AN4101
Application note

LIN communication with STM8A-DISCOVERY

Introduction

The purpose of this application note is to describe the LIN demonstration that is delivered with the STM8A-DISCOVERY.

The software running in the STM8AF microcontroller configures the STM8AF board as a basic LIN master node, while the software in the STM8AL microcontroller sets the STM8AL board as a basic LIN slave node. Each board is connected through a single wire LIN bus, thanks to the LIN transceivers embedded in the L99PM62GXP device.

This document is organized as follows:

■ *Chapter 2* gives a short introduction to the LIN bus standard.

■ *Chapter 3* briefly describes the LIN transceiver of the L99PM62GXP.

■ *Chapter 4*, *Chapter 5* and *Chapter 6* describe the different UART implementations found in the STM8AL and STM8AF microcontroller families, and explain the principle of a basic software LIN driver in master or slave mode.

■ *Chapter 7* describes the demonstration software delivered with the STM8A-DISCOVERY, which uses some of the basic LIN drivers from *Chapter 4* and *Chapter 5*.

You can also refer to AN4178 for an example of the STM8AF board in slave mode, using the driver described in *Chapter 6*.

All these examples can be used as references for understanding how to configure USART and LINUART, and for implementing your own LIN driver in the STM8AF and STM8AL microcontrollers.

Note: *A LIN Software Package and a J2602 Software Package are available for free from STMicroelectronics, as detailed in* Chapter 8.

Table 1.   Applicable products

Type	Part numbers
Microcontrollers	– STM8AF5xxx, STM8AF6x26/4x/66/68, STM8AF6x69/7x/8x/9x/Ax – STM8AL313x, STM8AL314x, STM8AL316x, STM8AL3L4x, STM8AL3L6x – STM8A-DISCOVERY

Contents

1 Related documents

The following documents are related to this product:

● STM8A-DISCOVERY User manual (UM1574)
● STM8AF5xxx STM8AF6x69/7x/8x/9x/Ax datasheet
● STM8AL316x STM8AL314x STM8AL3L6x STM8AL3L4x datasheet
● STM8S and STM8A microcontroller families Reference Manual (RM0016)
● STM8L05xx, STM8L15xx STM8L162x, STM8AL31xx and STM8AL3Lxx microcontroller family reference manual (RM0031)
● L99PM62GXP datasheet
● LIN communication with two STM8AF boards of STM8A-DISCOVERY (AN4178)
● CAN communication with two STM8AF boards of STM8A-DISCOVERY (AN4179)

2 Introduction to the LIN bus

2.1 LIN consortium

The LIN consortium was created in 1998 to standardize serial communication in automotive applications. LIN complements CAN as a cost-effective subnetwork. You can get your LIN specification version at http://www.lin-subbus.org/.

The key features of LIN are as follows:

● low-cost, single-wire implementation based on ISO9141 and V_{BAT} voltage

● speed up to 20 Kbps, acceptable for many applications (limited for EMI reasons)

● a single master, multiple slave concept, without need for arbitration

● low-cost silicon implementation based on UART

● self-synchronization of slave nodes, allowing low-cost inaccurate clock source usage

● guaranteed latency times for signal transmission, allowing predictable system design

This low-price serial communication protocol replaces CAN in various non safety-critical automotive applications (for example, in the body, door, roof or seat). LIN node functionality mainly involves switches, actuators and sensors.

2.2 LIN principle

The master node decides when and which frame shall be transferred on the bus. Slave nodes simply react when they detect a Break on the bus.

The master node has control over the whole bus and protocol. It decides which message at what time is to be transferred over the bus. It also does the error handling.

For this, the master:

● sends the Header, which consists of a Break, a Synch byte and an Identifier field

● monitors data bytes and checksum bytes to evaluate them for consistency

● is the reference for the clock base

The slave node receives or transmits data when an appropriate identifier is sent by the master.

