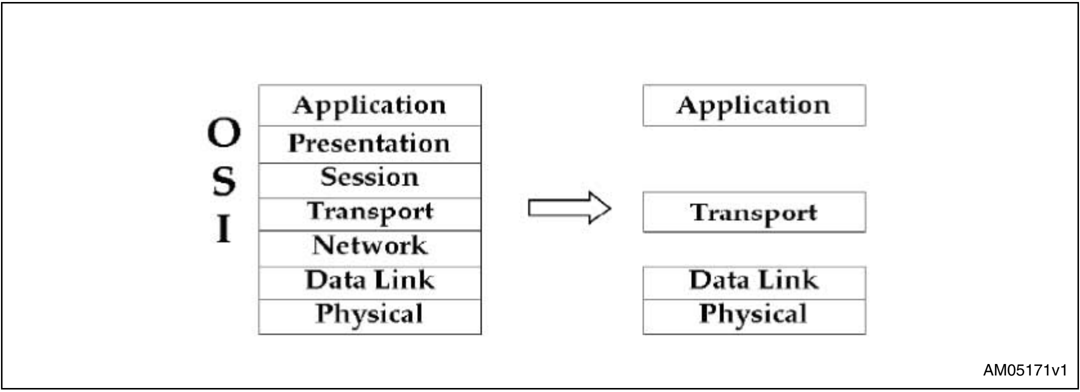
It may be possible that two nodes want to access the net at the same time by calculating the same random value; the BU check occurs in the same instant and both nodes see the free band and start communication at the same time.

In this way, both data frames result as corrupted by each other, and the receiver simply ignores them and does not send back an ACK frame. Once the ACK timeout has elapsed, each node re-tries to send the same frame, again calculating a new backoff time. It is highly unlikely that the new backoff time is the same for both. The process is terminated once both frames are delivered and acknowledged by the receiver.

2.2 Implemented ISO/OSI layers

The implemented structure is similar to the ISO/OSI (open system interconnection, *Figure 6*) and the derived Konnex KNX PL132.

Figure 6. ISO\OSI structure type



Even if the proposed model is not compliant with the Konnex standard, it implements many features similar to this protocol, such as conflict management, the acknowledgement mechanism, timing management, physical layer technology, data management, etc.

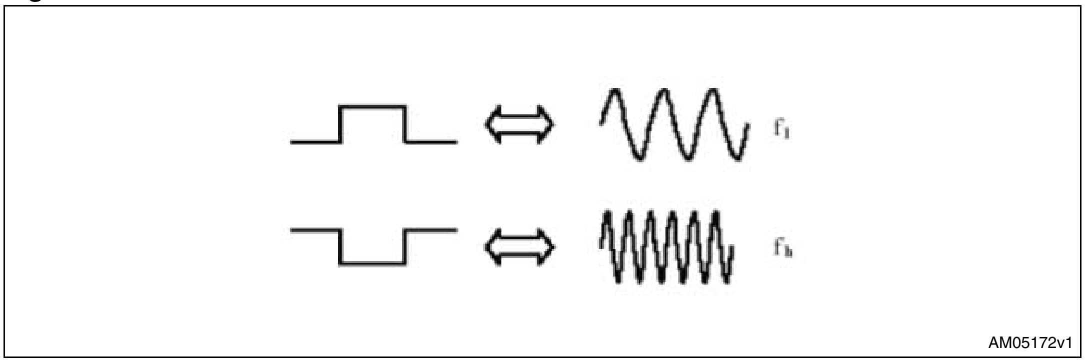
The following chapters describe each implemented layer in detail.

2.3 The physical layer

The physical layer is the lowest implemented layer. Its task is to solve the communication problems connected to the physical aspects of data communication.

Data communication is done by means of a power line modem, the ST7540, which is a half-duplex synchronous/asynchronous FSK (frequency shift key) modem, designed for power line communication network applications (see the ST7540 datasheet for details). The FSK modulation translates a digital bit into an analog signal with a different frequency, depending on the bit value (*Figure 7*).

Figure 7. FSK modulation



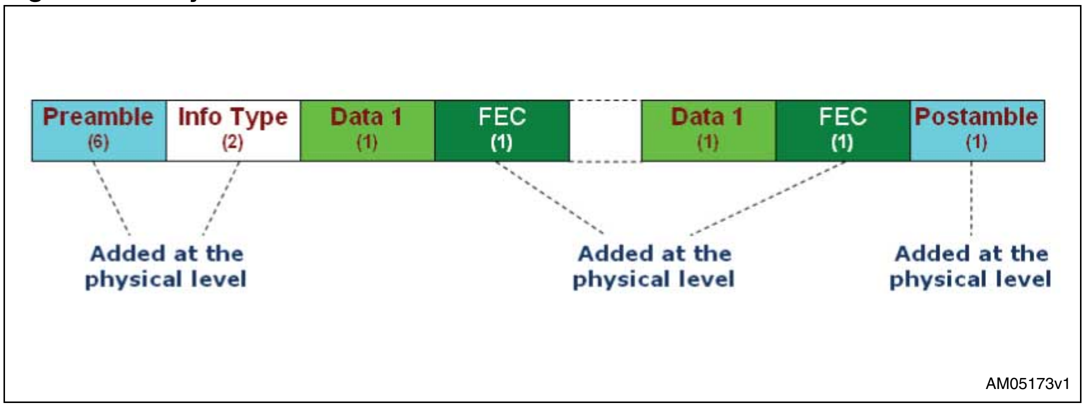
The average value of the two frequencies corresponds to the carrier frequency, which is fixed to 132.5 kHz ± 0.2 %. The data codification is an NRZ type with a baud rate of 2400 bps, the same as specified in the KNX PL132.

The databits are transferred by the microcontroller to the modem through an SPI interface plus some control pins and the REG/DATA pin, which specifies if the data transferred between the microcontroller and the PLM is configuration data (REG) or frame data (DATA). Another pin used is the RxTx pin, which specifies the direction of the communication (as it is a half-duplex communication).

With the PLM as the master device, it provides the clock signal through the CLR/T pin. The microcontroller manages the synchronization between the control pins, the SPI clock signal and the data sent or received on the TxD/RxD (MOSI and MISO for the microcontroller) pins. The physical layer isn't concerned with the correctness of the information transmitted; it just offers a physical interface between the upper layers and the hardware delegated to the communication, providing two communication routines (PL_PLMSetTX and PL_PLMSetRX) used in the upper level (data link).

The information used in the implemented protocol is shown in the following figure (*Figure 8*).

Figure 8. Physical frame structure


Preamble (6)
Info Type (2)
Data 1 (1)
FEC (1)
Data 1 (1)
FEC (1)
Postamble (1)
Added at the physical level
Added at the physical level
Added at the physical level
AM05173v1

- Preamble: this is a sequence of 6 bytes. The first 4 bytes are set to 0xAA and the last two bytes are set to 0xE389. The power line decodification starts when the PLM detects a sequence of “0101” or “1010” bits on the power line, and after that a hardware header equal to 0xE389.
- Info type: this identifies which kind of data is being transmitted. The possible values are 0xC153 for a data frame and 0xC1A1 for an ACK frame.
- Data: contains the information to be transmitted.
- FEC: another operation performed before data frame transmission, at the physical level, is the adding of the FEC (forwarding error correction) information to each application frame databyte. The FEC bytes are redundant data added to the original message, used to correct data in the case of a noisy network, within some boundries, reducing the number of re-transmissions. The implemented algorithm is the polynomial x6+x5+x4+x3+1 which is able to correct up to 3 consecutive bits in a sequence of 14 bits (including errors in the FEC field itself), optimized for noise in the mains with a maximum duration of 1 ms.
- Postamble: this is the complement of the last bit of the last FEC byte repeated twice (the significant postamble is two bits). For synchronization reasons, the FEC and the postamble are each contained in a byte, where dummy zeroes are added to complete the byte alignment (00XXXXXX and 000000YY, where X is the FEC and Y is the postamble).

2.4 The data link layer

The data link layer is the second layer of the proposed structure. The main function of this layer is the management of data transmission and reception, providing the upper level with data without communication errors or assuring the transmission of data received from the upper level.

The protocol implemented in this layer is the connectionless type with acknowledgement.

In the KNX standard extended mode, the frame length can be a maximum of 64 bytes, therefore, in order to transmit more than 64 bytes, the data frame must be segmented. The data link layer must provide the management of data frame segmentation and the routines for the composition of the original frame must be inserted here.

In the presented communication protocol, the data frame length is assumed to be the maximum 100 bytes (maximum transmission unit is 100), and segmentation is not implemented. In the case of the transmission of more than 100 bytes, fragmentation is necessary and can be implemented directly at the application level.

Other tasks that make up the data link layer are; network access management, frame error verification, ACK and timeout management and the transport of the consistent data received to the upper layer.

2.5 The transport layer

The transport layer in the proposed model is considered the “user” of the lower data link layer. This layer is similar to the network layer and the transport layer of the ISO/OSI model.

