



AN4018

Application note

Data link communication protocol for the ST7580 power line networking SoC

By Vincenzo Mormina

Introduction

This document describes the data link communication protocol and the services provided to the application layer.

This communication protocol is designed from the ST7580 power line modem devices.

Contents

1	Overview	5
2	Firmware architecture	6
3	Framing	8
4	Service description	10
4.1	System initialization services	10
4.1.1	Data link initialization	10
4.1.2	PLM modem channel frequency configuration	11
4.2	System configuration services	11
4.2.1	PLM modem configuration	11
4.2.2	Set MAC Address	12
4.2.3	Get MAC Address	13
4.3	Data link services	13
4.3.1	Allow join request	13
4.3.2	Join Indication	13
4.3.3	Data request	14
4.3.4	Data indication	14
4.4	MIB management services	15
4.4.1	MIB Reading	15
4.4.2	MIB Writing	15
4.4.3	MIB Erasing	15
4.4.4	MIB parameters	16
Appendix A	Join procedure	17
Appendix B	STM32Fx family modem configuration parameter description	18
Appendix C	Example to configure the communication protocol	19
C.1	Configure the STM32Fx family serial layer	19
C.2	Configure a run the data link layer	20
5	References	21

6 **Revision history** 22

List of figures

Figure 1. Firmware architecture 6

Figure 2. Firmware components and files 7

Figure 3. PSK frame 8

Figure 4. Data link frame 8

Figure 5. Join procedure diagram 17

Figure 6. Message sequences 17



1 Overview

The ST7580 data link communication protocol provides the services to build and manage a “master multi-slave” power line network.

Two different modes are described in this document:

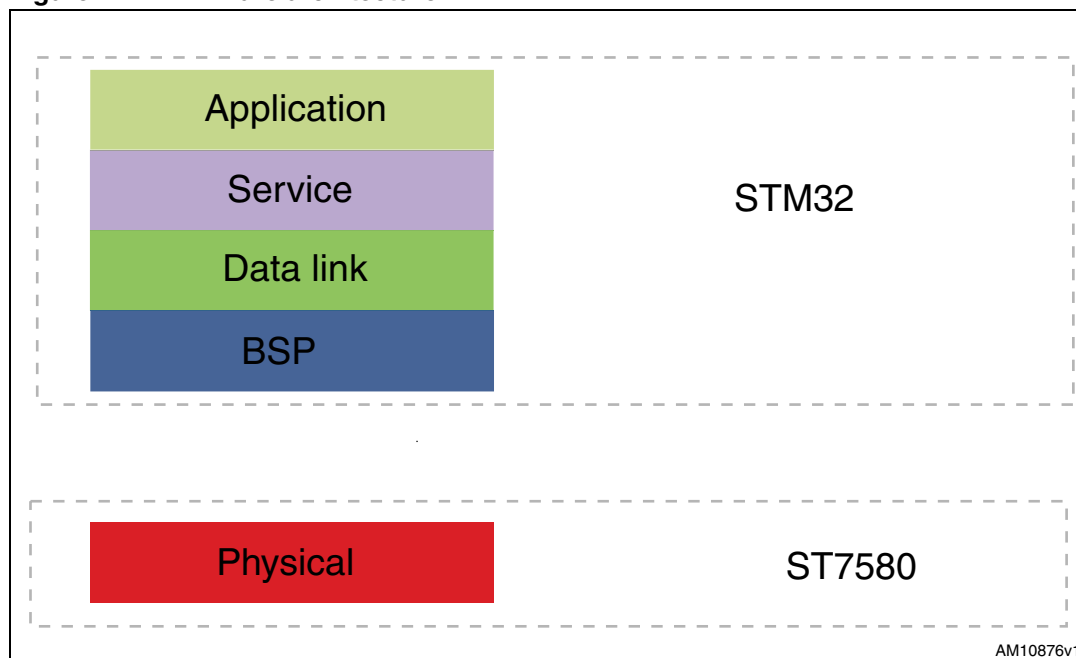
- Master-multi-slave mode:
 - In a master-multi-slave network, one node, and only one node, must be configured as a “master” while the other nodes must be configured as “slaves”. Each slave needs a join procedure with the master before starting communication; [Appendix A](#) describes the join procedure.
- Simple node mode:
 - In simple node mode all the nodes must be configured as “master”; in this case the nodes do not need a join procedure and they can all communicate with any other node.

