AN4457
Application note
Implementing an emulated universal asynchronous receiver transmitter on STM32F4 microcontroller series

Introduction

This application note describes how to implement an emulated universal asynchronous receiver transmitter (UART) on STM32F4 series.

Such an emulator is needed in applications that require more UARTs than the ones provided by STM32F4 microcontrollers.

The emulated UART is full-duplex, supports up to 9 data bits and baud rates up to 115200 bps. It also offers a high flexibility since any I/O pin can be used as TX or RX line. In addition, this UART emulation uses the DMA to minimize CPU usage.

This application note provides a basic example of communications between a hardware and a software UART as well as a summary of CPU load and firmware footprint.

The STSW-STM32156 firmware package is delivered with this document and contains the source code of the UART emulator with all firmware modules required to run the example.

Table 1. Applicable products

<table>
<thead>
<tr>
<th>Type</th>
<th>Part numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontrollers</td>
<td>STM32F4 series</td>
</tr>
<tr>
<td>Embedded software</td>
<td>STSW-STM32156</td>
</tr>
</tbody>
</table>
Contents

1 UART emulator description ................................................. 6
  1.1 Main features ......................................................... 6
  1.2 UART emulator block diagram ........................................ 7
  1.3 UART emulator principle .............................................. 8
    1.3.1 Data transmission ............................................... 8
    1.3.2 Single-frame transmission ...................................... 8
    1.3.3 Multiple-frame transmission ................................... 8
  1.4 Data reception ....................................................... 9
    1.4.1 Single-frame reception ......................................... 9
    1.4.2 Multiple-frame reception ....................................... 9
  1.5 Baudrate ............................................................. 10

2 Software description ..................................................... 11
  2.1 Implementation structure ........................................... 11
  2.2 Transmission ......................................................... 12
    2.2.1 Frame transmission ............................................. 12
    2.2.2 Transmission routine ........................................... 13
  2.3 Reception ........................................................... 14
    2.3.1 Frame reception ................................................ 14
    2.3.2 Reception routine ............................................... 15
  2.4 UART emulator peripherals and main functions ..................... 16
    2.4.1 Peripheral settings ............................................. 16
    2.4.2 Initialization and configuration function ..................... 16
    2.4.3 UART main functions ............................................ 17

3 Example ................................................................. 18
  3.1 Hardware requirements .............................................. 18
  3.2 Software settings ................................................... 19
  3.3 Running the example ................................................ 19
  3.4 Frame waveforms .................................................... 20

4 UART emulator CPU load and footprint ............................... 21
  4.1 CPU load ............................................................ 21
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>UART emulator memory footprint</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>Conclusion</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>Revision history</td>
<td>23</td>
</tr>
</tbody>
</table>
List of tables

Table 1. Applicable products ................................................................. 1
Table 2. Transmission/reception functions ................................................. 17
Table 3. Callback functions .................................................................. 17
Table 4. UART Emulator and UART hardware connection ......................... 18
Table 5. UART CPU Load ..................................................................... 21
Table 6. UART memory footprint ............................................................ 21
Table 7. Document revision history .......................................................... 23
List of figures

Figure 1. UART emulator block diagram .......................................................... 7
Figure 2. 9-bit data transmission waveforms .................................................. 8
Figure 3. 9-bit data reception waveforms ......................................................... 9
Figure 4. UART emulator application level view .............................................. 11
Figure 5. Frame transmission routine flowchart .............................................. 12
Figure 6. Transmission routine flowchart ......................................................... 13
Figure 7. Frame reception routine flowchart .................................................... 14
Figure 8. Reception routine flowchart ............................................................. 15
Figure 9. UART emulator and UART HW connection ..................................... 18
Figure 10. Example MDK-ARM workspaces .................................................. 19
Figure 11. UART emulator frame with no parity .............................................. 20
Figure 12. UART emulator frame with odd parity .......................................... 20
1 UART emulator description

The principle consists in emulating the UART protocol via GPIO, EXTI, timer and DMA peripherals.