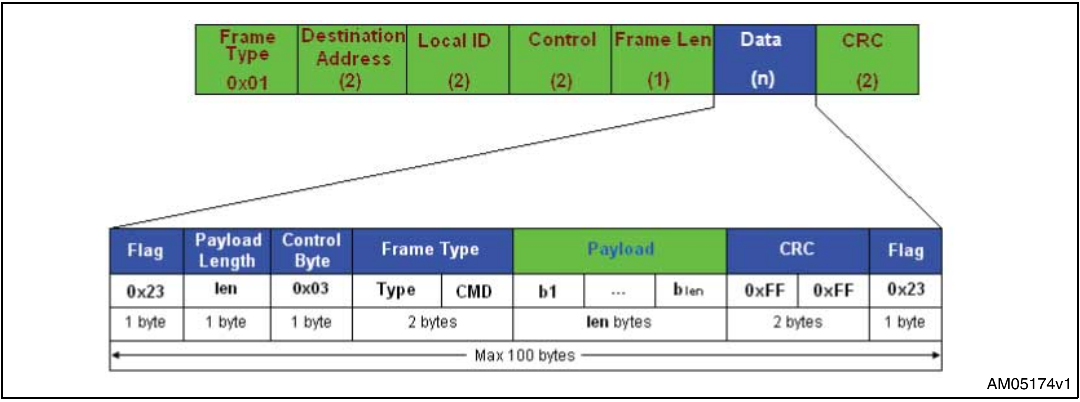
This layer supplies two services to the application layer:

```
TL_NetworkIndication
TL_NetworkRequest
```

These two services are used for data transmission and reception. The data frame and the ACK frame structure is shown in *Figure 9*.

- Frame type: this is a byte that can be 0x01 for a data frame or 0x03 for an ACK frame.
- Destination address: this is used for addressing a device connected to the network. Each device address is two bytes long, allowing the interfacing of 65536 devices.
- Local ID: this is the address of the transmitting device, this is transferred in order to acknowledge the correct device.
- Control byte: this is a fixed field with a value of 0xAA55 for data frame and 0x55AA for ACK frame.
- Frame len: this indicates the number of data bytes that follow this field.
- Data: present only on the data frame type, it is the information that comes from the application level. In the ACK frame there isn’t any additional information to send, therefore this field is not present.
- CRC: this is the cyclic redundancy check, it is the checksum of the entire frame.

Figure 9. Transport frame structure



2.6 The application layer

The application layer is the top layer of the structure. All the routines for frame management are implemented here, both from the node/repeater and the receiver side.

The application layer, in the node device, manages the requests coming from the pressure of the pushbutton or the data frame arriving through the RS232 channel, using here a filtering algorithm that, searching in the received data, detects the presence of the data frame enclosed between two flag fields.

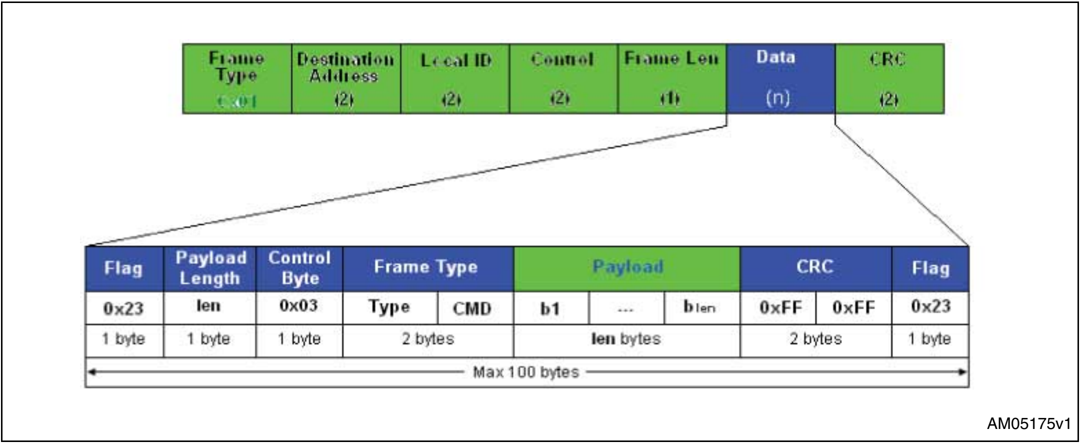
At the receiver side, the application layer manages the frame received from the lower layer and sends it through the RS232 channel. The data frames received can be visualized by using the PLM Frame Manager GUI interface.

Finally, at the repeater side, this layer manages the incoming frames direct to it, redirecting it to its target device (i.e. the receiver or another repeater for the next hop level).

The data frame used at the application level is shown in *Figure 10*. It is composed of the following parts:

- Flag: this is a fixed user-defined symbol (in the proposed firmware the flag is the symbol “#”, 0x23 in hexadecimal). The Flag is present at the beginning and at the end of the data frame.
- Payload length: the second byte of the data frame indicates the payload length, which can be a maximum of 92 bytes, reaching a total frame length of a maximum of 100 bytes.
- Control byte: the next byte represents a control byte, and is fixed to 0x03, but it can be easily changed by modifying a #define directive on the firmware.
- Frame type: the frame type field has two bytes: the first (TYPE) indicates if the frame is a programming frame (0xCA) or a data frame (any value different from 0xCA), the second (CMD) only has a meaning in programming mode, representing the command accepted by the GUI interface. The programming sequence is explained in the following paragraphs.
- Payload: the payload field contains the user-defined data which represents the information that must be transmitted to the receiver.
- CRC: the last field is the CRC, calculated for the user frame, which is not used in this firmware, but if used, can be inserted here.

Figure 10. Application frame structure

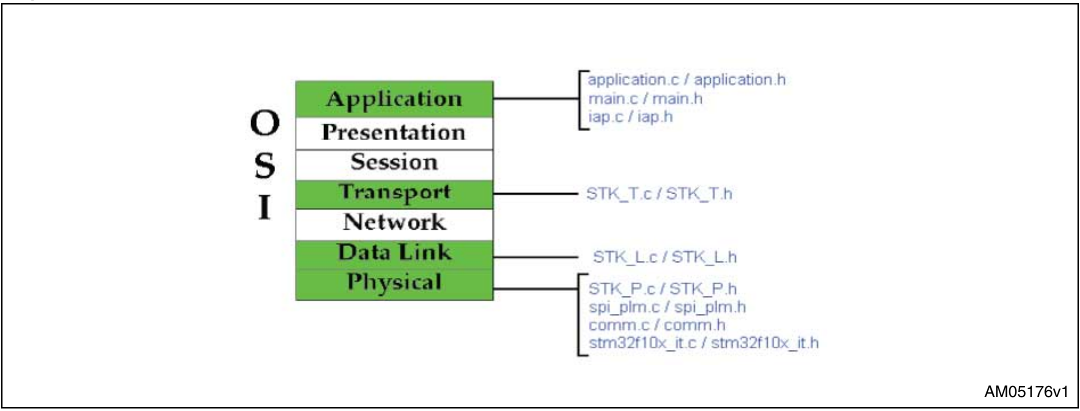


3 Firmware description

3.1 Introduction

The firmware architecture is done by following a modular approach: each layer is built upon the lower layer, starting from the physical layer that contains all the routines for the microcontroller peripheral initialization and interfacing, up to the application layer which implements the main state machine for the device behavior (*Figure 11*).

Figure 11. Firmware structure vs. OSI/ISO model



3.2 Firmware architecture

3.2.1 Main routine

After all the initializations, in the never ending loop placed in the main routine, two state machines run, one is the application layer engine and the other is the stack engine, which looks after all the communication and interfacing with the other layers.

These two routines are the APP_NetworkSM() and the LL_StackUpdate() and are updated during the normal working of the microcontroller core:

```
while(1)
{
  APP_NetworkSM();
  LL_StackUpdate();
}
```

Many other events are managed asynchronously by the interrupt routines, as explained further in this chapter.

3.2.2 STM32 standard library and physical layer APIs

The initialization of the microcontroller peripherals is done by using the STM32 standard library functions. The configuration and initialization are done in the routine Prog_Init() in the main.c file. The peripherals initialized in this routine are; the main and the peripheral clocks, the general purpose I/O pins used in the application (i.e. for the LEDs), the interrupt controller, the driver for the power line modem (SPI peripheral, IO pins for the modem interfacing and interrupt priorities), the ADC used for computing the random part of backoff time, and the USART for the data frame management.

The STK_P.c physical layer module contains all the routines used for the time management (timebase, timeouts, etc.), and for the band-in-use verification PL_BandInUse(), which tests the BU signal 10 times and returns TRUE only if the band is signaled as in-use at least 8 times, and the two transmission and reception APIs for PLM data management:

```
void PL_PLMSetTX(u8 Xmode, u8 *Source, u8 len, u8 Class_Value)
```

and

```
void PL_PLMSetRX(u8 Xmode, u16 nOfBytes)
```

The PL_PLMSetTX transmission routine accepts as parameters the PLM mode (PLM configuration register or data transmission), the data source, the length of the data frame, and the class value that is used for transmitting a data frame or an ACK frame.

The routine builds the physical frame, adding the information as the preamble (4 bytes to 0xAA), the physical header programmed on the PLM initialization registers, the class value, the FEC for each data byte (a FEC table stored in the Flash memory, tableFEC[] is used) and the postamble. After the frame preparation, the proper PLM pins are set (REG/DATA and Tx pin) and the SPI Tx interrupt is enabled in order to allow the SPI peripheral to send bytes through the MOSI line.

The data transmission is managed by the interrupt routine of the SPI peripheral, by means of the SPI APIs implemented in the spi_plm file.