The protocol has been specified for the STM32Fx family microcontroller. It utilizes a serial communication (UART) with the ST7580 power line device.

2 Firmware architecture

Figure 1 shows the firmware architecture:

Figure 1. Firmware architecture

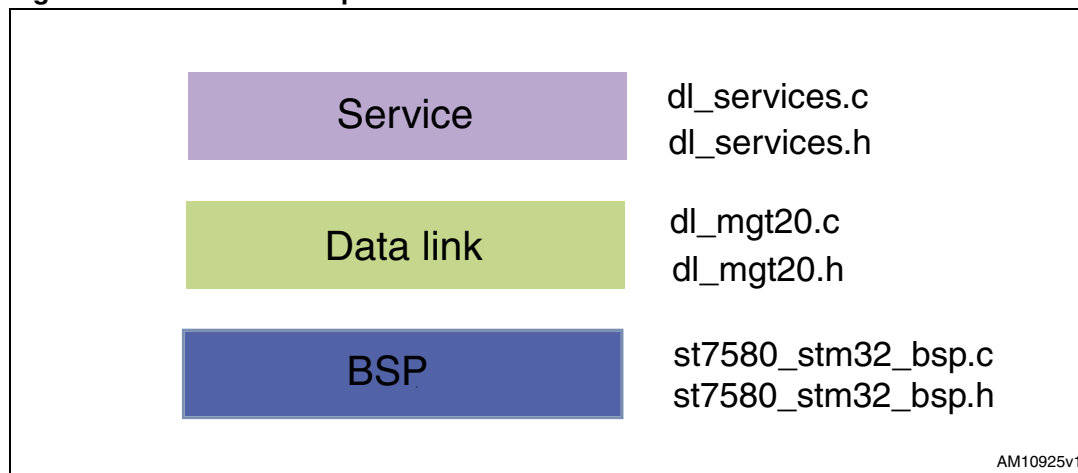


The physical layer of the ST7580 provides a basic MAC service through a serial communication interface.

- Serial layer:
 - The “BSP” layer manages the communication interfaces from the physical layer (ST7580) and the data link layer. Moreover, this manages the anti-collision repeating message.
- Data link layer:
 - The “data link” layer manages the framing sent and received, and the power line modem configuration.
- Service layer:
 - The “service” layer provides the data link services to the application layer; it allows messages and commands to be sent and received.
- Application layer:
 - The “application” layer can be customized by the user; in this section the user must design the application to create, manage and maintain the network.

Figure 2 shows the firmware components and their files:

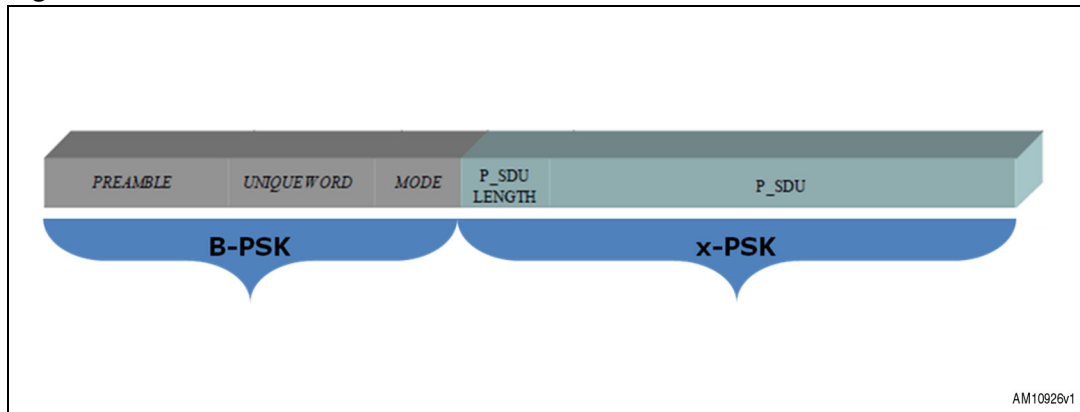
Figure 2. Firmware components and files



3 Framing

The communication modulation is an X-PSK, and the physical frame is the typical PSK frame, shown in [Figure 3](#).

Figure 3. PSK frame

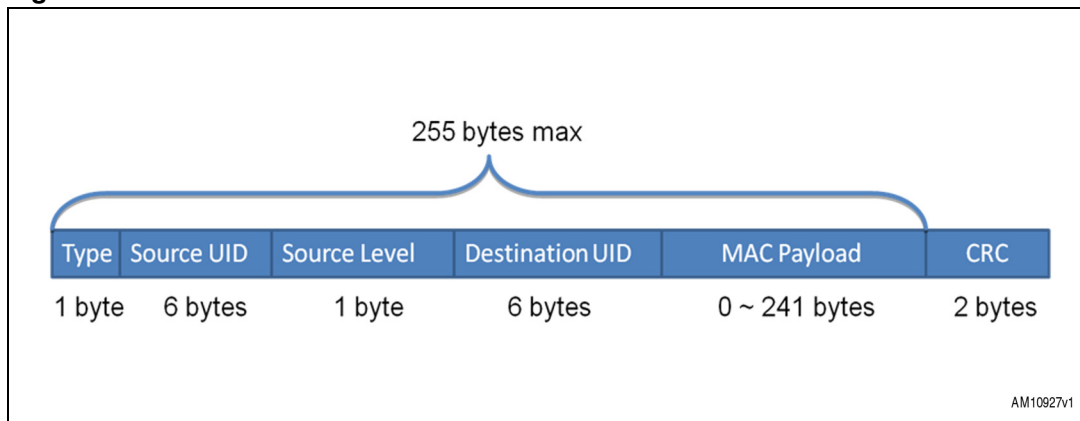


where:

- Preamble length: 32 bits
- Unique word: 16 bits
- Mode: 8 bits
- P_SDU length: 8 bits
- P_SDU: max. 255 bytes

The P_SDU contains the data link frame which has the structure shown in [Figure 4](#).

Figure 4. Data link frame



where:

- “Type” defines the type of frame, which can be:
 - AllowJoin
 - JoinReq
 - JoinAck

- Leave (not used in this protocol release)
- LeaveAck (not used in this protocol release)
- Data
- “Source UID” and “Destination UID” is the MAC address for sending and receiving respectively
- “Source level”, not used in this protocol release
- “MAC payload” is the data
- “CRC” depends on the ST7580, for more details see the ST7580 datasheet.

4 Service description

4.1 System initialization services

These services provide functions to initialize the data link layer and to configure and manage the modulation frequency of the power line modem.

4.1.1 Data link initialization

```
void DL_Init(u8 DLType) .
```

Description:

This function initializes the data link layer, with default parameters; the default parameters are:

- Frequency: 86000 Hz, 72000 Hz
- Rx mode: high channel only
- Rx H/L channel modulation: all PSK modulation
- Current control: enabled
- Gain: 21
- Zero crossing delay: 0
- PSK preamble length: 32 bits
- FSK baud rate: 2400 bps
- FSK deviation: 1
- FSK preamble length: 32 bit
- FSK unique word length: 16 bit
- FSK unique word: 0x9B58.

Input parameters:

- `DLType`: this parameter defines the data link type, master, slave or simple node; the value accepted is: "DL_MASTER", "DL_SLAVE" or "DL_SIMPLE_NODE".

Output parameters:

None.