1.1 Main features

The main features of the UART emulator are the following:

- Full-duplex asynchronous communications up to 115200 bps
- Programmable data word length: from 5 to 9 bits
- Flexible GPIO usage: all GPIOs can be configured as UART_TX/RX
- Configurable number of stop bits: 1 or 2 stop bits
- Parity control
  - Transmission of parity bit
  - Parity check of received data frame
- Transfer detection flags
  - Receive complete
  - Transmit complete
- Error detection flags
  - Frame error
  - Parity error
1.2 UART emulator block diagram

*Figure 1: UART emulator block diagram* gives an overview of the interaction between the hardware peripherals and the software modules that make up the UART emulator:

- **Software modules**
  They include the routines used for transmitting/receiving data and for formatting the data to be sent by DMA or stored into SRAM. Flags indicating the UART status are also available:
  - TC: Transmission Complete
    This flag is set by software when data transmission is complete.
  - RC: Reception Complete
    This flag is set by software when data reception is complete.
  - FE: Framing Error
    This flag allows frame error detection.
  - PE: Parity Error
    This flag is set by software when a parity error occurs in receive mode.

- **Hardware modules**
  They consist of all the STM32F4 peripherals involved in UART emulation:
  - GPIO: I/O ports used as transmitter or receiver pins.
  - EXTI: in reception mode, the start bit falling edge is detected via an EXTI.
  - DMA: data transfers are performed by DMA.
  - TIM: it ensures that data are transmitted and received at the required speed.
  - SRAM: data are formatted and stored in the SRAM.

---

1. The status flags and the error flags are shown in green and red, respectively.
1.3 UART emulator principle

1.3.1 Data transmission

The transmitter can send 5-, 6-, 7-, 8- or 9-bit data frames depending on user data length configuration.

*Figure 2* shows the waveforms of a 9-bit transmit emulation.

*Figure 2. 9-bit data transmission waveforms*

1.3.2 Single-frame transmission

The transmission sequence includes the following steps:

1. The CPU formats the frame to be sent to the memory.
2. The timer sends to the DMA a request to set GPIO to 0 (start bit).
3. The timer starts counting the start bit duration according to the defined baudrate.
4. When the bit duration has elapsed, the timer sends to the DMA a request to set the next bit from memory.

Once the frame transmission is complete, the TC flag (UART transmission complete) is set.

1.3.3 Multiple-frame transmission

Multiple-frame transmission is based on two buffers. When the DMA transfers the first frame into the first buffer, the CPU formats the second frame in the second buffer. After each frame transfer, the DMA TC flag must be cleared and the DMA reconfigured with the new source address (address of first or second buffer). This operation is repeated until all frames are transmitted. When this is done, the TC flag (UART transfer complete flag) is set.
1.4 Data reception

The reception is launched when the start bit is detected via an external interrupt on UART RX line.

*Figure 3* shows the waveforms of a 9-bit receive emulation.

**Figure 3. 9-bit data reception waveforms**

![Waveform Diagram]

1.4.1 Single-frame reception

Reception starts when the start bit is detected. The reception sequence includes the following steps:

1. The CPU checks if the UART is ready and the RX buffer empty.
2. The timer sends a request to the DMA to start data transfer after a half bit duration.
3. The timer starts counting the bit duration according to the defined baudrate and then sends to the DMA a request to transfer the next bit. This step is repeated until the end of the frame.

Once the frame reception is complete, the RC flag (UART reception complete) is set.

1.4.2 Multiple-frame reception

When the falling edge of the start bit is detected, the reception goes on until all frames are received and the RC flag is set.

When the DMA transfer of each frame is complete, the CPU formats data and stores them in SRAM.

*Note:* *During DMA transfer, the CPU is free for other tasks.*

*The application must assign the highest priority to the external interrupt for bit start detection.*
1.5 **Baudrate**

The UART emulator supports baudrates up to 115200 bps.

The baudrate generation is ensured by the timer using different parameters such as clock division, prescaler and period (ARR value).

The calculation of timer period (bit duration) is based on the following formula:

\[
\text{timer\_period} = \frac{\text{timer\_frequency}}{\text{UART baudrate}}
\]

**Example**

- **Configuration**
  - Timer prescaler = 0
  - Timer clock division = 0
  - AHB clock = system clock = 84 MHz
  - APB2 frequency = 21 MHz
  - UART baudrate = 9600 bps

- **Result**

  APB2 prescaler = 4

  as a result

  \[
  \text{timer\_frequency} = \frac{84}{4} \times 2 = 42 \text{ MHz}
  \]

  and

  \[
  \text{timer\_period} = \frac{42000000}{9600}
  \]
2 Software description

2.1 Implementation structure

Figure 4 shows the UART emulator software structure starting from application level.