For this, the slave:

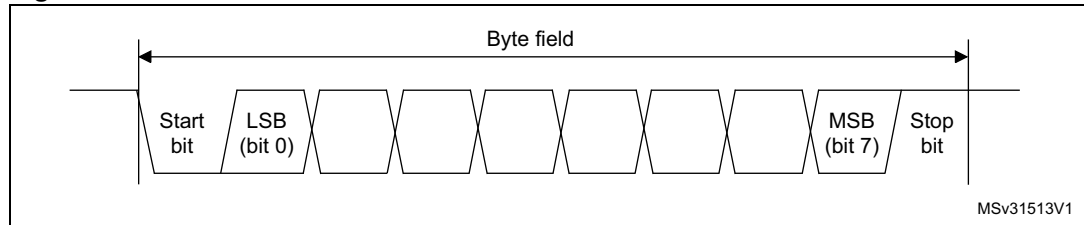1. checks the identifier

2. depending on the identifier value, determines which of the following to do:
   – receive data
   – transmit data
   – do nothing

Note: *When transmitting, the slave sends from 1 to 8 data bytes, and 1 checksum byte.*

2.3 LIN frame

LIN is based on the standard UART format, with 1 start bit, 8 data bits (LSB first), and 1 stop bit.

Figure 1. LIN frame

The LIN frame is composed of a header and a response.

The header consists of:

● the Break field

● the Synch field

● the Protected Identifier field

The response consists of:

● from 1 to 8 bytes of data

● 1 checksum byte

Figure 2. LIN frame header and response
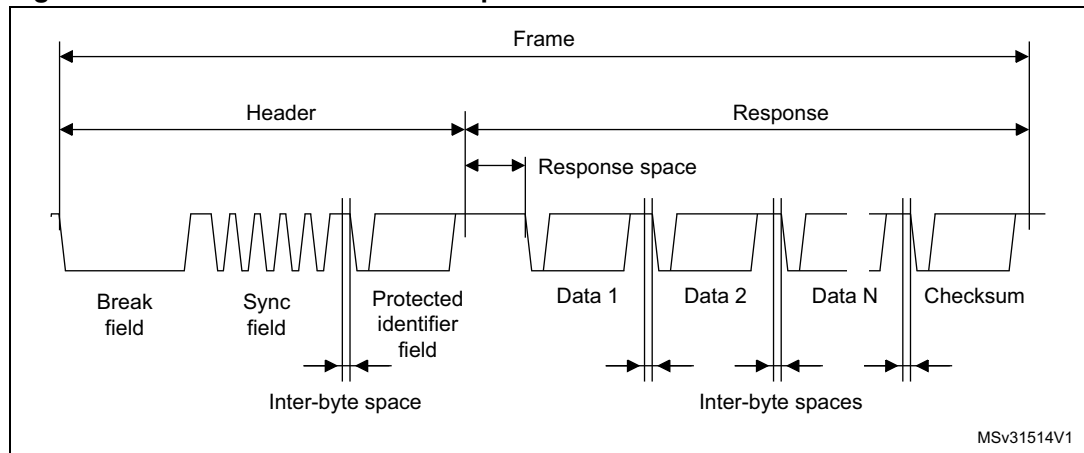
The header is always sent by the master node. The response is sent by only one node: either the master node, or one of the slave nodes.

The node sending a frame response is called the publisher. There is always only one publisher for each frame.

A node interested in a frame response is called a subscriber. There can be several subscribers for a frame.

2.4 LIN physical layer

The LIN bus is single-wire, and is connected via a pull-up resistor to the battery voltage V_{BAT}. The LIN transceiver is an enhancement of the ISO 9141 standard.

There are two possible bus states:

● dominant
● recessive

The LIN bus does not need to resolve bus collisions, since only one message is allowed on the bus at a time. This guarantees a deterministic system with known latency times.

The default state is recessive. A dominant state occurs when the transmitter pulls down the bus line. All LIN transmitters operate as and gates, which means that they must all be in a recessive state for the bus to be in a recessive state.

In *Figure 3*, Rx and Tx signals refer to the microcontroller USART or LINUART signals, while V_{BUS} refers to the LIN bus.