Return value:

None.

Note:

This function must be called during the system startup.

4.1.2 PLM modem channel frequency configuration

```
void DL_FrequenciesConfig(u32 HChannel, u32 LChannel, FREQ_TYPE FreqType)
```

Description:

This service configures the PLM ST7580 modem frequency channel.

The use of this service has a lasting effect, in fact, it changes the default configuration parameters of the frequency channel. If this service is used, the value of the frequency channel is maintained for all successive system activations.

Input parameters:

- HChannel: Frequency channel to configure the high channel, for the accepted value see the ST7580 datasheet.
- LChannel: Frequency channel to configure the low channel, for the accepted value see the ST7580 datasheet.
- FreqType: Type of change, valid values "PERMANENT", "NOT_PERMANENT".

Output parameters:

None.

Return value:

None.

Note:

This service can be used to change the default frequency channel in the configuration parameters; remember that, changing the frequency channel, the values of frequencies set must be maintained until an ST7580 reset.

If the parameter "FreqType" is "PERMANENT", the new modem frequency value is registered in the RAM and in the Flash memory and this is the new default value.

If the parameter "FreqType" is "NOT_PERMANENT", the new modem frequency value is registered only in the RAM memory; at reset this modem frequency value is lost and the system reloads the old default value.

4.2 System configuration services

These services provide the functions to configure the power line modem and manage the MAC address of the system.

4.2.1 PLM modem configuration

```
void DL_ModemConfig(MOD_ConfigTypeDef* ModConfParam, CONF_TYPE ConfType)
```

Description:

This service is used to configure the modem with a different parameter with respect to the default values. This service does not have a lasting effect; the effect of this service is valid while a reset occurs or a new startup is performed.

Input parameters:

- **ModConfParam**: pointer to “MOD_ConfigTypeDef” struct, this includes all Modem configuration parameters, this struct, all parameters and the valid parameter values are described in the [Appendix B](#).
- **ConfType**: type of change, valid values “PERMANENT”, “NOT_PERMANENT”.

Output parameters:

None.

Return value:

None.

Note:

This service is not a substitute for DI_Init. In fact, the DI_Init must be performed at the system initialization layer at all times.

If the parameter “ConfType” is “PERMANENT”, the new modem configuration value is registered in the RAM and in the Flash memory and this is the new default value.

If the parameter “ConfType” is “NOT_PERMANENT”, the new modem configuration value is registered only in the RAM memory; at reset this modem configuration value is lost and the system reloads the old default value.

4.2.2 Set MAC Address

```
void DL_AddressSet(u8* NewAddress, CH_ADDR_TYPE AddType)
```

Description:

This service is used to set a new MAC address. At the first startup, the system gets a UID MAC address, this is the UID of the microcontroller. This service can change this address permanently or temporarily. If the service is used “temporarily” at the reset, the UID MAC address is reloaded; if the service is used as “permanent”, the MAC address is changed permanently.

Input parameters:

- **NewAddress**: pointer to new address, (6-byte array).
- **AddType**: type of change, valid value “PERMANENT”, “NOT_PERMANENT”, “AddType”.

Output parameters:

None.

Return value:

None.

Note: If the parameter “AddType” is “PERMANENT”, the new address is registered in the RAM and in the Flash memory, and this is the new default value. If the parameter “AddType” is “NOT_PERMANENT”, the new address is registered only in the RAM memory; at reset this address is lost and the system reloads the old default address.

4.2.3 Get MAC Address

```
void DL_AddressGet(u8* Address)
```

Description:

This service is use to read the current MAC address.

Input parameters:

- Address: pointer to actual address, (6-byte array).

Output parameters:

None.

Return value:

None.

4.3 Data link services

These services provide the functions for sending and receiving messages and for managing the network joining.

4.3.1 Allow join request

```
s8 DL_AllowJoinRequest(u8* DAddress, u8* Data, u8 Dlen).
```

Description:

This service is used by the application layer to perform an “allow join request” to another node.

