Low Energy UART

AN0017 - Application Note

Introduction

This application note demonstrates how to use the Low Energy UART (LEUART™) module on the EFM32 microcontrollers. The LEUART is able to run full UART communication even when the device is in deep sleep mode EM2. Together with intelligent interrupt functions and flexible DMA integration, this enables simple and energy friendly communication .

This application note includes:

- This PDF document
- Source files (zip)
    - Example C-code
    - Multiple IDE projects

1 LEUART Theory

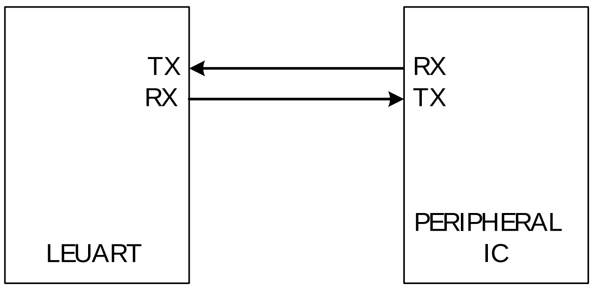
1.1 General theory

The EFM32 LEUART is a unique Low Energy UART that offers two-way communication on a very strict power budget. Only a 32.768 kHz clock source is needed to allow UART communication up to 9600 baud/s. This means that the EFM32 LEUART can operate in deep sleep mode EM2, and wait for an incoming UART frame while consuming extremely little energy. When a UART frame is completely received by the LEUART, the CPU can quickly be woken up. Alternatively, multiple frames can be transferred to memory by the DMA before waking up the CPU. The EFM32 LEUART also incorporates functionality to handle higher level communication protocols, e.g. the option to block incoming frames until a configurable start frame is detected, and to detect a configurable signal frame (to indicate e.g. the end of a transmission).

In the same way as received, data can be transmitted in EM2 either on frame by frame basis with data directly from the CPU, or in larger groups of frames trough the DMA. The EFM32 LEUART includes all needed hardware support to make asynchronous serial communication possible with minimum software interference, while consuming extremely small amounts of energy.

The advantage of the LEUART is the ability to operate in EM2, while most other modules are turned off for energy conservation. The option to enable low energy serial communication, in combination with the DMA's ability to read and write from memory without CPU intervention, offers wide functionality for system designers using the EFM32 in low energy applications.

Figure 1.1. LEUART two link full duplex connection



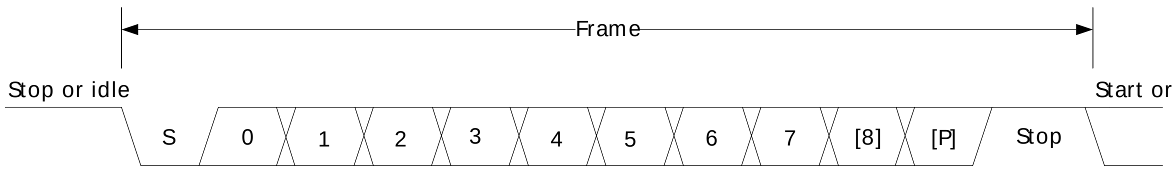
2 LEUART Configuration

2.1 Frames, Transmission & Parity

The LEUART relates to frames for data transmission. A LEUART frame consists of a start bit, 8 or 9 data bits, an optional parity bit, and 1 or 2 stop bits (Figure 2.1 (p. 3)). A transmission is initiated by a start bit that pulls the line down from its idle high state. After that the data and parity bits are sent sequentially until the frame transmission is ended by the stop bits that holds the line high. Then the line either enters its high idle state, or a new start bit is sent. Technically, when a frame is ready to be transmitted, it is transferred from the transmit register to a shift register where the bits are sent one by one, least significant bit first. The entire frame format can be inverted, to e.g. allow use of low idle state.

The parity bit at the end of the data transmission is an optional method for light error detection. Three different parity modes are available: that is no parity, even parity and odd parity. All parity generation and checking are done in hardware, and interrupt flags are available to indicate if parity error in the frame is detected. The frame format wanted is set during initialization of the selected LEUART. The emlib offers a initialization function that defines all the necessary settings to start communication using the LEUART. All parties of the communication channel must agree on the frame format for communication to be possible. For more extensive information on the LEUART registers and opportunities, refer to the reference manual for the device.