Figure 4. UART emulator application level view
2.2 Transmission

2.2.1 Frame transmission

*Figure 5* describes the implementation of the function that sends a given number of bytes using DMA and timer.

**TxXferCount** is a counter that is incremented after the completion of each DMA transfer.

**FirstBuffer_Tx** and **SecondBuffer_Tx**: the DMA source address switches between two buffers addresses. The first buffer is used by DMA as source data, while the second buffer is used by the CPU to format data.

*Figure 5. Frame transmission routine flowchart*
2.2.2 Transmission routine

*Figure 6* gives an overview of the UART emulator transmission routine.

*Figure 6. Transmission routine flowchart*
2.3 Reception

2.3.1 Frame reception

*Figure 7* describes the implementation of the function that receives a given number of bytes via DMA and timer.

**RxXferCount**: is a counter that is incremented after the completion each DMA transfer.

**FirstBuffer_Rx** and **SecondBuffer_Rx**: the DMA destination address switches between two buffers addresses. The first buffer is used by DMA as data transfer destination, while the second buffer is used by the CPU to format data.

*Figure 7. Frame reception routine flowchart*
2.3.2 Reception routine

*Figure 8* flowchart gives an overview of the UART emulator reception routine.

*Figure 8. Reception routine flowchart*
2.4 UART emulator peripherals and main functions

2.4.1 Peripheral settings

This section describes the configuration of the peripherals used inside the emulator.

- **GPIO**
  - BSRR and IDR are used as destination and source registers for DMA transfers.
  - Two pins must be configured as input and output by the user application.
  - The input pin mode is configured as an EXTI line with falling edge detection.

- **DMA2**
  - Channel 6 and stream 1 are used for transmission.
  - Channel 6 and stream 2 are used for reception.
  - The transfer is performed by words.
  - DMA transfer complete interrupt is used at the end of frame transfers.

- **Timer 1**
  - Timer channel 1 is configured as capture compare for DMA transmit requests.
  - Timer channel 2 is configured as capture compare for DMA receive requests.
  - No clock division: CKD[1:0] =00.
  - No prescaler: PSC[15:0]=0.

- **SRAM**
  Four SRAM buffers are used to format data:
  - `uint16_t pFirstBuffer_Tx[12]` and `uint16_t pSecondBuffer_Tx[12]` are the buffers for formatting data in transmission mode.
  - `uint16_t pFirstBuffer_Rx[12]`, `uint16_t pSecondBuffer_Rx[112]` are the buffers for formatting data in reception mode.

2.4.2 Initialization and configuration function

The initialization of the UART is performed by `HAL_UART_Emul_Init` function which allows to:

- setup the following UART parameters:
  - Baudrate
  - Frame length
  - Stop bit
  - Parity.
- enable clocks for all the peripherals used: Timer, DMA, GPIOs
- configure the DMA: channel, stream, mode, TC interrupt,...
- configure the Timer: channel, period,...
2.4.3 UART main functions

This section provides a set of functions ensuring UART transmission/reception emulation.

### Table 2. Transmission/reception functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| HAL_UART_Emul_Transmit_DMA (UART_Emul_HandleTypeDef *huart, uint8_t *pData, uint16_t Size) | huart: UART Emulator handle  
pData: Pointer to data buffer  
Size: Amount of data to be sent | Sends data |
| HAL_StatusTypeDef HAL_UART_Emul_Receive_DMA (UART_Emul_HandleTypeDef *huart, uint8_t *pData, uint16_t Size) | huart: UART Emulator handle  
pData: Pointer to data buffer  
Size: Amount of data to be received | Receives data |

Callback functions are also available. They allow the user to implement his own code in the user file.

### Table 3. Callback functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__weak void HAL_UART_Emul_TxCpltCallback (UART_Emul_HandleTypeDef *huart)</td>
<td>huart: UART Emulator handle</td>
<td>This function is called at the end of transmit process.</td>
</tr>
<tr>
<td>__weak void HAL_UART_Emul_RxCpltCallback (UART_Emul_HandleTypeDef *huart)</td>
<td>huart: UART Emulator handle</td>
<td>This function is called at the end of receive process.</td>
</tr>
<tr>
<td>__weak void HAL_UART_Emul_ErrorCallback (UART_Emul_HandleTypeDef *huart)</td>
<td>huart: UART Emulator handle</td>
<td>This function is called when a communication error is detected.</td>
</tr>
</tbody>
</table>
3  Example

The example provided in STSW-STM32156 illustrates data exchange between the UART emulator and the hardware UART.