Input parameters:

- DAddress: pointer to MAC address by destination node
- Data: pointer to data collection to send at the node (free application data)
- Dlen: length of data, number of bytes.

Output parameters:

None.

Return value:

Data link error code value.

4.3.2 Join Indication

```
u8* DL_JoinIndication(void).
```

Description:

This service reports to the application layer if a join indication occurs.

Input parameters:

None.

Output parameters:

None.

Return value:

Join indication identifier.

4.3.3 Data request

```
s8 DL_DataRequest(u8* DAddress, u8 DLen, u8* Data).
```

Description:

This service is used by the application layer to send a data collection to another node.

Input parameters:

- DAddress: pointer to MAC address by destination node.
- Data: pointer to data collection to send to the node.
- DLen: length of data, number of bytes.

Output parameters:

None.

Return value:

Data link error code value.

4.3.4 Data indication

```
u8* DL_DataIndication(u8* SAddress, u8* DLen, u8* DiParam).
```

Description:

This service reports to the application layer if a data indication occurs.

Input parameters:

None.

Output parameters:

- SAddress: pointer to MAC address by source node.
- DLen: length of data payload, number of bytes.
- DiParam: data indication parameters, 4 bytes (see the UM0932 user manual).

Return value:

Pointer to data collection (payload).

4.4 MIB management services

These services provide functions to read, write or erase the MIB (management information base) of the power line modem, for object details, see [Table 1](#) below.

For more information about the MIB and their management, please see the ST7580 datasheet.

4.4.1 MIB Reading

```
s8 DL_MIB_ReadRequest (MIB_INDEX_OBJ_T MIBIndex, u8* Data) .
```

Description:

This service is used by the application layer to read the MIB object information.

Input parameters:

- MIBIndex: index of MIB object, (see [Table 1](#)).

Output parameters:

- Data: pointer to MIB data object (the length depends on the object, see the ST7580 datasheet).

Return value:

Status or error value, (see [Table 2](#)).

4.4.2 MIB Writing

```
s8 DL_MIB_WriteRequest (MIB_INDEX_OBJ_T MIBIndex, u8* Data) .
```

Description:

This service is used by the application layer to write new data to the MIB object.

It is possible to use this service only for the MIB object with the following index: 0x00, 0x01, 0x02, 0x09.

Input parameters:

- MIBIndex: index of MIB object, (see [Table 1](#)).
- Data: pointer to MIB data object (the length depends on the object, see the ST7580 datasheet).

Output parameters:

None.

Return value:

Status or error value, (see [Table 2](#)).

4.4.3 MIB Erasing

```
s8 DL_MIB_Erase (MIB_INDEX_OBJ_T MIBIndex) .
```

Description:

This service is used by the application layer to erase the MIB object information.

It is possible to use this service only for the MIB object with the following index: 0x06, 0x07, 0x08.

Input parameters:

- MIBindex: index of MIB object, (see [Table 1](#)).

Output parameters:

None.

Return value:

Status or error value, (see [Table 2](#)).

4.4.4 MIB parameters

Table 1. MIB object index

Index	Name	Length (byte)	R/W/E
00h	Modem configuration	1	R/W
01h	PHY configuration	14	R/W
02h	SS key	16	R/W
03h	Reserved	1	R
04h	Last data indication	4	R
05h	Last TX confirm	5	R
06h	PHY_Data	10	R/E
07h	DL_Data	12	R/E
08h	SS_Data	10	R/E
09h	Host interface timeout	3	R/W
0Ah	Firmware version	4	R