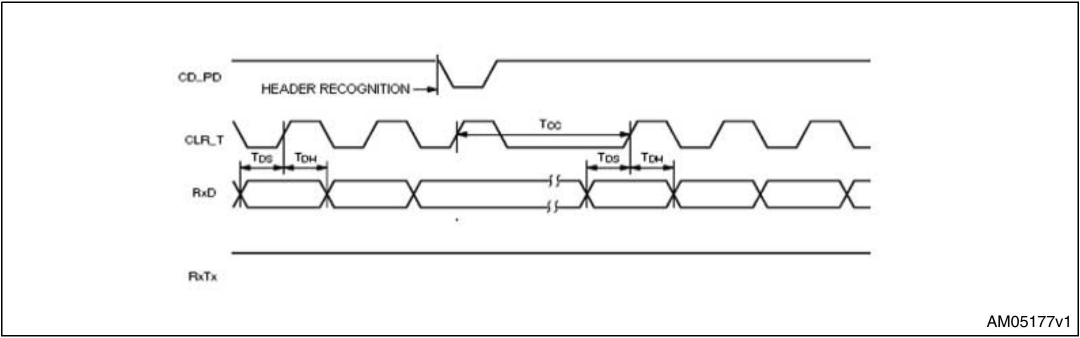
The PL_PLMSetRX reception routine is used for receiving data from the PLM, and accepts as parameters the PLM mode (register or data transmission) and the number of bytes to receive.

At the beginning of the routine it checks if the data to receive is the internal registers of the PLM or a datagram. In the second case, a flag indicates at the SPI interrupt routine level, that a preamble and a header must be received. This is not the case while receiving the internal registers.

As the PLM is configured to work in the control register extended mode, the CD_PD line is forced low for one period of CLR/T line when a header sent by another device is recognized. In this working mode, the CLR/T and RxD signal are always present, even if no header has been recognized, so it is necessary to synchronize the interrupt routine of the MCU SPI peripheral and the PLM SPI interface (which acts as a master device). For this purpose, the SPI_WaitCLKLow() function has been implemented.

Once the header is detected, the CD_PD line goes down for a clock time, and the CLR_T line is forced low for a time (TCC) variable between Tb and 2*Tb (where Tb = 1 / BAUDRATE). Once the line is low, the SPI is enabled, firstly clearing all the flags. With this mechanism, the data reception is perfectly synchronized with the clock generated continuously by the PLM (*Figure 12*).

Figure 12. PLM SPI synchronization



3.2.3 Data link APIs

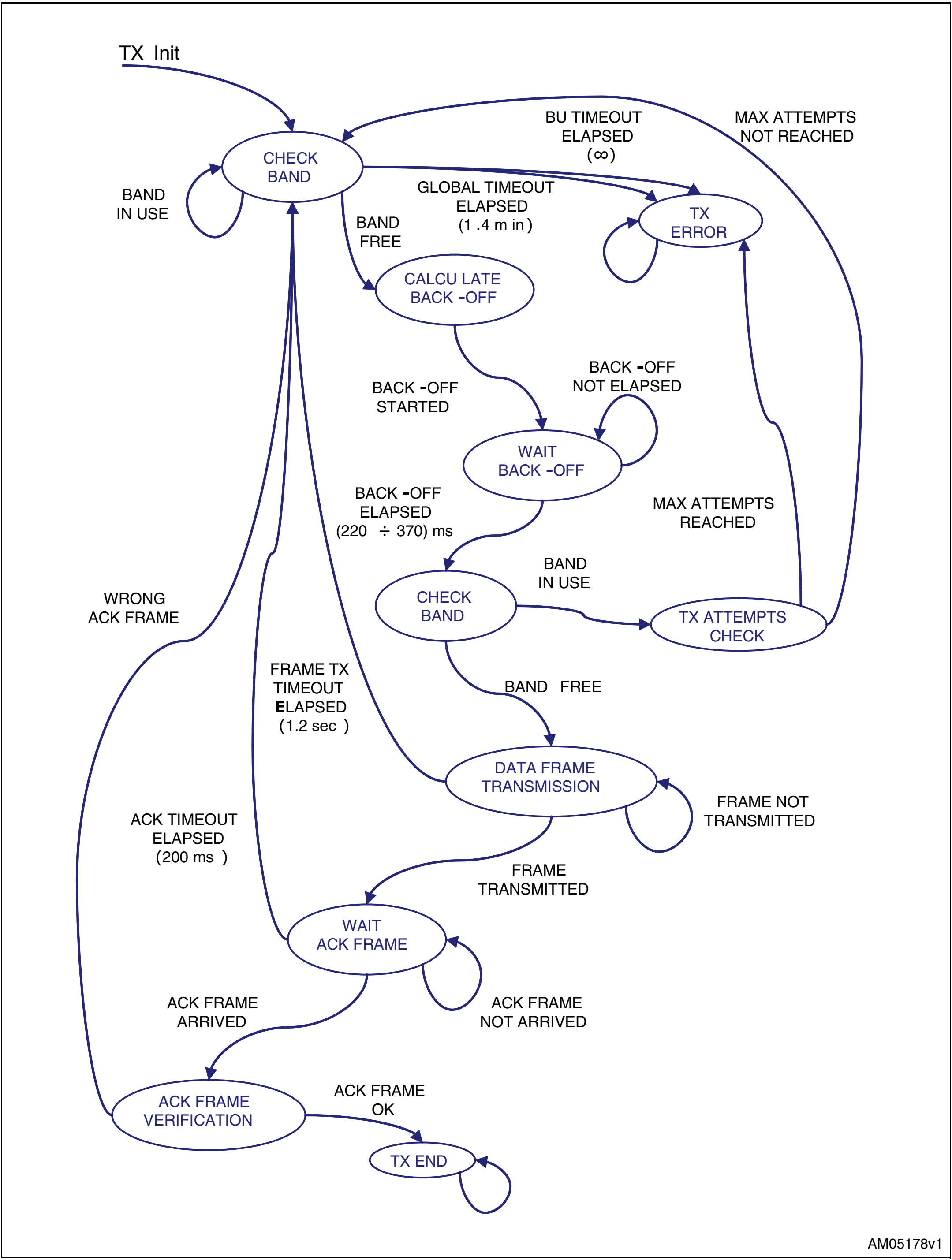
The data link layer assures data without communication errors to the upper level. This layer implements the state machine managing all the communication routines, data frame decoding, ACK management, timing management, backoff calculation, etc.

The main API of this layer is the LL_StackUpdate(), which implements the state machine of the communication. The SM is the same for both the transmission and the reception flow, as the communication is a half-duplex, therefore, there aren't transmissions and receptions to manage at the same time.

Of course, depending on the dongle type (node, repeater, receiver), the state machine starts from the Tx or the Rx side.

Figure 13 shows the transmission state machine.

Figure 13. Transmission state machine



Once a dongle starts a communication (the command is sent by the upper level), the state machine first resets all the timeout counters. Each timeout can have a different value, set by some #define directives, and are all managed by the routine:

```
LL_TimingManager(u32 tTconst, STK_SMIndex_t tTOStep)
```

In this routine the first parameter is the timeout value, and the second parameter is the state where the SM must jump once the timeout is elapsed. If the timeout value is set to "0", it is not managed.

After the timeout counters are reset, the band-in-use signal is tested, and the state loops until the band is free.

At this point, the next state calculates and waits for the backoff time. Once the backoff time has elapsed, the band is checked again on the next state. If the band is still in use, the backoff is calculated again and the state is set back, again waiting until backoff has elapsed.

If the band is free, data transmission is started by means of the transmission routine:

```
PL_PLMSetTX(PLM_DATA, NTW_NetworkFrame, NTW_NetworkFrame[7],
PLM_HEADER)
```

This function, which belongs to the physical layer, enables the SPI interrupt routine after adding some bytes to the frame, as already described. Once the SPI is enabled, the state waits for transmission completion, signaled by the PLM_Flag.PLM_F_TXOK flag, or timeout elapsing for the transmission set to 1.2 sec.

Once the transmission is completed, the next step waits for the ACK frame, which must be sent before the delay of 200 ms has elapsed. During this time, each ACK frame not addressed to the dongle is ignored.

The function used for receiving frames through the PLM modem is:

```
PL_PLMSetRX(PLM_DATA, ACK_TOTAL_LEN)
```