Figure 3.   LIN bus circuit diagram
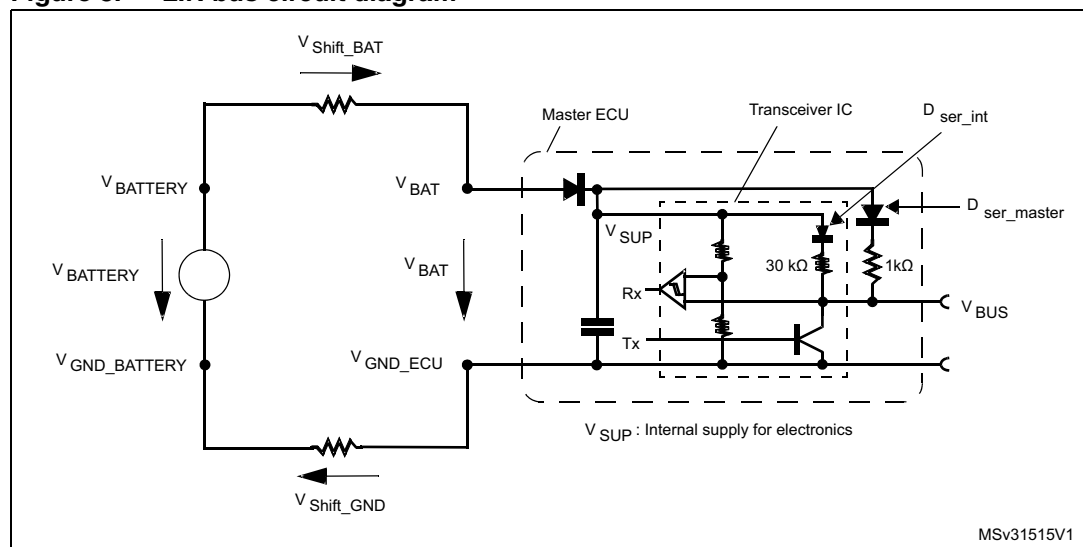
3   LIN transceiver

A LIN transceiver is embedded in the L99PM62 power management integrated circuit, which is present on both STM8AF and STM8AL boards. For more information, refer to the UM1574 STM8A-DISCOVERY user manual, and to the L99PM62GXP datasheet.
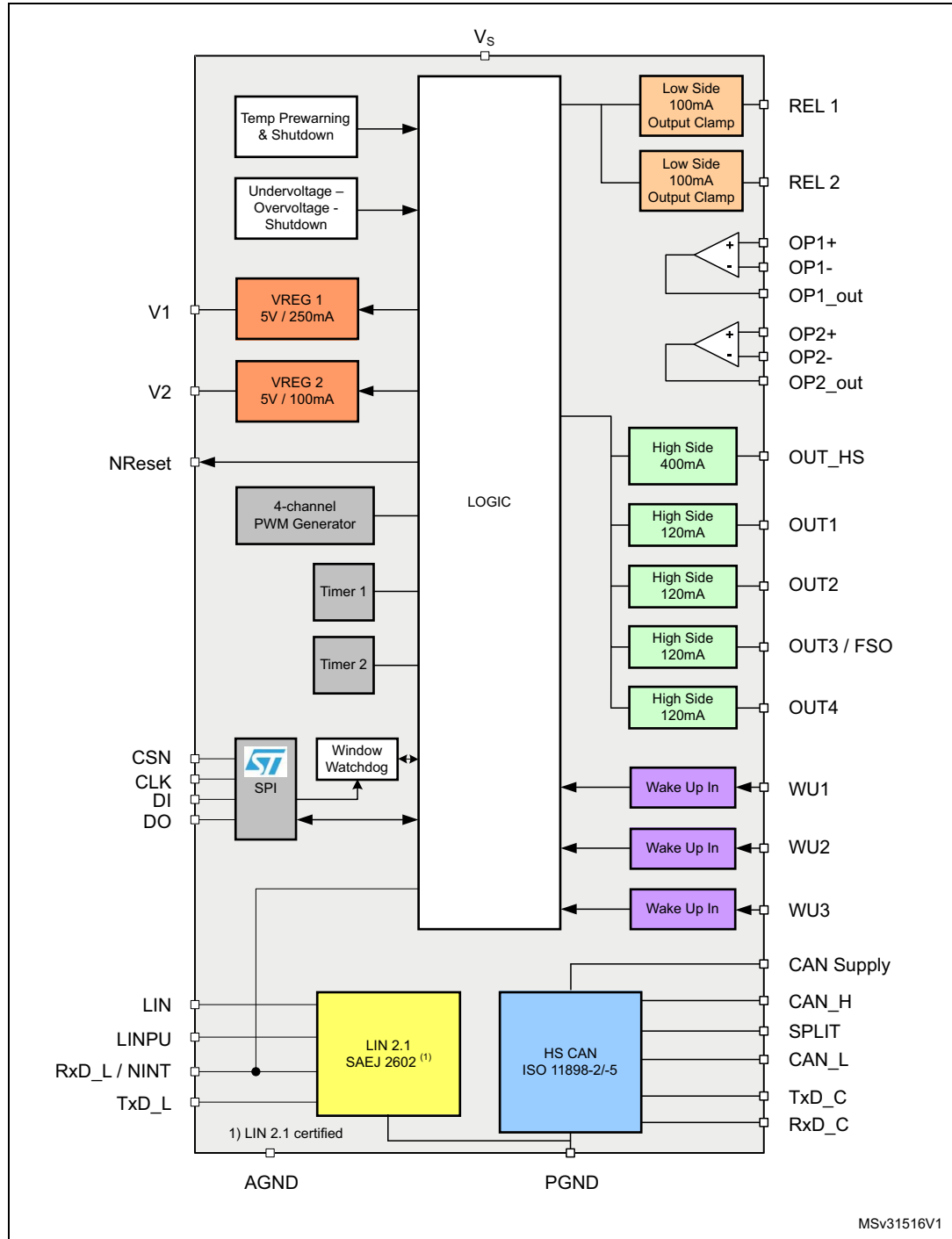
Figure 4.   LIN transceiver embedded in the L99PM62GXP