Figure 2.1. LEUART frame format



2.2 Clock Sources

The LEUART, like other LE peripherals in the EFM32 microcontrollers, can be driven by three different clock sources, the Low Frequency RF Oscillator (LFRCO), the Low Frequency Crystal Oscillator (LFXO) and the High Frequency Core Clock divided by 2 (HFCORECLK_LE/2). The HFCORECLK_LE can in turn be driven by either the High Frequency RC Oscillator (HFRCO), High Frequency Crystal Oscillator (HFXO) or even any of the LF oscillators mentioned. This flexibility in assigning clocks, gives the system designer a wide range of possibilities for using the LEUART features. This means that beside running on a LF clock source during low energy operation in EM2, the LEUART can also be utilized as a supplement to the UART when more UART communication channels are needed. However for the LEUART to achieve baud rates above 9600 baud/s, the chosen clock source must be the HFCORECLK_LE/2 which is only available in EM1-EM0.

2.3 Baud rates

The selected LEUART clock source defines the baud rates that are obtainable through the LEUART. In standard Low Energy operation, the LEUART offers baud rates between 300 to 9600 baud/s on a 32.768 kHz clock. For details on which baud rates are supported, refer to the reference manual for the device. The emlib also includes methods for calculating which baud rates are available, extracting the current baud rate, and setting the baud rate. With the HFCORECLK_LE/2 selected as th LEUART clock source, higher baud rates are obtainable, but this also implies that the LEUART it will not operate below EM1 and more energy is needed. In addition, since one of the HF clock need to be running, both the UART and USART are also available.

2.4 DMA Integration

The LEUART has full DMA support even in EM2. In integration with the LEUART, the DMA is a very powerful tool to minimize the need for CPU interference.

- The LEUART can be configured to request DMA for data either when the transmit buffer is empty or if both the transmit buffer and the shift register are empty.
- The LEUART can be configured to request the DMA to read when the receive buffer is full.
- When a frame of parity error is detected in the receive register, the `ERRSDMA` bit in the `LEUARTn_CTRL` register can be set to omit the read request to the DMA.

When operating in EM2, the DMA controller must be powered up in order to perform the transfer. This is automatically performed for read operations if `RXDMAWU` in `LEUARTn_CTRL` is set and for write operations if `TXDMAWU` in `LEUARTn_CTRL` is set. To make sure the DMA controller still transfers bits to and from the LEUART in low energy modes, these bits must be configured accordingly. In EM2 the DMA runs off the HFRCO which is also woken up and shut off automatically. The DMA must also be enabled and configured correctly to handle the LEUART data. For more information on how to initialize an interaction between the LEUART and DMA, see the supplied software examples and the reference manual for the device.

2.5 Pulse Generator and Extender

The LEUART has an optional pulse generator for the transmitter output, and a pulse extender on the receiver input. It will change the input and output format of the LEUART from NRZ to RZI. The width of the pulses from the pulse generator can be configured from 31.25 µs to 500 µs. At 2400 baud/s or lower, the pulse generator is also able to generate RZI pulses compatible with the IrDA physical layer specification.

2.6 Interrupts

A wide variety of interrupts are available, both during receive and transmit, to support the low energy advantages of interrupt driven applications. An interrupt can be triggered when the receive or transmit registers are empty, or if any errors are detected during transmit. Also there is the ability for the LEUART to trigger an interrupts when specific configurable frames are detected. This allows the construction of higher level communication protocols on top of the LEUART. A special multi processor mode is even available to enable individual addressing and trigger only the desired MCU to receive and act on the data sent. This is a useful feature in systems where multiple UART ICs are communicating on the same channel. In this way the desired receiver can be addressed by starting and ending the transmission with certain frames that will only trigger an interrupt in the desired receiver. In the software examples supplied, the Signal Frame Interrupt functionality is used to wake the CPU only when a specific frame is detected. All other frames are loaded into the memory by the DMA and do not generate any response from the CPU until the preconfigured Signal Frame is detected by the LEUART.

2.7 Freeze Mode and LF Domain Synchronisation

Synchronization into the low frequency (LF) domain is necessary to modify some of the LEUART registers. To avoid unnecessary stalling when multiple registers are to be modified, all register writes should be done inside a block initiated by `void LEUART_FreezeEnable(LEUART_TypeDef *leuart, bool enable)` where enable is set true, and ended by the same function call where enable is set false. In this way all register modifications will be performed during a single synchronisation. See the reference manual for the device for more information on accessing and modifying asynchronous registers.

2.8 Half-Duplex Operation

The LEUART offers an option to locally loopback the transmitted data to the receive pin. This is useful for debugging, as the LEUART can receive the data it transmits, but it is also used to allow the LEUART to read and write to the same pin, which is required for some half duplex communication modes. When doing full duplex communication, two data links are provided, making it possible for data to be sent and received at the same time. In half duplex mode, data is only sent in one direction at a time. There are several possible half duplex setups. Both single and double data-links, or with an external driver. When communicating over a single data-link, the transmitter must be tristated whenever data is not transmitted. The LEUART can automatically tristate the transmit pin whenever the transmitter is inactive, if the `AUTOTRI` pin in the `LEUARTn_CTRL` register is set.