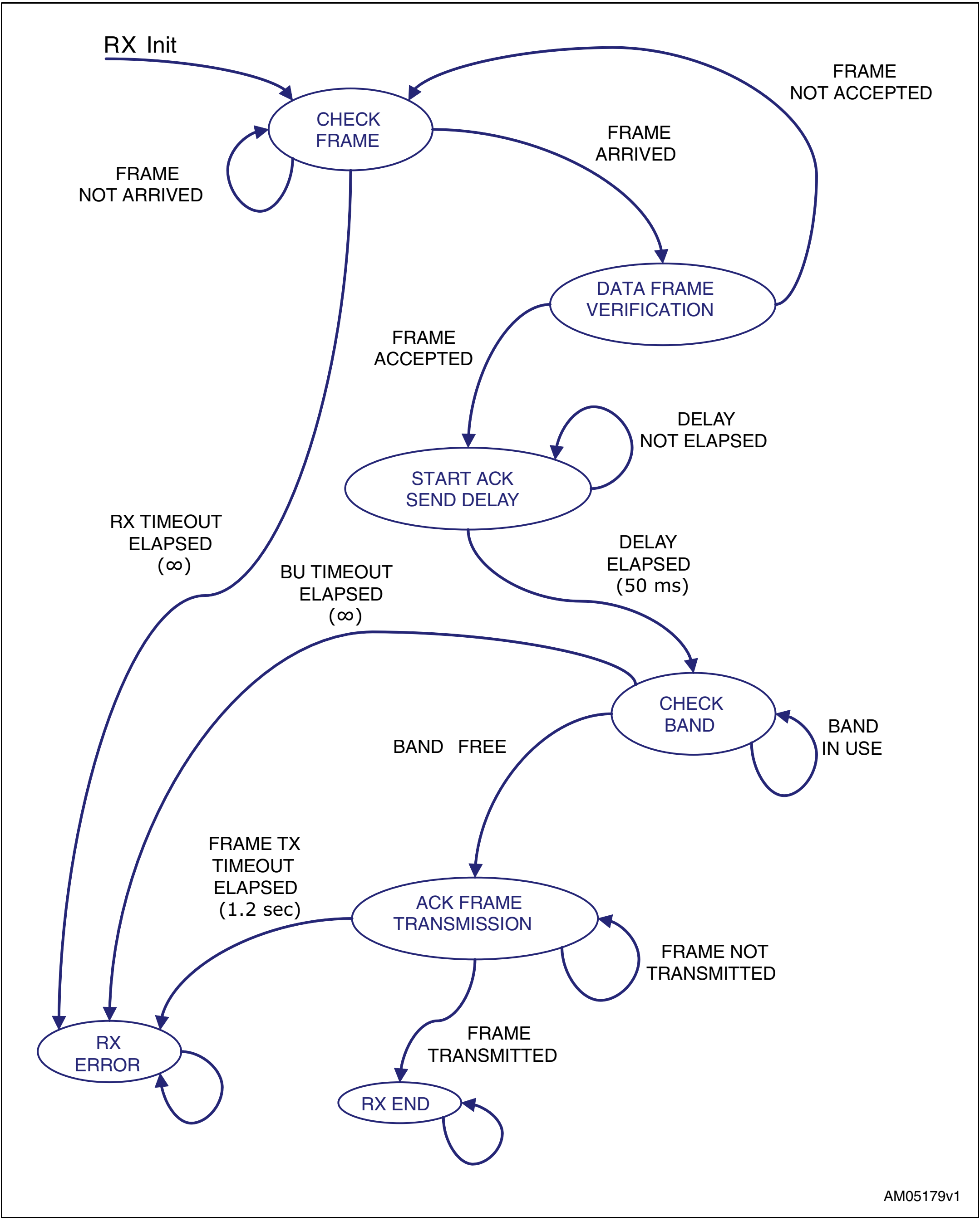
This function resets and initializes all the communication flags and indexes, the buffer counter, and enables the SPI peripheral, once the synchronization with the PLM clock (see *Figure 12*) has occurred, as described previously.

If the ACK frame is received, it must pass the acceptance criteria (postamble and FEC verification and eventually correction, a correct class value, correct CRC and sender address) and, if this is the case, the transmission phase is completed. If not, the transmission process is started again, within the global timeout timing.

In *Figure 14* the reception state machine is shown.

The initialization of the reception phase is done through the TL_NetworkIndication function in the STK_T file. Once a data frame is received, this function signals the event to the upper level (application level), which manages the data received. In the proposed model, the data frame is sent through the USART peripheral, as explained in the following paragraphs.

Figure 14. Reception state machine

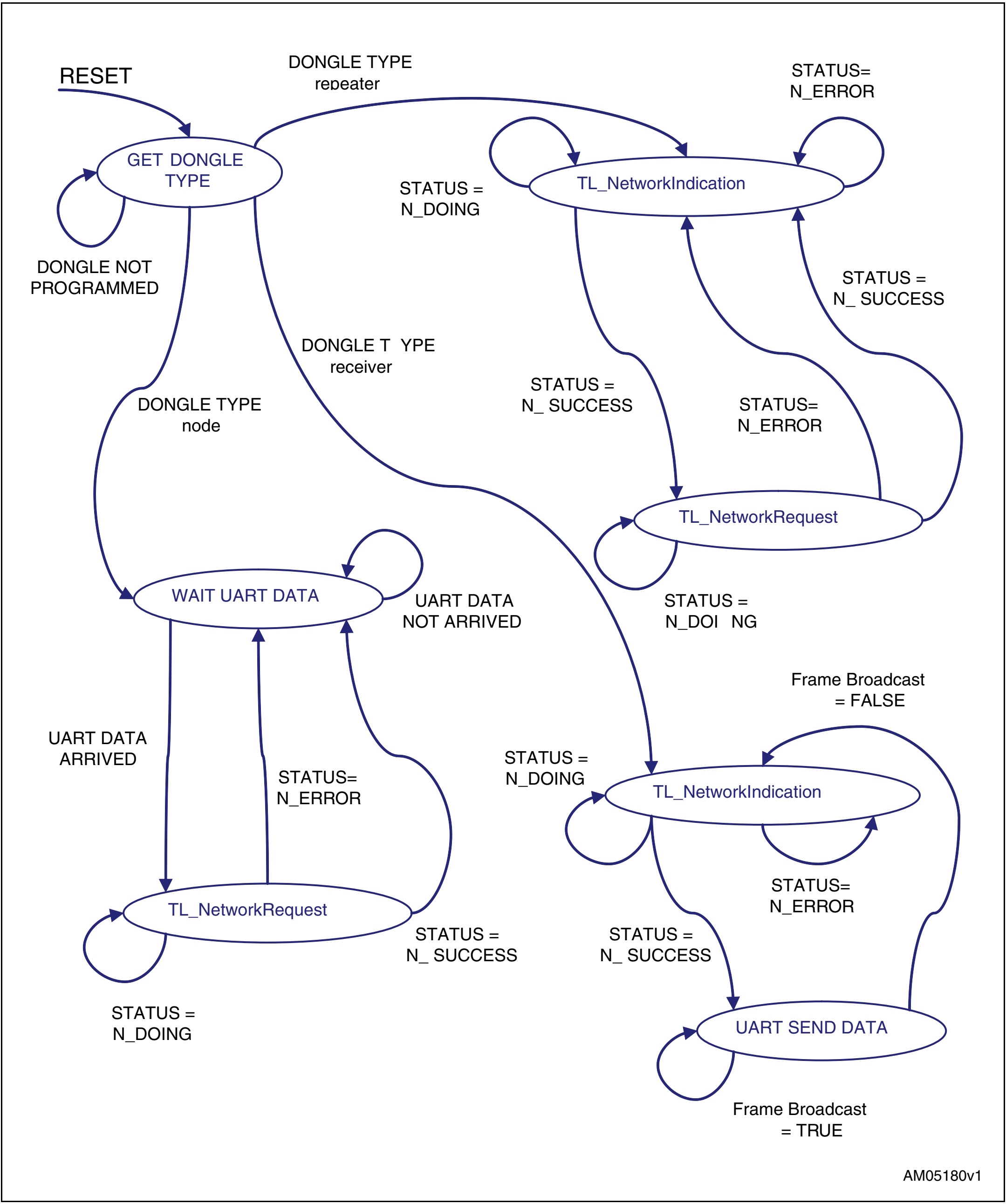


3.3 Application layer

The application layer implements the dongle working mode, and it is the upper level built on STK_T.

The main function of this layer is the APP_NetworkSM(). This function implements the state machine, shown on *Figure 15*.

Figure 15. Application state machine



Once the state machine is initialized, the dongle data is read in the Flash memory zone reserved for dongle data storage. If the dongle was never programmed, the state machine waits until the programming phase is completed (by means of the PLM ST PLM Dongle Programmer GUI and the USART interface).

The start address of the flash area reserved for the dongle information is 0x0800EA00, and it is organized as follow:

- 0x0800EA00: device type (receiver, repeater, node, NONE)
- 0x0800EA02: local ID
- 0x0800EA04: primary remote ID
- 0x0800EA06: secondary remote ID

If the device is programmed as a receiver or as a node, the IDLE state is set, and a data frame is awaited from the USART (in the case of a node) or from the PLM (in the case of a receiver).

Only during the power-on, if the device is programmed as a repeater, is a test frame sent to the primary remote ID (and in the case of failure of the max attempts to this address, to the secondary remote ID) in order to test the communication line between these two devices. If the dongle is programmed as a node, the test frame is sent immediately after the programming command is sent. In this case it is not possible to re-program the node before the communication test is completed.

In IDLE state, a node waits for a data transmission request (the data could originate from RS232 or be internally generated by pressing the pushbutton mounted on the dongle board). Once the data is ready to be transmitted, the green LED is switched on, the transmission state of the data link layer is set, and the data is passed to this layer. Once the communication is completed without errors, the IDLE state is set again, and the transmission state machine is re-initialized.

The success of the transmission is signaled by the green LED switching off. If any communication error occurs (timeout, max attempts reached, etc.) the red LED is switched on and the state machines are re-initialized waiting for another data frame from the RS232 port.

If the dongle is programmed as a repeater, the only information managed by the RS232 port is the programming frames. Each time a data frame is sent by a node to the repeater, the latter first sends an ACK frame to the sender, and the frame received is re-transmitted to its primary remote ID (and in the case of failure of the max attempts to this address, to the secondary remote ID). The LED signal is not different to the one used during frame transmission for a node.

3.4 Programming routines

The programming routines allow you to set up the working mode for a generic dongle, set the dongle type and store the primary and the secondary remote ID (the address of the target devices).