4 Implementing a LIN master node with STM8AF board

4.1 STM8AF microcontroller LINUART and USART features for LIN master

The LINUART and USART inside the STM8AF microcontroller implement both generation and detection of LIN Breaks.

LIN Break generation allows sending of 13-bit Breaks on the TX pin, thanks to the LIN protocol requirement.

LIN Break detection allows detection of feedback from the LIN transceiver, before the Synch Field (55h data byte) transmission is requested on the TX pin. There is a dedicated flag, LBDF, which may or may not trigger the receive interrupt depending on the LBDIE bit configuration.

4.2 Software algorithm

The software algorithm described below is implemented in the demonstration software of the STM8A-DISCOVERY (STM8AF microcontroller). Refer to *Section 7.2: STM8AF software description* for details.

4.2.1 LIN state machine

The LIN task is called regularly by the main routine. This task is a basic LIN master driver that implements a state machine with the following states:

● Idle
● Break
● SynchField
● Identifier
● DataReception
● DataTransmission

4.2.2 LINUART interrupt

The LINUART interrupt reads and clears the hardware flags in the LINUART registers, and sets the appropriate software flags to allow transitions in the basic LIN driver state machine:

● **BreakReceived**, which is set if the break detection flag (LBDF) is set in the interrupt routine
● **DataReceived**, which is set if the RXNE flag is set
● **ReceptionError**, which is set if the case framing error flag (FE) or the overrun flag (OR) is set

5 Implementing a LIN slave node with STM8AF board

5.1 STM8AF microcontroller LINUART features for LIN slave

LINUART supports autonomous handling of LIN headers. The software is notified of header reception through a dedicated flag, LHDF. This means that a single interrupt occurs only when the entire header is received, including the Break, the Synch Field and the Identifier Field.

LINUART implements a mute mode, allowing the software to remain quiet without generating any interrupts until reception of the next header. This is very useful for filtering frames for which the STM8AF is neither publisher nor subscriber. All data bytes of the frame response will thus be ignored by the LINUART, and will not generate any interrupts.

The automatic resynchronization feature allows a LIN slave to be built with the STM8AF microcontroller, without the need for an external crystal, oscillator or resonator. The LINUART can synchronize on the Synch Field of each received header, and automatically adjusts the baudrate prescaler in order to receive and send the rest of the frame with the right frequency tolerance, whatever the frequency deviation of the embedded internal RC oscillator (HSI) may be.

If one of the following errors is detected during the header reception, the software is notified through the LHE flag:

- **resynchronization error**, if the deviation is greater than 14%
- **identifier error** (framing error or parity error)
- **break delimiter error**, if the delimiter is too short
- **header timeout**, if the LIN header is longer than the T_HEADER_MAX defined in the LIN standard

In order to enhance system robustness, the LINUART can detect a Break at any time, even while receiving a data byte, and resynchronizes upon detecting the Break.

5.2 Software algorithm

The software algorithm described below is NOT implemented in the demonstration software of the STM8A-DISCOVERY. Refer to AN4178 to use another piece of demonstration software for the STM8A-DISCOVERY running the STM8AF board as a LIN slave.