3.1  Hardware requirements

The hardware required to run this example is the following:

- Two Nucleo boards (NUCLEO-F401RE)
- Two Mini-USB cables to power the boards and to connect the Nucleo embedded ST-LINK for debugging and programming.

The connection between the two Nucleo boards through UART lines is described in Figure 9 and Table 4.

**Figure 9. UART emulator and UART HW connection**

![UART emulator and UART HW connection diagram]

**Table 4. UART Emulator and UART hardware connection**

<table>
<thead>
<tr>
<th>Nucleo board A (UART emulator)</th>
<th>Nucleo board B (hardware UART)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART TX (PC10)</td>
<td>UART RX (PA10)</td>
</tr>
<tr>
<td>UART RX (PC11)</td>
<td>UART TX (PA9)</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
</tbody>
</table>
3.2 Software settings

The project example includes two workspaces: UART_EMUL and UART_HW (see Figure 10).

To make the program work, follow the steps described below:
1. Open your preferred toolchain (EWARM or MDK-ARM).
2. Rebuild all files and load your image into target memory.
3. Run the example.

Figure 10. Example MDK-ARM workspaces

3.3 Running the example

To run the example follow the sequence below:
1. Power on the two boards.
2. Load the code in each board MCU.
3. Press the user button key on board A. The example then starts running and the UART emulator starts transmitting data.
4. The UART hardware receives the data and sends them back to UART emulator.
5. The data transmitted by the UART emulator is compared to received ones: if data do not match, the green LED (LED2) toggles continuously.

Note: For more details, refer to the readme.txt inside the firmware package.
3.4 Frame waveforms

*Figure 11* and *Figure 12* show examples of configurations of ‘A’ character transfer:

- First configuration (see *Figure 11*):
  - Baudrate: 9600 baud
  - Word length: 8 bits
  - Number of stop bit: 1 stop bit
  - Parity: none.

![Figure 11. UART emulator frame with no parity](MSv37816V1)

- Second configuration (see *Figure 12*):
  - Baudrate: 9600 baud
  - Word length: 8 bits
  - Number of stop bit: 1 stop bit
  - Parity: parity odd.

![Figure 12. UART emulator frame with odd parity](MSv37817V1)
4 UART emulator CPU load and footprint

The UART emulator uses the CPU for several tasks such as data formatting, DMA interrupt handling both for transmission and reception and EXTI interrupt handling for reception.

4.1 CPU load

The CPU load depends on whether the transmit or receive process is active. When UART full-duplex mode is active, the CPU load is increased. Refer to Table 5 for an example.

The software settings used to obtain the results given in Table 5 are the following:

- System clock: 84 MHz
- Toolchain: MDK-ARM V5.14, optimization level3(-O3) for size
- Word length: 8 bits
- One stop bit
- No parity.

Table 5. UART CPU Load

| Baudrate (bps) | Transmission | | Reception |
|----------------|--------------| |           |
|                | Load CPU (%) | MIPS | Load CPU (%) | MIPS |
| 4800           | 2.4          | 2    | 4            | 3    |
| 9600           | 5            | 4    | 8.8          | 7    |
| 115200         | 6            | 5    | 9            | 7.5  |

4.2 UART emulator memory footprint

Table 6 gives an estimate of the code size required by the UART emulator software compiled with MDK-ARM V5.14, optimization level3(-O3) for size.

Table 6. UART memory footprint

<table>
<thead>
<tr>
<th>Flash memory footprint (bytes)</th>
<th>RAM footprint (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2872</td>
<td>400</td>
</tr>
</tbody>
</table>
5 Conclusion

This application note demonstrates that the product capability can be increased by adding an emulated UART.

This solution has many advantages such as saving CPU usage by using DMA for data transfer. In addition, the user can configure any GPIO as UART transmitter or receiver.
6 Revision history

Table 7. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>30-Mar-2015</td>
<td>1</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>