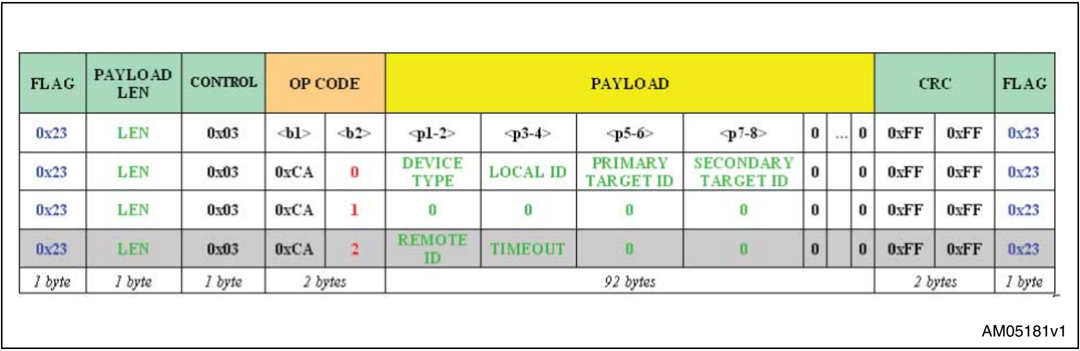
These routines are stored in the IAP.c file. The interface routine with the RS232 port is the following:

```
FrameType_t IAP_ProgrammingCommand(void)
```

In this routine the data received flag set in the USART interrupt routine is first tested (`bSTBFrameBroadcast`) and, if necessary (`bSTBFrameBroadcast = TRUE`), the frame is analyzed, in order to understand whether it is a programming frame or a data frame.

The programming frame structure is shown in *Figure 16*.

Figure 16. Programming frame structure



FLAG	PAYLOAD LEN	CONTROL	OP CODE		PAYLOAD							CRC		FLAG
0x23	LEN	0x03	<b1>	<b2>	<p1-2>	<p3-4>	<p5-6>	<p7-8>	0	...	0	0xFF	0xFF	0x23
0x23	LEN	0x03	0xCA	0	DEVICE TYPE	LOCAL ID	PRIMARY TARGET ID	SECONDARY TARGET ID	0		0	0xFF	0xFF	0x23
0x23	LEN	0x03	0xCA	1	0	0	0	0	0		0	0xFF	0xFF	0x23
0x23	LEN	0x03	0xCA	2	REMOTE ID	TIMEOUT	0	0	0		0	0xFF	0xFF	0x23
1 byte	*1 byte*	*1 byte*	*2 bytes*		*92 bytes*							*2 bytes*		*1 byte*

The frame length is fixed to 100 bytes (between two flags) but it may be reduced by acting on the LEN parameter.

Each programming command starts with 0xCA (parameter <b1>). The second byte (<b2>) can assume the values: 0, 1 or 2.

The command “0” is the command SET dongle data, and the parameters sent with the data frame are the device type (255=None, 1=Receiver, 2=Repeater, 3=Node), the local ID, the primary target ID and the secondary target ID. All the other bytes must be set to “0”.

The command “1” is the GET command, and it is used to obtain all the programming information from a generic dongle. The received frame is filled with the dongle type, primary and secondary target ID, and has the same format as the SET frame.

The last implemented command is “2” which is the test command. This command is used for testing a remote dongle where the address is given by the software GUI, passed as a parameter. The second parameter is the timeout, which must be awaited for the remote dongle response. This function may be used for a network discovery procedure, but it is not yet implemented in the current software release ver. 1.0.0

The `IAP_ProgrammingCommand` function returns the received frame type (if any):

- `TY_PROGRAMMING_FRAME_STORED`: The programming frame has been received and the dongle data has been stored
- `TY_PROGRAMMING_FRAME_DATA_READY`: The dongle data is ready to be sent through the RS232 port
- `TY_PROGRAMMING_FRAME_TEST_REMOTE_ID`: The remote dongle address has been set and is ready to be addressed
- `TY_STB_FRAME`: Data frame arrived
- `TY_NO_FRAMES`: No frame arrived

Each time a programming command is decoded and executed, the orange LED blinks three times and the corresponding action is performed.

The result of the programming operation, which is the parameter returned by the programming function, is directly assigned to the state machine variable, and the status is set according on the operation requested after the programming phase.

4 PC GUI software description

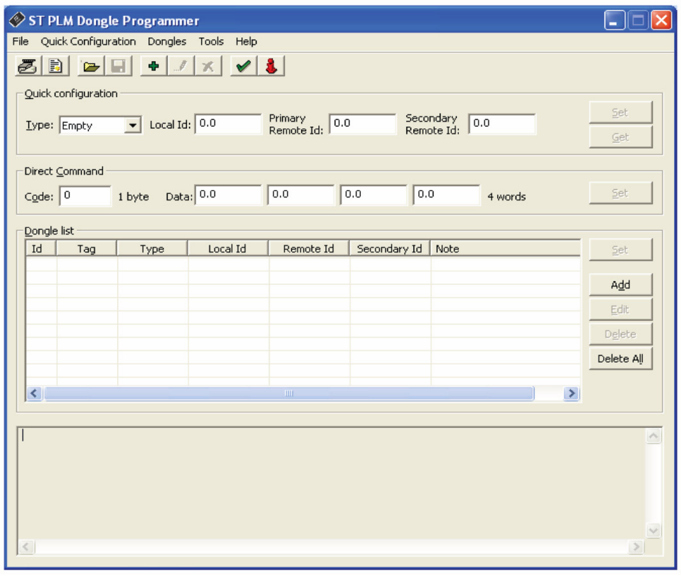
4.1 Introduction

The software used for interfacing the dongles with a PC are two: the ST PLM Frame Manager and the ST PLM Dongle Programmer. The first GUI is used for capturing the data frames addressed to the receiver device from a node, directly or by means of a repeater, through the RS232 port, or it can be used for sending a data frame with a user-defined payload. The second GUI is used to program the dongles to work as a receiver, a repeater or a node, setting the target and the backup address, and reading the information of any dongle.

4.2 ST PLM Dongle Programmer

Figure 17 shows the main interface of the GUI.

Figure 17. ST PLM Dongle Programmer main view



Quick configuration

Type: Empty Local Id: 0.0 Primary Remote Id: 0.0 Secondary Remote Id: 0.0

Direct Command

Code: 0 1 byte Data: 0.0 0.0 0.0 0.0 4 words

Dongle list

Id	Tag	Type	Local Id	Remote Id	Secondary Id	Note

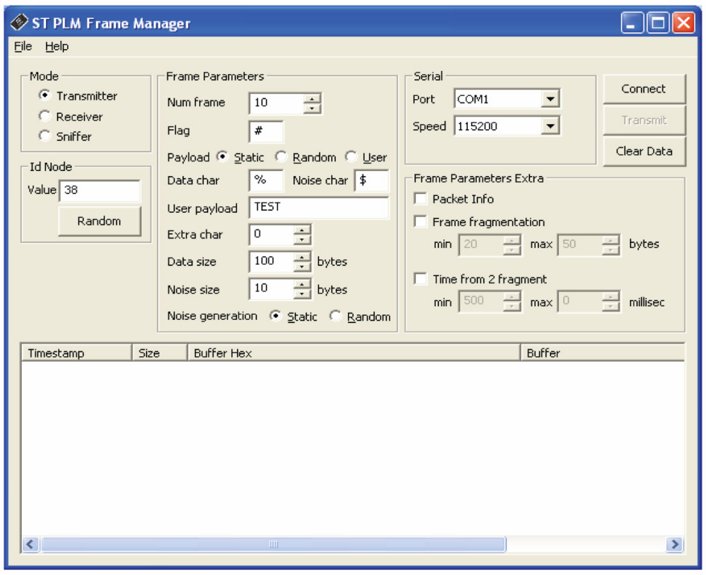
The quick configuration group allows the configuration of a single device at a time, while by using the dongle list area it is possible to firstly prepare logically the network, and programme each dongle, once the appropriate row in the grid row is selected.

By using the direct command group it is possible to send commands other than GET and SET that could be implemented on the firmware.

4.3 ST PLM Frame Manager

Figure 18 shows the main interface of the GUI.

Figure 18. ST PLM Frame Manager main view



Mode: [✓ Transmitter] Receiver Sniffer

Id Node: Value 38

Frame Parameters: Num frame 10; Flag #; Payload [✓ Static] Random User; Data char %; Noise char $; User payload TEST; Extra char 0; Data size 100 bytes; Noise size 10 bytes; Noise generation [✓ Static] Random

Serial: Port COM1; Speed 115200

Frame Parameters Extra: Packet Info; Frame fragmentation min 20 max 50 bytes; Time from 2 fragment min 500 max 0 millisec

Timestamp	Size	Buffer Hex	Buffer

AM05183v1

In the mode group it is possible to define if the interface must be used as a transmitter, a receiver or just a traffic sniffer on the RS232. In the frame parameters group it is therefore possible to set some transmission parameters, such as the flag that must be used in the transmission, the number of consecutive frames to send, the payload type (static tandom or defined by the user typing text in the User payload text box) and some extra char that is filtered by the firmware in the USART interrupt routine.

The data size and the noise generated can be set in the two dedicated boxes.

The serial group contains the RS232 COM port parameters (leave the baud rate at 115200bps), while the extra frame parameters allow the user to segment the frame, setting up a delay between packets, in order to see if the filtering algorithm recognizes a data frame in any case, avoiding any data loss.

In the message window it is possible to see the traffic of any incoming frame direct to the receiver, connecting it through the RS232 port.