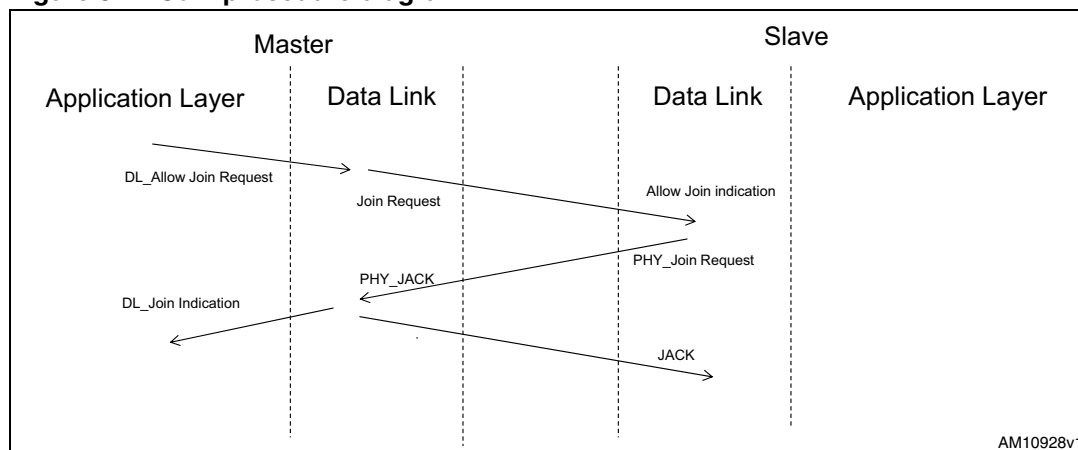
Table 2. Services return values

Value	Description
DL_BUSY	Data link is busy with another process, the service can't be performed
DL_WAIT	Wait state, the service is in execution phase
DL_TIMEOUT	This a an error timeout, the service failed due to timeout
DL_ERROR	Generic error. Service failed
DL_OK	Service processed correctly

Appendix A Join procedure

Figure 5 shows the join procedure diagram.

Figure 5. Join procedure diagram



The actions performed by the master are:

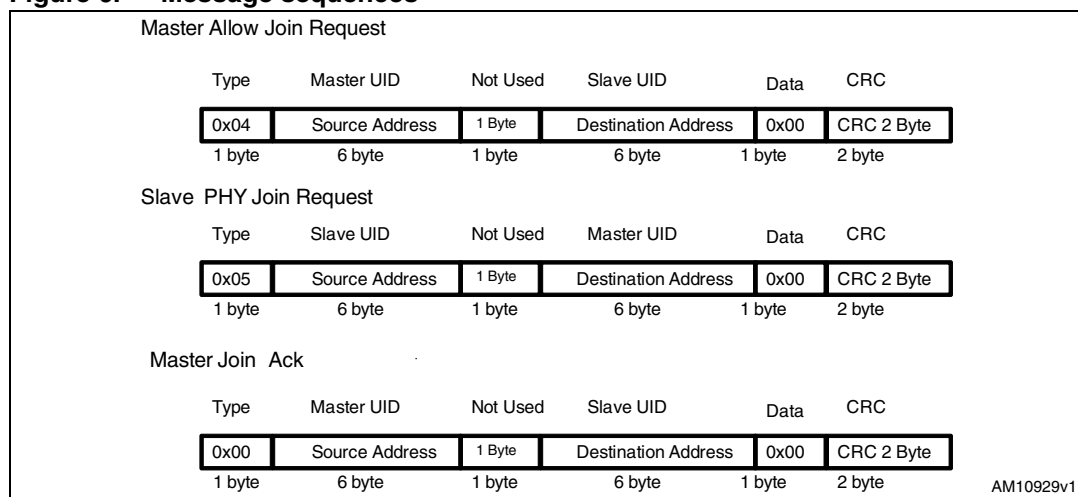
- Send the allow join
- Wait for the join acknowledge
- When the join request arrives, it sends a join ACK and marks the module as “Joined”.

At the same time the actions performed by the slave are:

- Waits for the allow join request
- When an allow join request arrives, sends a join request
- Waits for the JACK.
- When the JACK arrives, the module is able to receive a data.

The message sequences are described in Figure 6:

Figure 6. Message sequences



The join procedure is mandatory for nodes configured as slave; if the joined procedure doesn't perform correctly, a slave node isn't able to communicate with the master.