Note: *The demonstration described in AN4178 is based on two STM8AF boards connected together: one a LIN master, and the other a LIN slave.*

5.2.1 LIN state machine

The LIN task is called regularly by the main routine. This task is a basic LIN slave driver that implements a state machine with the following states:

- Idle
- Identifier
- DataReception
- DataTransmission

5.2.2 LINUART interrupt

The LINUART interrupt reads and clears the hardware flags in the LINUART registers, and sets the appropriate software flags to allow transitions in the basic LIN driver state machine:

● **HeaderReceived**, which is set if the header detection flag (LHDF) is set in the interrupt routine

● **DataReceived**, which is set if the RXNE flag is set

● **IdentifierParityError**, which might be set if the header detection flag is set, after first checking the parity flag

● **ReceptionError**, which is set if the case framing error flag (FE), the overrun flag (OR), or the LIN synchro flag (LSF) is set

6 Implementing a LIN slave node with STM8AL board

6.1 STM8AL microcontroller USART features

USART does not embed any particular LIN features. A LIN slave node can be implemented by building a break detection based on a 00h data reception with a framing error (FE flag).

An external resonator or crystal is required in order to meet the ±1.5% maximum clock deviation allowed by the LIN standard.

6.2 Software algorithm

The software algorithm described below is implemented in the demonstration software of the STM8A-DISCOVERY (STM8AL microcontroller).

6.2.1 LIN state machine

The LIN task is called regularly by the main routine. This task is a basic LIN slave driver that implements a state machine with the following states:

- Idle
- Break
- SynchField
- Identifier
- DataReception
- DataTransmission

6.2.2 LINUART interrupt

The LINUART interrupt reads and clears the hardware flags in the LINUART registers, and sets the appropriate software flags to allow transitions in the basic LIN driver state machine:

- **BreakReceived**, which is set if both of the following conditions are met:
  - a data byte with the value 00h is in the data register (DR)
  - the framing error flag (FE) is set
- **DataReceived**, which is set if the RXNE flag is set
- **ReceptionError**, which is set if the framing error flag (FE) or the overrun flag (OR) is set

7 LIN demonstration software

STM8A-DISCOVERY contains all you need to start a basic LIN communication:

● STM8AF board with the following:
    – a STM8AF5288 microcontroller
    – a LIN transceiver
    – push buttons
    – LEDs
    – a ST-LINK/V2 for debugging and programming
● STM8AL board with the following:
    – a STM8AL3L68 microcontroller
    – a LIN transceiver
    – push buttons
    – LEDs
    – a LCD display
    – a ST-LINK/V2 for debugging and programming
● Dual type A to mini-B USB cable to connect the boards to your PC

The following sections introduce the demonstration software running in the STM8AL and STM8AF boards.

This application does not require any additional hardware. Once the STM8A-DISCOVERY is powered up through the dual type A to mini-B USB cable connected to the host PC, the application is ready to start LIN communication, and updates the LEDs and LCD display according to the user's action with the push buttons and the trimmer.

This software example comes preloaded with your STM8A-DISCOVERY, and is readily accessible at application power-up.

Note: *There are hidden test modes in this demonstration software. Consequently, the software source code contains additional modules not described in this application note.*

7.1 Getting started

1. Connect CN3 and CN4 on the STM8AF board to CN3 and CN4 on the STM8AL board.
2. Connect both type A connectors of the USB cable to a PC.
3. Connect the mini-B connector of the USB cable to the STM8AF board. All LEDs should blink once on the STM8AF board and twice on the STM8AL board, and then the LIN communication starts between both boards.
4. Turn the RV1 trimmer on the STM8AF board to adjust the value from 0 to 100 on the STM8AL board LCD through the LIN bus.
5. Push button USER1 on the STM8AF board in order to sequentially display the LCD bars on the STM8AL board through the LIN bus. Push button USER2 on the STM8AF board to sequentially switch them off one by one.
6. Push button USER1 on the STM8AL board to sequentially switch on green LEDs LD4, LD5, LD6 and LD7 on the STM8AF board through the LIN bus. Push button USER2 on the STM8AL board to sequentially switch them off one by one.

7.2 STM8AF software description

7.2.1 STM8AF peripherals

The following STM8AF peripherals are used by the application:

● **SPI**, which is used in master mode at 250 kHz to initialize the L99PM62GPX and refresh its watchdog.

● **ADC**, which is used to read the RV1 trimmer value

● **TIM4**, which allows generation of a 1 ms timebase

● **HSE**, the High Speed External Clock, which when enabled allows the use of the external 16 MHz crystal oscillator

● **LINUART**, which is used to perform LIN communication in master mode

● **GPIO**

– PA3, PD3, PD0, PE3 and PC3 ports are used for LED display.