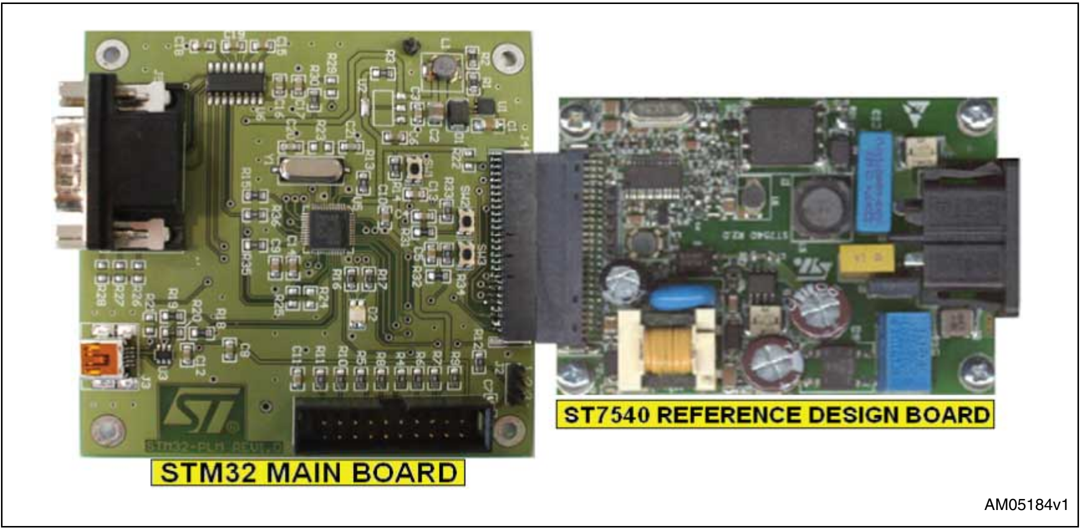
5 Hardware dongle description

There are two solutions which are useful to test the proposed application. The first is the use of the ST7540 reference design board and an STM32 prototype board, the second solution is the use of the STEVAL-IHP003V1 board which has the ST7540 and the STM32 microcontroller inside.

5.1 Dongle system board

The STM32 main board schematic is shown in *Section 8*. In *Figure 19* the two boards which make up the dongle system board are shown. The STM32-based board is a demonstration prototype, not orderable through ST.

Figure 19. Dongle system



5.1.1 The ST7540 reference design board

With the ST7540 reference design board (EVALST7540-1) it is possible to evaluate the ST7540 features, in particular its transmitting and receiving performances, through actual communication on the power line.

The ST7540 reference design is composed of three main sections:

- Power supply section; specifically tailored to match power line coupling requirements and to operate within a wide range of the input mains voltage
- Modem and crystal oscillator section
- Line coupling interface section

The coupling interface is designed to allow the ST7540 FSK transceiver to transmit and receive on the mains using 72 kHz carrier frequencies, within the European CENELEC standard A-band specified for automatic meter reading. The board is provided with an integrated and isolated power supply (SMPS) which supplies the power to the PLM and to the microcontroller.

The dongle needs only the two mains wires (line and neutral), which are used both for power supply and for data modulation and demodulation.

The system is provided with a band pass filter with the center frequency set to 132.5 kHz and integrates a power amplifier, which is able to drive low line impedance, and two linear regulators providing 5 V and 3.3 V. Please refer to the AN2451 application note for more details on filter calculations and considerations.

5.1.2 STM32 main board

The main board is based on an ARM-based 32-bit MCU STM32F103C8 microcontroller. It is provided by an RS232 plug, used for the dongle programming, for the node frame transmission, and for the output of the incoming frames of a receiver.

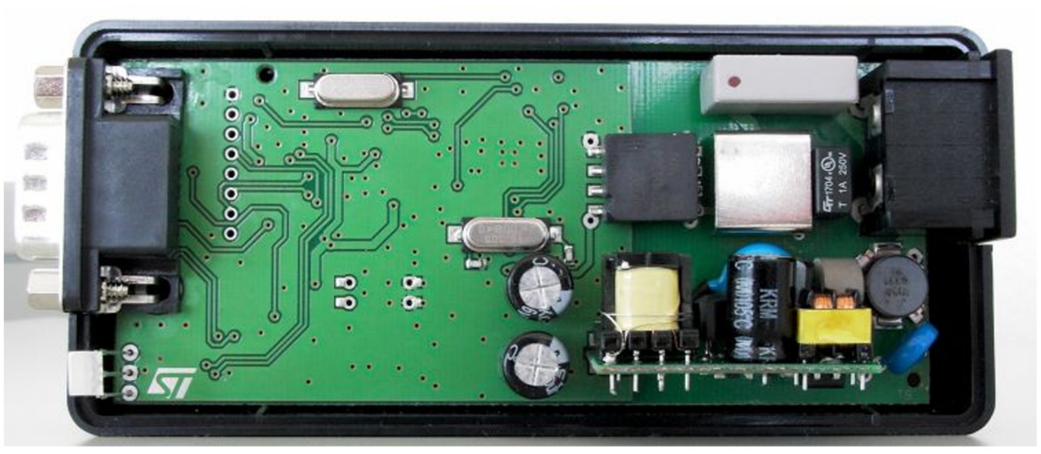
It also includes two general purpose pushbuttons, plus a third pushbutton to reset the MCU. For visual output, three colored LEDs exist for signaling purposes. Although there is a USB connector, it is not used for this application.

A 20-pin connector (JTAG) is used for microcontroller programming and debugging, by using the appropriate J-Link debugger/programmer dongle and the IAR software tool explained in the next section.

5.2 STEVAL-IHP003V1 board

As an alternative to the previous system, the STEVAL-IHP003V1 board can be used. This board integrates both the ST7540 reference design board and the STM32 main board. More details on this board are available at www.st.com.

Figure 20. STEVAL-IHP003V1



6 Getting started with the system

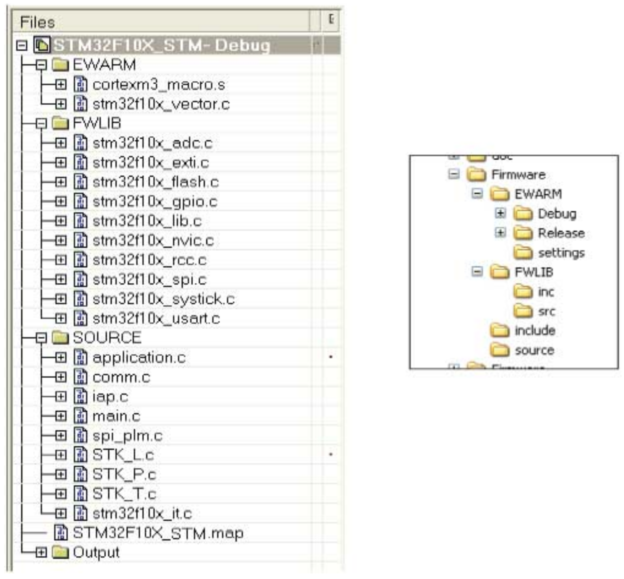
The steps needed for compiling the firmware and programming the dongle are explained in this section. If the compiler and the J-Link programmer are not available, it is possible to use the STEVAL-IHP003V1 board and a dedicated software, available from www.st.com (explained further in this section), which programmes the board with the compiled binary file of this example provided with the source code.

6.1 Configuring the IAR tool for building, debugging, and programming the software

The tool used for writing the firmware and debugging the application is the IAR embedded Workbench ver. 4.42 A. The workspace is created using the IAR embedded Workbench 4.42 A, using the ARM-based 32-bit MCU STM32F101xx and STM32F103C8 firmware library (ver. 1.0) in C language.

The tree structure of the project is organized by separating and grouping the source files with the header files, both for the project files and the library files, as shown in *Figure 21*.