Appendix B STM32Fx family modem configuration parameter description

The following is a description of the modem configuration data struct:

```
typedef struct{
    u32 m_Ser_H_Freq;           /* Frequency H Channel */
    u32 m_Ser_L_Freq;           /* Frequency L Channel */
    u8 m_Rx_Control;            /* RX Control Parameters */
    u8 m_SerGain;               /* Gain*/
    u16 m_SerZeroCrossing;      /*Zero Crossing*/
    SV_PRE_LEN m_SerPSKPreamLen; /*PSK Preamble Length*/
    u8 m_FSK_Config_Byte        /* FSK Configuration Byte */
}MOD_ConfigTypedef;
```

The valid value of the parameters are:

¢ m_Ser_H_Freq;	CENELEC A-B or C, compliant frequency integer value.
¢ m_SerL_Freq;	CENELEC A-B or C, compliant frequency integer value.
¢ m_ m_Rx_Control;	Rx Control byte value (see datasheet).
¢ m_SerGain;	integer value from 0 to 31.
¢ m_SerZeroCrossing;	Zero Crossing delay value.
¢ m_SerPSKPreamLen;	BIT_16, BIT_24, BIT_32,BIT_40.
¢ m_FSK_Config_Byte;	FSK Configuration Byte, (see datasheet).

Appendix C Example to configure the communication protocol

C.1 Configure the STM32Fx family serial layer

Follow the STM32Fx family USART configuration used inside the serial layer to communicate with a PLM device:

Usart ' USART3

USART_RX_GPIO ' GPIO_Pin_11_PORT_B

USART_TX_GPIO ' GPIO_Pin_10_PORT_B

The part of the code to modify is:

```
/*-----*/
/* Serial Define */
#define USART          USART3
#define USART_PORT     GPIOB
#define USART_TX       GPIO_Pin_10
#define USART_RX       GPIO_Pin_11
#define USART_TR       GPIO_Pin_9
#define USART_IRQ      USART3_IRQn
#define RCC_APBPeriph_USART RCC_APB1Periph_USART3
#define RCC_APBPeriph_USART_ClockCmd(Periph_USART,value)
RCC_APB1PeriphClockCmd(Periph_USART,value)
#define RCC_APBPeriph_GPIO RCC_APB2Periph_GPIOB
#define RCC_APBPeriph_GPIO_ClockCmd(Periph_GPIO,value)
RCC_APB2PeriphClockCmd(Periph_GPIO,value)
/*-----*/
The names "RCC_APB2Periph_GPIOB", "GPIO_Pin_x", "USART3" ecc.
```

The names are defined in the STM32F10x_StdPeriph_Driver.

Furthermore, it is necessary to call the function "vUARTInterruptHandler()" defined in serial.c in the appropriate InterruptServiceRoutine. The following example shows the ISR of USART3 in the Interrupt file of the STM32Fx family library:

Example:

```
void PPP_IRQHandler(void)
{
}
void UART4_IRQHandler(void)
{
}
void USART3_IRQHandler(void)
{
    vUARTInterruptHandler();
}
void USART1_IRQHandler(void)
{
}
```

C.2 Configure and run the data link layer

To configure the data link layer, it is necessary to call the function “DL_Init(DLType)” provided in the services.h, only one time in the main.c.

To run the data link it is necessary to call, in a loop, the function “DL_FSM()” provided in the dl_mgt20.h.

The following example shows these:

Example:

```
void main(void)
{
    DL_Init (DL_MASTER) ;
    while(1)
    {
        DL_FSM() ;
    }
}
```

5 References

1. ST7580 datasheet.
2. UM0932 user manual.

6 Revision history

Table 3. Document revision history

Date	Revision	Changes
04-Oct-2012	1	Initial release.

Please Read Carefully:

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

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

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

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

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

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

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

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

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

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

© 2012 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

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

www.st.com