– PE2 and PE1 are used for USER1 and USER2 push buttons.

– PC7, PC6, PC5 and PE5 are used respectively for the SPI MISO, MOSI, SCK and NSS.

– PD5 and PD6 are used for LINART TX and RX.

7.2.2 Software modules

main

This module contains the initialization routines and the main loop.

appli

This module contains the application's main tasks:

● reading the status of the USER1 and USER2 buttons, and updating the corresponding signals in the LIN frame that will be sent to the STM8AL board

● reading the value of the RV1 trimmer (thanks to the ADC peripheral), and updating the corresponding signal in the LIN frame that will be sent to the STM8AL board

● updating the status of the LEDs according to the signals read in the LIN frame received from the STM8AL board

If the LIN communication is broken, the application will switch on the red LD3 LED.

l99pm62drv

This module contains the driver for the L99PM62GXP device, which is controlled by the STM8AF microcontroller through SPI.

If the L99PM62GXP device reports an error to the STM8AF through SPI, the software will switch on the red LD3 LED. The other LEDs will give additional information as follows:

● LD4 on: SPI error

● LD5 on: V_S out of range

● LD6 on: thermal shutdown

lin

This module contains the LIN driver state machine described in *Section 4.2.1: LIN state machine*.

adc

This module contains the ADC initialization routine, and the function for reading a converted value on the ADC channel 8, mapped on the PE7 port.

can

This module contains the CAN communication functions which are used only by a hidden test mode, and are not used in this demonstration software.

stm8s_it

This module contains the following interrupt routines:

● External interrupt of the port E routine, and update of the UserButton1/UserButton2 software flag, when a falling edge is detected on the PE2/PE1 ports

● CAN receive interrupt routine, which is not activated in the demonstration software (test mode only)

● LINUART receive/error interrupt routine, which implements the algorithm described in *Section 4.2.2: LINUART interrupt*

● ADC2 interrupt routine, which reads the converted value and stores it in the Conversion_Value variable

● TIM4 interrupt routine, which sets timebase ticks for LIN, CAN (not used in the demonstration software), application, and L99PM62GXP watchdog refresh

7.2.3 STM8S/A standard peripheral library modules

The demonstration software is based on the STM8S/A standard peripheral library version 2.1.0, dated November 2011.

The following driver modules are used:

● **stm8s_exti.c**, for the external interrupts on port E (USER1 and USER2 buttons)

● **stm8s_adc2.c**, for converting the RV1 trimmer value

● **stm8s_can.c**, for the CAN test mode (not used in the demonstration software)

● **stm8s_clk.c**, for configuring clocks and enabling HSE

● **stm8s_gpio.c**, for initializing, reading and updating all ports

● **stm8s_spi.c**, for SPI communication with the L99PM62GXP device

● **stm8s_tim4.c**, for generating a 1 ms timebase

● **stm8s_uart3.c**, for initializing LINUART

Note: *LINUART is described as UART3 in the RM0016 STM8S and STM8A reference manual.*

7.3 STM8AL software description

7.3.1 STM8AL peripherals

The following STM8AL peripherals are used by the application:

● **SPI**, which is used in master mode at 250 kHz to initialize the L99PM62GPX and refresh its watchdog

● **TIM4**, which allows generation of a 1 ms timebase

● **HSE**, the High Speed External Clock, which when enabled allows the use of the external 16 MHz crystal oscillator

● **USART**, which is used to perform LIN communication in slave mode

● **GPIO**

– PE7 and PC7 ports are used for LED display.

– PE1 and PE2 are used for the USER1 and USER2 push buttons.

– PB7, PB6, PB5 and PB4 are used respectively for the SPI MISO, MOSI, SCK and NSS.

– PC3 and PC2 are used for USART TX and RX.

7.3.2 Software modules

main

This module contains the initialization routines and the main loop.

appli

This module contains the application's main tasks:

● reading the status of the USER1 and USER2 buttons, and updating the corresponding signals in the LIN frame that will be sent to the STM8AF board

● updating the status of the LEDs according to the signals read in the LIN frame received from the STM8AF board

If the LIN communication is broken, the application will switch on the red LD3 LED and display "ERR2" on the LCD screen.

l99pm62drv

This module contains the driver for the L99PM62GXP device, which is controlled by the STM8AF microcontroller through SPI.