Figure 21. Workspace structure



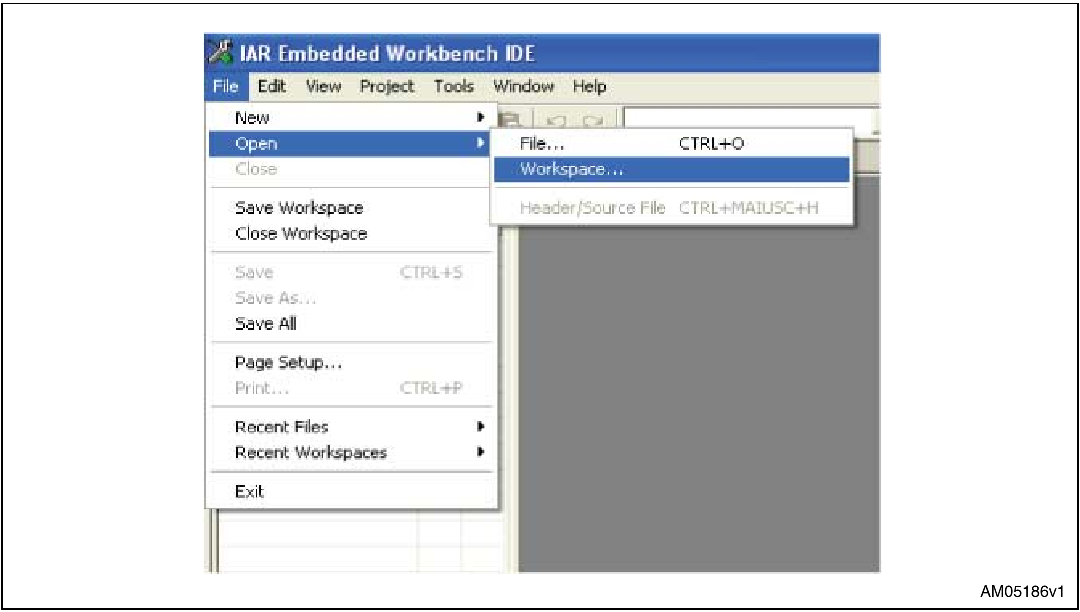
```
Files
⊟ STM32F10X_STM - Debug
  ├⊟ EWARM
  │ ├⊞ cortexm3_macro.s
  │ └⊞ stm32f10x_vector.c
  ├⊟ FWLIB
  │ ├⊞ stm32f10x_adc.c
  │ ├⊞ stm32f10x_exti.c
  │ ├⊞ stm32f10x_flash.c
  │ ├⊞ stm32f10x_gpio.c
  │ ├⊞ stm32f10x_lib.c
  │ ├⊞ stm32f10x_nvic.c
  │ ├⊞ stm32f10x_rcc.c
  │ ├⊞ stm32f10x_spi.c
  │ ├⊞ stm32f10x_systick.c
  │ └⊞ stm32f10x_usart.c
  ├⊟ SOURCE
  │ ├⊞ application.c
  │ ├⊞ comm.c
  │ ├⊞ iap.c
  │ ├⊞ main.c
  │ ├⊞ spi_plm.c
  │ ├⊞ STK_L.c
  │ ├⊞ STK_P.c
  │ ├⊞ STK_T.c
  │ └⊞ stm32f10x_it.c
  ├── STM32F10X_STM.map
  └⊞ Output
```

```
⊟ Firmware
  ⊟ EWARM
    ⊞ Debug
    ⊞ Release
      settings
  ⊟ FWLIB
      inc
      src
    include
    source
```

AM05185v1

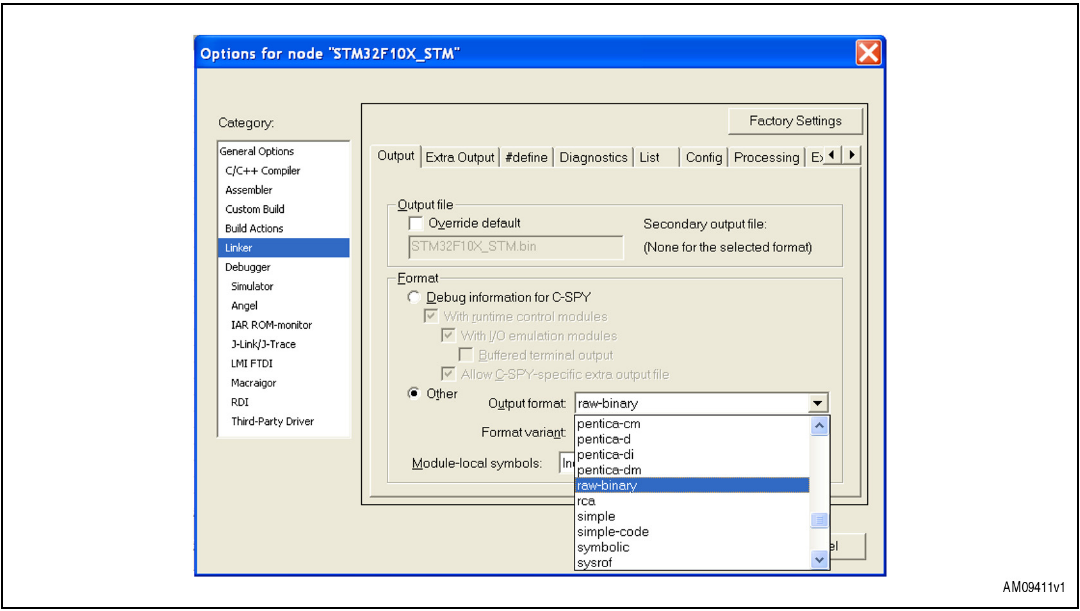
In order to load the project, click on file\open\workspace and in the window that appears select \project folder\EWARM and the file STM32F10X_STM.eww (*Figure 22*).

Figure 22. IAR embedded Workbench main window



On the main node, where the program name (STM32F10X_STM in bold) is shown, located in the files window, right click and select options. In the opened window select the linker item, and in the format group box select the other option. The output format list box is enabled, and select the item raw-binary from the list (*Figure 23*).

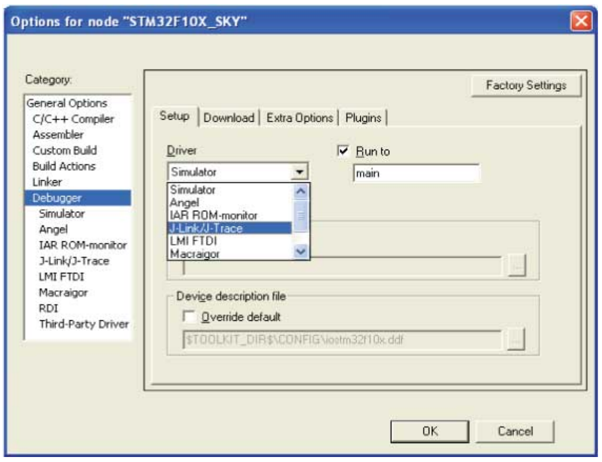
Figure 23. Select binary output



6.2 Programming the dongle system board

On the main node, where the program name (STM32F10X_STM in bold) is shown, located in the files window, right click and select options. In the opened window select the debugger item in the category list box, and in the driver list box select the proper debugging tool, then press the OK button (*Figure 24*). The J-Link dongle is used in the proposed example. Press the Make icon or click project\rebuild all. Any error or warning should appear once the compiling is 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, following the scheme shown in *Section 5: Hardware dongle description*. 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. If you want 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 24. IAR embedded Workbench debugger options


Options for node "STM32F10X_SKY"

Category: Debugger

Driver: Simulator

J-Link/J-Trace

[✓] Run to: main

Device description file

[ ] Override default

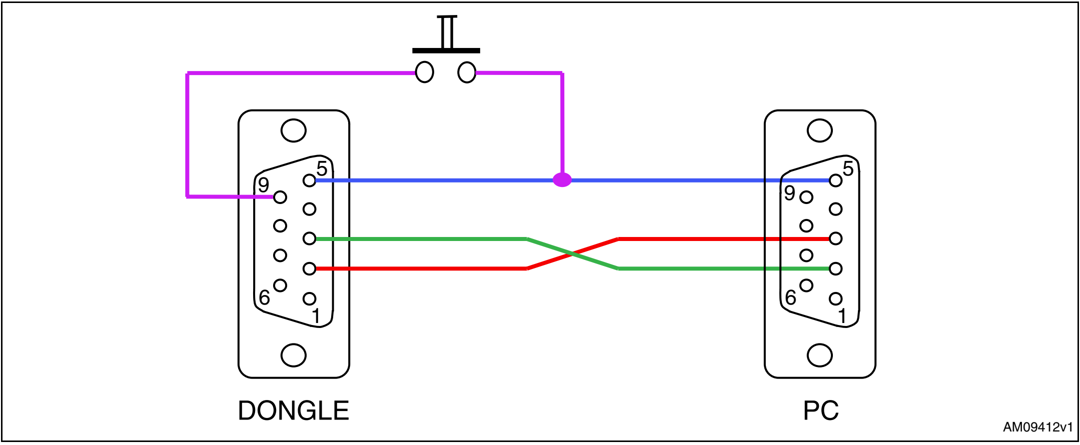
$TOOLKIT_DIR$\CONFIG\iostm32f10x.ddf

AM05187v1

6.3 Programming the STEVAL-IHP003V1 board

Connect a null modem cable between an RS232 connector of the PC and the RS232 connector of the STEVAL-IHP003V1 board, which should have the possibility to connect pin 9 to ground during the power on; for instance, by placing a pushbutton between pin 5 and pin 9 of the RS232 connector in the dongle side (*Figure 25*). This is used to put the dongle into programming mode, allowing the firmware updating by means of the STM32 boot loader, using the serial COM port instead of a J-Link programmer.

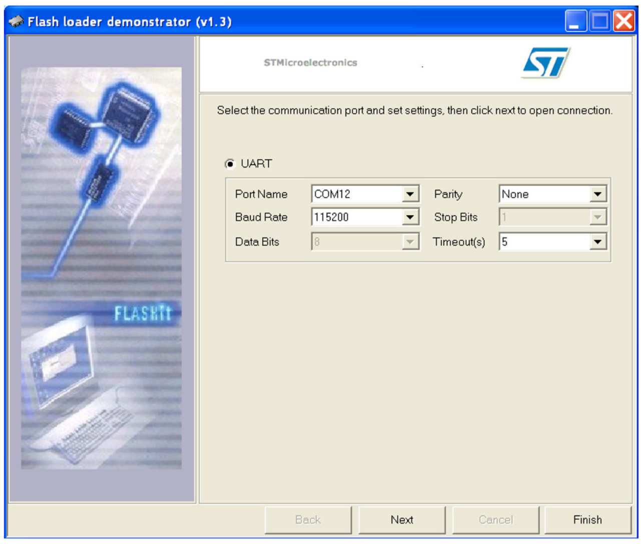
Figure 25. STEVAL-IHP003V1 programming cable



After grounding pin 9, power on the dongle by connecting the power cable. The boot loader of the STM32 starts and the microcontroller is put into programming mode. Install and run the software Flash loader demonstrator for STM32, which can be downloaded from www.st.com, following the instructions referred to in the UM0462 user manual.