2.9 GPIO and Routing

The LEUART modules have the ability to route its TX and RX pins to some different predefined locations on the MCU pinout. The `LEUARTn_ROUTE` register must be set to enable and route the pins to the desired location.

To enable the LEUART to interact with any external system components, like the RS232 port or another peripheral IC, the GPIO I/O pins must be configured accordingly. The GPIO has a variety of different pin modes available, and in the supplied code examples the TX pins are configured as push-pull output. The RX pins are enabled as inputs with a pull-up. The pull-up helps to define the input state in case line is not driven to a defined value by the other part. This can happen if you enable the RX module before the TX module. Refer to the application note "AN0012 GPIO" more examples on GPIO mode setting.

3 Software Examples

The following software examples demonstrates a way to achieve full UART communication, using the LEUART functionality of the EFM32.

3.1 Listening and Receiving in EM2.

In this example, LEUART communication is initialized using the DMA to transfer received data from the LEUART receive register to the system memory. When a preconfigure signal frame is received, the LEUART generates a interrupt, and the CPU wakes up and writes all the received data from the memory to the LCD.

3.1.1 Example 1: Receiving on DVK

This DVK can receive data both trough RS232, and directly through the pinout on the Prototype Board. This means that the example can be run in a variety of different setups, receiving data both from another DVK, a STK or a PC trough the RS232 port on the DVK. The LEUART1 RX pin is located on pin J10 on the DVK Prototype Board, and needs to be connected to the TX pin on the transmitting DVK or STK, or trough a RS232 cable to a PC. Note that the sender and received must have same ground and supply voltage when connecting the RX and TX lines directly without using the RS232 port.

3.1.2 Example 2: Receiving on STK

On the STK the data can be received trough the pinout located along the rim of the board. The LEUART1 RX pin is located on the PC7 pin on the bottom row.

3.2 Transmitting in EM2.

In this example, LEUART communication is initialized using the DMA to transfer data from a location in the local system memory to the LEUART, on given RTC(Real Time Clock) interrupts. Every other second, the RTC interrupt initializes the DMA to start feeding the LEUART TX transmit register with data, and returns to EM2. One of two predefined strings is then transmitted trough the LEUART to anyone listening. The two strings will alternate.

3.2.1 Example 3: Transmitting from DVK

The DVK can send data both directly through the pinouts on the Prototype board, and through the RS232 port provided. The example can in this way be set up and run in several different setups, both transmitting to another DVK, a STK or a PC. The LEUART1 TX pin is located on the pin J9 on the DVK Prototype Board, and needs to be connected to the RX pin on any receiving device, or to a PC through the use of a RS232 cable to a PC. Note that the sender and received must have same ground and supply voltage when connecting the RX and TX lines directly without using the RS232 port.

3.2.2 Example 4: Transmitting from STK

On the STK the data can be transmitted through the pinout located along the rim of the board. The LEUART1 TX pin is located on the PC6 pin on the bottom row.

4 Revision History

4.1 Revision 1.09

2014-05-07

Updated example code to CMSIS 3.20.5

Changed to Silicon Labs license on code examples

Added example projects for Simplicity IDE

Removed example makefiles for Sourcery CodeBench Lite

4.2 Revision 1.08

2013-10-14

New cover layout

4.3 Revision 1.07

2013-05-08

Added software projects for ARM-GCC and Atollic TrueStudio.

4.4 Revision 1.06

2012-11-12

Adapted software projects to new kit-driver and bsp structure.

4.5 Revision 1.05

2012-08-13

Adapted software projects to new peripheral library naming and CMSIS_V3.

4.6 Revision 1.04

2011-11-15

Updated transmit examples to use DMA callback to allow sleeping while transmitting.

4.7 Revision 1.03

2011-10-21

Updated IDE project paths with new kits directory.

4.8 Revision 1.02

2011-05-18

Updated projects to align with new bsp version

4.9 Revision 1.01

2010-11-16

Changed example folder structure, removed build and src folders.

Updated chip init function to newest efm32lib version.

Updated register defines in code to match newest efm32lib release.

4.10 Revision 1.00

2010-09-24

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

A.2 Trademark Information

B Contact Information

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701

Please visit the Silicon Labs Technical Support web page:
http://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Table of Contents

List of Figures

silabs.com

ZERO
ARM Cortex-M0+

TINY
ARM Cortex-M3

GECKO
ARM Cortex-M3

LEOPARD
ARM Cortex-M3

GIANT
ARM Cortex-M3

WONDER
ARM Cortex-M4