AN4178
Application note

LIN communication with two STM8AF boards of STM8A-DISCOVERY

Introduction

This application note describes a LIN demonstration. One STM8AF board is configured as a basic LIN master node, while the other STM8AF board is configured 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.

Please read the AN4101 first, which describes the demonstration delivered with the STM8A-DISCOVERY.

You can use all these examples as a reference for understanding how to configure LINUART, and how to implement your own LIN driver in STM8AF microcontrollers.

Note: A LIN Software Package and a J2602 Software Package are available for free from STMicroelectronics. Please contact your STMicroelectronics sales office for details.

Table 1. Applicable products

Type	Part numbers
Microcontrollers	– STM8AF5xxx – STM8AF6x26/4x/66/68 – STM8AF6x69/7x/8x/9x/Ax – 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
- STM8S and STM8A microcontroller families Reference Manual (RM0016)
- L99PM62GXP datasheet
- LIN communication with STM8A-DISCOVERY (AN4101)
- CAN communication with two STM8AF boards of STM8A-DISCOVERY (AN4179)

2 Implementing a LIN master node with STM8AF board

2.1 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.

2.2 Software algorithm

The software algorithm described below is implemented in the STM8AF master software. Refer to *Section 4.2: STM8AF LIN master software description*.

2.2.1 LIN state machine

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

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

2.2.2 LINUART interrupt

The LINUART interrupt reads and clears the hardware flags in the LINUART registers, and sets 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
- **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

3 Implementing a LIN slave node with STM8AF board

3.1 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 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 more 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.

3.2 Software algorithm

The software algorithm described below is implemented in the STM8AF slave software. For details, refer to *Section 4.3: STM8AF LIN slave software description*.

3.2.1 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

3.2.2 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

4 LIN demonstration software

In order to run this demonstration, you must have two STM8AF boards from two STM8A-DISCOVERY bundles.

This demonstration does not require any additional hardware, but you will need to change a solder bridge configuration on one board.

Before running this demonstration, make sure to load the LIN slave software and the LIN master software in each of the two STM8AF boards.

4.1 Getting started

1. Set the SB8 solder bridge to ON by soldering it on the first STM8AF board. This board will be the STM8AF master board.
2. Connect CN3 and CN4 on the first STM8AF board to CN3 and CN4 on the second STM8AF board.
3. Connect both type A connectors of the USB cable to a PC.
4. Connect the mini-B connector of the USB cable to the second STM8AF board. This board will be the STM8AF slave board.
5. Load the STM8AF_LIN_Slave.s19 file into the STM8AF slave board using the STVP tool.
6. Disconnect the STM8AF slave board from the mini-B connector of the USB cable.
7. Connect the mini-B connector of the USB cable to the STM8AF master board.
8. Load the STM8AF_LIN_Master.s19 file into the STM8AF master board using the STVP tool.
9. All LEDs should blink once on STM8AF LIN master board, and all LEDs should be switched on for one second on the STM8AF LIN slave board. The LIN communication then starts between both boards.
10. Push button USER1 on the first STM8AF board in order to sequentially switch on green LEDs LD4, LD5, LD6 and LD7 on the second STM8AF board through the LIN bus. Push button USER2 on the first STM8AF board to sequentially switch them off one by one.
11. Push button USER1 on the second STM8AF board in order to sequentially switch on green LEDs LD4, LD5, LD6 and LD7 on the first STM8AF board through the LIN bus. Push button USER2 on the second STM8AF board to sequentially switch them off one by one.

Note: If you want to reuse the STM8AF Master board to run the initial demonstration delivered with the STM8A-DISCOVERY and described in AN4101, you will need to set the SB8 solder bridge to OFF by unsoldering it.

4.2 STM8AF LIN master software description

4.2.1 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
- **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 the 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.

4.2.2 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 STM8AF LIN slave board
- updating the status of the LEDs according to the signals read in the LIN frame received from the STM8AF LIN slave 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 2.2.1: LIN state machine*.

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

4.2.3 STM8S/A standard peripheral library modules

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

The following driver modules are used:

- **stm8s_exti.c**, for the external interrupts on port E (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_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.

4.3 STM8AF LIN slave software description

4.3.1 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
- **TIM4**, which allows generation of a 1 ms timebase
- **LINUART**, which is used to perform LIN communication in slave mode. The automatic resynchronization feature is enabled.
- **GPIO**
  - PA3, PD3, PD0, PE3 and PC3 ports are used for LED display.
  - PE2 and PE1 are used for the 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.

4.3.2 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 LIN master board
- updating the status of the LEDs according to the signals read in the LIN frame received from the STM8AF LIN slave 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 3.2.1: LIN state machine*.

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
- TIM4 interrupt routine, setting timebase ticks for LIN, CAN (not used in demonstration software), application, and L99PM62GXP watchdog refresh
- LINUART receive/error interrupt routine, which implements the algorithm described in *Section 3.2.2: LINUART interrupt*

4.3.3 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_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.

4.4 Debugging with ST visual develop (STVD)

4.4.1 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

4.4.2 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".

4.4.3 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 UM1574 STM8A-DISCOVERY User manual for more information about debugging.

5 Revision history

Table 2. Document revision history

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