Select the COM port where the cable is connected (it should appear among the available COM ports in the port name list box), and set the baud rate to 115200 bps and the parity to none, leaving the timeout at 5 seconds, as default (*Figure 26*). By clicking the next button, if the connection is established between the dongle and the software, the green light should appear with the message “Target is readable”. Click the next button again and select the STM32F10xxCxx as the target device, then click the next button once again.

Figure 26. Flash loader demonstrator main window



Select the communication port and set settings, then click next to open connection.

UART

Port Name	COM12	Parity	None
Baud Rate	115200	Stop Bits	1
Data Bits	8	Timeout(s)	5

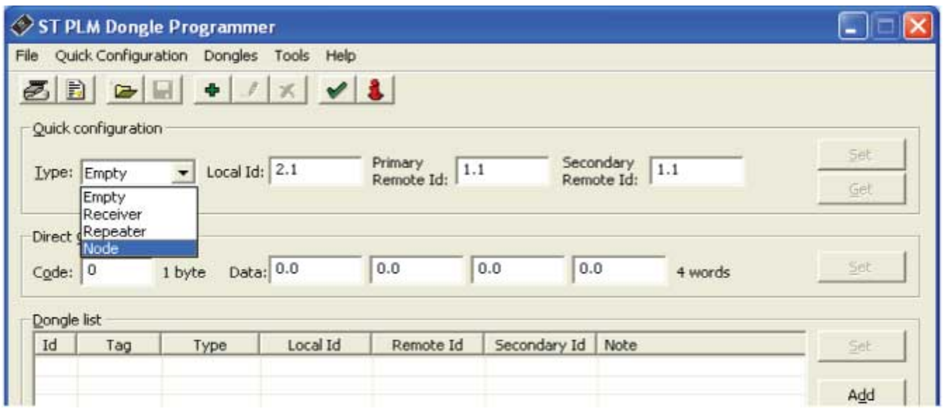
Select the download to device option, and load the binary file of the firmware (created selecting the raw-binary format in the linker item, as previously explained in *Section 6.1*) by clicking on the folder button, located on the right side of the download from file text box. The STM32F10X_STM.bin binary file is located in the \EWARM\debug\exe folder. In that folder the STM32F10X_STM.bin file of the firmware, explained in this application note, is already provided. Confirm that the “Verify After Download” check box is selected. Click the next button for the last time, and the firmware loads into the STM32 microcontroller. Remove the power cable and disconnect pin 9 from ground.

6.4 Customizing the system

Download the firmware in a dongle, as previously explained. Connect the power cable and a crossed RS232 cable between the serial port of the PC and the dongle. Once installed, launch the program ST PLM Dongle Programmer and click the configuration icon or file\configure. Select the right COM port from the list box and set a baud rate of 115200, then select apply. In the quick configuration window, select the required working mode of the dongle. As each node transmits a test frame once programmed, the correct programming order should be the receiver first, then the repeater and, finally, the node. Select the working mode in the type list box and set the local ID, which is the device address and must be unique to the network where the dongle is installed.

Fill in the two primary and secondary remote ID fields, which are the addresses of the devices that must be addressed. If the device is a receiver, these last two fields are inapplicable (*Figure 27*).

Figure 27. Dongle type programming



AM05188v1

Press the connect icon or click file\connect in order to open the COM port, and press the Set button. The data is sent to the dongle which signals correct data reception through the blinking of the orange LED.

It is possible to prepare all the data to send to all the dongles that must be installed in the network. By pressing the Add button, a dongle add window appears, and in addition to the fields described above, it is possible to add some notes.

Once the list is complete, connecting each dongle to the RS232 port and selecting the corresponding configuration row in the dongle list window, it is possible to program the device by pressing the Set button in the same window.

In the direct command window it is possible to write directly the command code and send it to the dongle. This feature can be used if any further commands are implemented in the firmware.

In order to send and receive a user-defined data frame and see it on a PC, run the ST PLM Frame Manager.

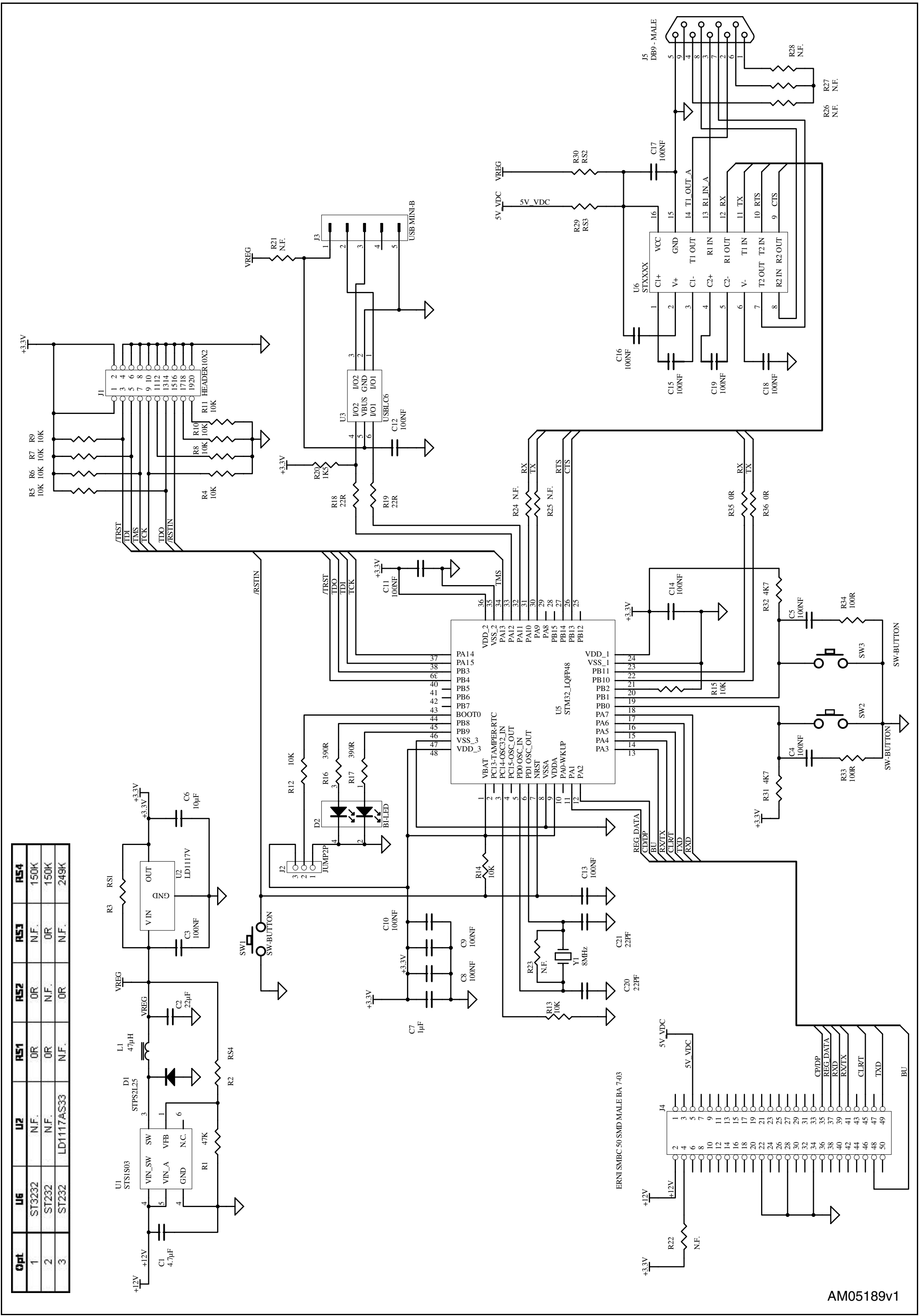
7 Conclusion

This project is an example of how it is possible to easily create a simple static network by means of dongles connected to the power line. The implemented solution is a robust protocol that manages all the data allowing multimaster configuration and a concentrator (receiver) that collects all the data. It is always possible to only configure nodes, without using any receiver, statically addressed by each other, or modifying the firmware in order to create broadcast commands or a dynamic network solution.

Each node can also be interfaced with an external device, through the RS232 port or by using other general purpose IO pins of the microcontroller, through a simple hardware modification, in all applications like power management, heating or cooling system management, lighting control, alarm systems, and appliance remote control, where there is the need to not use additional wiring which is different to the electrical network in homes and buildings.

8 Schematic

Figure 28. Schematic



Opt	U6	U2	RS1	RS2	RS3	RS4
1	ST3232	N.F.	0R	0R	N.F.	150K
2	ST232	N.F.	0R	N.F.	0R	150K
3	ST232	LD1117AS33	N.F.	0R	N.F.	249K

9 References

1. ST7540; FSK power line transceiver datasheets (2006)
2. Home and building electronic systems (HBES) - Eur. Family of Std. 50090 (2005)
3. STM32F103C8 microcontroller datasheets (2008)
4. UM0427
5. AN2451
6. UM0239

10 Revision history

Table 1. Document revision history

Date	Revision	Changes
06-Sep-2010	1	Initial release.
23-Feb-2011	2	*Section 5* e *6* modified