If the L99PM62GXP device reports an error to the STM8AL through SPI, the software will display "ERR1" on the LCD screen. The LCD bars will give additional information as follows:

● BAR0 on: SPI error

● BAR1 on: V_S out of range

● BAR2 on: thermal shutdown

lin

This module contains the LIN driver state machine described in *Section 6.2.1: LIN state machine*.

lcd

This module contains functions to drive the LCD display.

delay, icc_measure, lowpower

These modules are used for low-power modes. They are used by a hidden test mode, and are not used in this demonstration software. Please refer to AN4102 for a description of these modules.

stm8l15x_it

This module contains the following interrupt routines:

● External interrupt of the pin1 routine, and update of the UserButton1 software flag, when a falling edge is detected on the PE1 port

● External interrupt of the pin2 routine, and update of the UserButton2 software flag, when a falling edge is detected on the PE2 port

● External interrupt of the pin6 routine, which is not activated in the demonstration software (test mode only)

● TIM4 interrupt routine, which sets timebase ticks for LIN, CAN (not used in the demonstration software), application, and L99PM62GXP watchdog refresh

● USART receive/error interrupt routine, which implements the algorithm described in *Section 6.2.2: LINUART interrupt*

7.3.3 STM8L15x/16x standard peripheral library modules

The demonstration software is based on the STM8L15x/16x standard peripheral library version 1.5.0, dated May 2011.

The following driver modules are used:

● **stm8l15x_flash.c**, used for test mode only

● **stm8s_adc.c**, used for test mode only

● **stm8l15x_exti.c**, for the external interrupts on pin1 and 2 (USER1 and USER2 buttons)

● **stm8s_clk.c**, for configuring clocks and enabling HSE

● **stm8s_gpio.c**, for initializing, reading and updating all ports

● **stm8s_spi.c**, for SPI communication with the L99PM62GXP device

● **stm8s_lcd.c**, for SPI communication with the L99PM62GXP device

● **stm8s_tim4.c**, for generating a 1 ms timebase

● **stm8s_usart.c**, for initializing USART

7.4 Debugging with ST visual develop (STVD)

7.4.1 Downloading the software tools

1. Download your free software environment (IDE). Choose between:
    –   ST's MCU toolset, composed of ST Visual Develop (STVD) and ST Visual Programmer (STVP), available at http://www.st.com
    –   IAR Embedded Workbench for STM8 30-day time-limited edition, available at http://www.iar.com under the "Software Download" tab
2. Download your free compilers if you chose STVD. While IAR Embedded Workbench for STM8 includes its own compiler, STVD must be used together with one of the following:
    –   Cosmic 32 K 1-year time-limited edition, available at http://www.cosmicsoftware.com/download_stm8_32k.php
    –   Raisonance 32 K, available at http://www.mcu-raisonance.com

7.4.2 Opening the workspace and selecting the correct project

Before opening the workspace, first unzip the zip file delivered with this application note.

1. In ST Visual Develop, click File / Open Workspace...
2. Select the stm8a_discover_workspace.stw file in either the "Projects\STVD\cosmic" folder or the "Projects\STVD\raisonance" folder, depending on the compiler you want to use.

    The workspace contains the following projects:
    –   stm8af_discover
    –   stm8al_discover
3. Choose a project by selecting it in the workspace window and right-clicking it with the mouse. Choose "Set as Active Project".

7.4.3 Rebuilding and debugging the project

You can rebuild the project, download the code and debug. Make sure to connect the USB cable to the appropriate board.

Refer to the UM1574 STM8A-DISCOVERY User manual for more information on debugging.

8   LIN and J2602 software packages

This application note and the attached project show how to use the STM8AF and STM8AL features to implement a LIN driver.

A complete LIN Software Package solution is available for free from STMicroelectronics for STM8AF and STM8AL microcontroller families. This software package is compliant with the LIN 1.3, LIN 2.0 and LIN 2.1 standards.

A J2602 software package is also available for the STM8AF family.

A LIN 2.1 conformance test and J2602 conformance test have been run by a test house accredited by the LIN consortium. Certificates are available at www.st.com.

Please refer to UM0941 and UM1583 for more details, and contact your STMicroelectronics sales office to get one of these packages.

9 Revision history

Table 2. Document revision history

Date	Revision	Changes
03-Dec-2012	1	Initial release.